تحليلات البيانات الرياضية باستخدام الذكاء الاصطناعي

بيانات رياضية تم تحليلها وشرحها باستخدام الذكاء الاصطناعي

ترجمة واعداد: د.علاء طعيمة

Sports Analytics using Artificial Intelligence

بـــمه تعالى

تحليلات البيانات الرياضية باستخدام الذكاء الاصطناعي

بيانات رياضية تم تحليلها وشرحها باستخدام الذكاء الاصطناعي

ترجمة واعداد:

د. علاء طعيمة

مقدمة المترجم

في السنوات الأخيرة، شهد عالم الرياضة تحولاً ثوريًا بفضل دمج الذكاء الاصطناعيفي تحليلات الرياضة. لم يغير هذا التقدم التكنولوجي الطريقة التي تضع بها الفرق استراتيجياتها فحسب، بل رفع أيضًا الأداء العام وقيمة الترفيهفي الأحداث الرياضية.

الذكاء الاصطناعي، والذي يُشار إليه غالبًا بالاختصار AI، هو محاكاة للذكاء البشريفي الآلات المبرمجة للتفكير والتعلم مثل البشر. وعند تطبيقه على تحليلات الرياضة، يستخدم الذكاء الاصطناعي الخوارزميات وتحليل البيانات لمعالجة كميات هائلة من المعلومات التي يتم جمعها أثناء الأحداث الرياضية. وتشمل هذه البيانات إحصائيات اللاعبين وديناميكيات اللعبة وحتى العوامل الخارجية مثل الظروف الجوية. .في هذا الكتاب، نتعمقفي عالم الذكاء الاصطناعيفي تحليلات الرياضة، ونستكشف تطبيقاته وفوائده والمستقبل المثير الذي يحمله للرياضيين والمدربين والمشجعين على حد سواء.

لقد حاولت قدر المستطاع ان اترجم المقالات والمشاريع الأكثر طرحاًفي مجال تحليل البيانات الرياضية باستخدام الذكاء الاصطناعي مع الشرح المناسب والكافي، ومع هذا يبقى عملاً بشرياً يحتمل النقص، فاذا كان لديك أي ملاحظات حول هذا الكتاب، فلا تتردد بمراسلتنا عبر بريدنا الالكتروني alaa.taima@qu.edu.iq.

نأمل ان يساعد هذا الكتاب كل من يريد ان يدخلفي مجال تحليلات الرياضة باستخدام الذكاء الاصطناعي ومساعدة القارئ العربي على تعلم هذا المجال. اسأل الله التوفيقفي هذا العمل لأثراء المحتوى العربي الذي يفتقر أشد الافتقار إلى محتوى جيد ورصينفي مجال التعلم الآلي والتعلم العميق وعلم البيانات. ونرجو لك الاستمتاع مع الكتاب ولا تنسونا من صالح الدعاء.

د. علاء طعيمة

كلية علوم الحاسوب وتكنولوجيا المعلومات

جامعة القادسية / العراق

المحتويات

0) التحليلات التنبؤية الرياضية باستخدام بايثون Sport Predictive Analytics using Python

تُحدث التحليلات التنبؤية Predictive analytics في إدارة الرياضة sports management ثورة في كيفية اتخاذ الفرق للقرارات. باستخدام بايثون، وهي لغة برمجة متعددة الاستخدامات، يمكنك تحليل البيانات الرياضية للتنبؤ بنتائج مثل نتائج المباريات وأداء اللاعبين ومخاطر الإصابة. إليك دليل بسيط للبدء:

- **فهم الأساسيات Understand the Basics**: تعرف على أساسيات بايثون ومفاهيم التعلم الآلي والمكتبات الرئيسية مثل NumPy وPandas وScikit-Learn.
- **جمع البيانات وإعدادها Collect and Prepare Data**: استخدم واجهات برمجة التطبيقات العامة Public APIs وكشط الويب web scraping وتقنيات تنظيف البيانات لجمع بياناتك وتجهيزها للتحليل.
- **استكشاف البيانات وتصورها Explore and Visualize Data**: استخدم أدوات التحليل الإحصائي والتصور لاكتشاف الأفكار.
- **هندسة الميزات Feature Engineering**: عزز بياناتك من خلال إنشاء أو تحويل الميزات لتحسين دقة النموذج.
- **بناء النماذج وتقييمها Build and Evaluate Models**: تدريب النماذج مثل الغابات العشوائية Random Forest أو الشبكات العصبية Neural Networks ومقارنة أدائها وضبطها للحصول على تنبؤات أفضل.
- **نشر نموذجك Deploy Your Model**: اجعل أداة التنبؤ الخاصة بك متاحة عبر تطبيقات الويب باستخدام منصات مثل Flask أو Streamlit أو Heroku.

يساعد هذا النهج الفرق الرياضية على التخطيط لاستراتيجيات أفضل وإدارة صحة اللاعبين واستكشاف المواهب الجديدة بشكل فعال. سواء كنت مبتدئًا أو تتطلع إلى تحسين مهاراتك، فإن بايثون يوفر لك مجموعة أدوات قوية لتحليلات الرياضة.

ما تحتاج إلى معرفته أولاً

لإنشاء أداة تتنبأ بنتائج الرياضة باستخدام بايثون، إليك ما يجب أن تعرفه بالفعل:

أساسيات بايثون

كيفية استخدام كائنات بايثون البسيطة مثل القوائم lists والخرائط maps للعمل مع NumPy وPandas للتعامل مع البيانات إنشاء مخططات باستخدام Matplotlib وSeaborn لعرض بياناتك بصريًا.

فهم التعلم الآلي

كيفية تقسيم بياناتك إلى مجموعات تدريب واختبار معرفة كيفية التحقق مما إذا كان نموذجك جيداً باستخدام أشياء مثل الدقة accuracy والضبط precision وكيفية التأكد من أن نموذجك لا يحفظ البيانات فقط (الضبط الزائد overfitting).

أدوات بايثون المهمة

- NumPy: يساعدكفي الأمور الرياضيةفي بياناتك.
- Pandas: يتيح لك ترتيب بياناتك والاطلاع عليها بسهولة Scikit-Learn ـ حيث يمكنك بناء نموذج التعلم الآلي.
- Matplotlib وSeaborn: لجعل بياناتك تبدو جيدةفي المخططات.
- Jupyter Notebook: مفيد أيضًا لتجربة التعليمات البرمجية أثناء العمل.

أين تحصل على البيانات

- **مواقع الويب التي تحتوي على إحصائيات رياضية Websites with sports stats** مثل sports-reference.com. لديهم الكثير من الأرقام حول المباريات واللاعبين السابقين.
- **مجموعات البيانات عبر الإنترنت Websites with sports stats**، مثل تلك الموجودة على Kaggle، مع بيانات رياضية جاهزة للاستخدام.
- **بيانات من التكنولوجيا القابلة للارتداء Websites with sports stats** التي توضح أداء الرياضيينفي الوقت الفعلي.

بعد إتقان هذه الأساسيات، تكون مستعداً لبدء استخدام التعلم الآليفي البيانات الرياضية باستخدام بايثون. دعنا نتعمقفي كيفية القيام بذلك، خطوة بخطوة.

فهم بيانات الرياضة

أنواع بيانات الرياضة

عندما نتحدث عن تحليلات الرياضة، فإننا نتحدثفي الواقع عن التعمقفي أنواع معينة من المعلومات. فيما يلي ملخص سريع لأنواع البيانات التي قد تستخدمها:

- **إحصائيات اللعبة Game statistics:** تتعلق هذه بالأرقام التي تظهر أثناء المباريات، مثل عدد الأهداف المسجلة، ومن يقوم بالتمريرات الحاسمة، أو المدة التي تظل فيها الكرةفي اللعب. يمكنك الحصول على هذه المعلومات من الشركات التي تجمعها أو عن طريق تدوينها بنفسك.
- **بيانات أداء اللاعبين Player performance data:** هذه تفاصيل حول مدى أداء اللاعبين، بمافي ذلك مستويات لياقتهم البدنية ومهاراتهم وكيفية اتخاذ القراراتفي الملعب. يمكن أن تأتي هذه البيانات من الأدوات التي يرتديها الرياضيون أو مراجعات الفيديو أو الاختبارات التي يجرونها.
- **تقارير الإصابات Injury reports:** تتضمن هذه معلومات عن أي إصابات يتعرض لها اللاعبون ومدة غيابهم ومدى احتمالية تعرضهم للإصابة مرة أخرى. إنها ضرورية للحفاظ على صحة اللاعبين.
- **بيانات الكشف عن المواهب Scouting data:** عندما تبحث الفرق عن لاعبين جدد محتملين، فإنها تنظر إلى هذه البيانات. وهي تغطي أشياء مثل سرعة اللاعب ومدة ركضه وقدراته الإجمالية.
- **مقاييس الأعمال Business metrics:** تتعلق هذه المقاييس بالجانب المالي للرياضة، مثل عدد التذاكر المباعة، وكمية البضائع المباعة، وعدد زيارات الموقع الإلكتروني. وهي تساعدفي التخطيط للميزانية.

ستكون البيانات المختلفة مهمة اعتمادًا على ما تحاول القيام به. للنظرفي كيفية أداء الفرق أو اللاعبين في المستقبل، فإن إحصائيات المباريات واللاعبين هي ما تحتاجه عادةً. للحفاظ على صحة اللاعبين، تعد بيانات الإصابات أمرًا أساسيًا. وإذا كنت تبحث عن مواهب جديدة، فستحتاج إلى هذه التفاصيل المحددة.

مصادر البيانات

من أين تأتي كل هذه البيانات الرياضية؟ إليك بعض الأماكن:

- **واجهات برمجة التطبيقات العامة Public APIs:** تقدم مواقع الويب مثل sports-reference.com وصولاً مجانيًا إلى بياناتها من خلال واجهات برمجة التطبيقات، والتي تعد بمثابة بوابات إلى معلوماتها. إنها سهلة الاستخدام ولكنها قد لا توفر لك كل ما تحتاجه.
- **الخدمات المميزة Premium services:** هذه خيارات مدفوعة تقدم مجموعات بيانات مفصلة حقًا. يمكن أن تكون باهظة الثمن ولكنها منظمة جيدًا للتحليل العميق.
- **كشط الويب Web scraping:** هذا يعني سحب البيانات مباشرة من مواقع الويب. يمنحك الكثير من الحرية ولكن قد يكون من الصعب القيام به.

- **الأجهزة القابلة للارتداء Wearable devices:** تتبع هذه الأدوات جميع أنواع الإحصائيات البدنية للرياضيين، مثل مدى تعبهم أو مقدار الضغط الذي تتعرض له عضلاتهم. إنها محددة للغاية ولكنها مفيدة للغاية لتحليلات معينة.

سيعتمد اختيارك لمصدر البيانات على ما تحاول تحقيقه. بالنسبة للمشاريع البسيطة، قد تكون واجهات برمجة التطبيقات المجانية كافية. ولكن إذا كنت تقوم بشيء أكثر تعقيداً، مثل بناء نموذج التعلم الآلي، فقد تحتاج إلى البيانات التفصيلية من الخدمات المدفوعة.

تذكر فقط أنه مهما كانت البيانات التي تستخدمها، تأكد من أنه يُسمح لك باستخدامها. قد تكون لبعض البيانات الرياضية قواعد حول كيفية استخدامها.

إعداد بيئة بايثون

يتضمن تجهيز الكمبيوتر لبناء أداة تحليل تنبؤية لإدارة الرياضة باستخدام بايثون بضع خطوات. دعنا نوضحها:

تثبيت بايثون والمكتبات المطلوبة

- **أولاً**، قم بتنزيل أحدث إصدار من بايثون من python.org. تأكد من السماح له بإضافة بايثون إلى مسار نظام الكمبيوتر الخاص بك أثناء التثبيت.
- **بعد ذلك**، افتح موجه الأوامر أو المحطة الطرفية واكتب `pip install numpy pandas scikit-learn matplotlib seaborn jupyter`. يلتقط هذا الأمر جميع مكتبات بايثون المهمة التي سنستخدمها.

إعداد بيئة افتراضية (موصى بها)

يساعد إنشاء بيئة افتراضية لمشروعكفي الحفاظ على كل شيء منظمًا ومنفصلًا عن المشاريع الأخرى. إليك كيفية القيام بذلك:

1. في المحطة الطرفية، انتقل إلى مجلد مشروعك واكتب `python -m venv my_env` لإنشاء بيئة جديدة.
2. للبدءفي استخدامه، اكتب `my_env\Scripts\activate` على Windows أو `source my_env/bin/activate` على Mac/Linux.
3. تذكر تنشيط بيئتك الافتراضية كلما كنت تعمل على مشروعك.

تثبيت Jupyter Notebook (اختياري)

يعد Jupyter Notebook أداة رائعة لتجربة كود بايثون أثناء العمل. لتثبيته:

1. مع تنشيط بيئتك، اكتب `pip install jupyter`.

2. قم بتشغيل Jupyter بكتابة `jupyter notebook`، وسيتم فتحهفي متصفح الويب الخاص بك.
3. يمكنك الآن إنشاء Notebook جديدة لكتابة كود بايثون واختباره.

وهذا كل شيء! مع إعداد بايثون وجميع المكتبات الضرورية مثل Pandas وNumPy وScikit-Learn، تكون جاهزًا لبدء إنشاء نماذج لتحليلات الرياضة.

جمع البيانات واستخراجها

الوصول إلى واجهات برمجة التطبيقات العامة

إليك كيفية الحصول على بيانات رياضية من موقع ويب يتشاركها بحرية. يوضح لك هذا المثال كيفية الحصول على بيانات حول مباريات NFL باستخدام بايثون:

```
import requests
import pandas as pd

url = "https://api.sportsdata.io/v3/nfl/scores/json/GamesByWeek/2022/1"

params = {
  "key": "YOUR_API_KEY"
}

response = requests.get(url, params=params)
data = response.json()

df = pd.DataFrame(data)
print(df.head())
```

يصل هذا الكود إلى موقع الويب، ويطلب البيانات باستخدام رمز الوصول الخاص بك (مفتاح API)، ثم يضع البيانات التي يحصل عليها مرة أخرىفي جدول (إطار بيانات dataframe) حتى تتمكن من العمل معه. سيحتوي هذا الجدول على معلومات حول مباريات الدوري الوطني لكرة القدم الأمريكية NFL من الأسبوع الأول من موسم 2022.

هناك الكثير من مواقع الويب مثل sportsdata.io التي تشارك بيانات رياضية. بمجرد معرفة البيانات التي تريدها، يمكنك العثور على موقع ويب يحتوي عليها واستخدام واجهة برمجة التطبيقات الخاصة به للحصول عليها.

إنشاء أداة استخراج بيانات من الويب

إذا لم تكن البيانات التي تحتاج إليها متاحة من خلال واجهة برمجة التطبيقات، فيمكنك الحصول عليها مباشرةً من موقع ويب باستخدام أداة استخراج البيانات من الويب web scraping. فيما يلي مثال بسيط باستخدام بايثون للحصول على بيانات من موقع ويب حول إحصائيات كرة القدم:

```
import requests
from bs4 import BeautifulSoup
```

```
url = "https://www.pro-football-reference.com/years/2021/rushing.htm"

response = requests.get(url)
soup = BeautifulSoup(response.content, 'html.parser')
table = soup.find('table', {'id': 'rushing'})

headers = [header.text.strip() for header in table.find_all('th')]
rows = []
for row in table.find_all('tr')[1:]:
    rows.append([val.text.strip() for val in row.find_all('td')])

import pandas as pd
df = pd.DataFrame(rows, columns=headers)
print(df.head())
```

يطلب هذا الكود من موقع الويب بياناته، ويجد الجدول المحدد الذي نهتم به، ثم يستخرج رؤوس وصفوف الجدول. ويضع هذه المعلوماتفي إطار بيانات، والذيفي هذه الحالة، يحتوي على إحصائيات حول أداء لاعبي NFLفي الاندفاعفي عام 2021.

يتيح لك كشط الويب Web scraping جمع بيانات محددة قد لا تتم مشاركتها من خلال واجهات برمجة التطبيقات APIs. ويتضمن ذلك النظرفي كود موقع الويب للعثور على مكان تخزين البيانات، ولكنها طريقة مفيدة لجمع معلومات مفصلة لمشاريعك.

تنظيف البيانات ومعالجتها مسبقًا

قبل أن نتمكن من استخدام بيانات الرياضة للتنبؤ بالنتائج، نحتاج إلى تنظيفها قليلًا. وهذا يعني إصلاح أي مشكلات حتى تتمكن نماذج التعلم الآلي لدينا من فهمها بشكل أفضل. فيما يلي دليل بسيط حول كيفية القيام بذلك باستخدام بايثون:

التعامل مع القيم المفقودة

في بعض الأحيان، قد تكون البيانات مفقودة missing. قد يكون هذا بسبب عدم تسجيلها بشكل صحيح أو وجود خطأ. فيما يلي طريقة سريعة للتحقق من البيانات المفقودة وإصلاحها باستخدام Pandas:

```
import pandas as pd
import numpy as np

df = pd.read_csv("data.csv")

# Check for missing values
print(df.isnull().sum())

# Drop rows with missing targets
df.dropna(axis=0, subset=["target"], inplace=True)

# Fill numeric columns with mean
df.fillna(df.mean())
```

يمكنك إزالة الصفوف التي تحتوي على معلومات مفقودة أو ملء الفراغات بالقيم المتوسطة.

إصلاح القيم المتطرفة

القيم المتطرفة Outliers هي نقاط بيانات غريبة لا تتناسب مع بقية البيانات. ويمكنها أن تفسد توقعاتنا. وإليك كيفية اكتشافها وإصلاحها:

```
import pandas as pd

df = pd.read_csv("data.csv")

for name in numeric_cols:
  Q1 = df[name].quantile(0.25)
  Q3 = df[name].quantile(0.75)
  IQR = Q3 - Q1

  df.loc[df[name] < (Q1 - 1.5 * IQR), name] = Q1 - 1.5 * IQR
  df.loc[df[name] > (Q3 + 1.5 * IQR), name] = Q3 + 1.5 * IQR
```

تبحث هذه الطريقة عن القيم المتطرفة وتغيرها إلى قيم أكثر شيوعًا.

تحويل أنواع البيانات

في بعض الأحيان، لا يكون نوع البيانات الذي تخمنه Pandas هو ما نريده لنماذجنا. وإليك كيفية تغييره:

```
df["column"] = df["column"].astype(float)
df["category"] = df["category"].astype('category')
```

يضمن هذا التعامل مع الأرقام numbers كأرقام والتعامل مع الفئات categories كفئات.

هندسة الميزات

يمكننا أيضًا إنشاء أعمدة جديدة قد تساعد نماذجنافي تقديم تنبؤات أفضل. على سبيل المثال:

```
df["age_bucket"] = pd.cut(df["age"], bins=[0, 20, 30, 40, np.inf],
                          labels=["under 20", "20-30", "30-40", "over 40"])
```

يؤدي هذا إلى تغيير رقم (مثل العمر age) إلى فئة (مثل الفئة العمرية age group)، وهو ما قد يكون أسهل بالنسبة لنماذجنافي الاستخدام.

من خلال تنظيف بياناتنا وتحضيرها بهذه الطريقة، نسهل على أدوات مثل Scikit-Learn وXGBoost القيام بسحرهافي التحليلات التنبؤية، وخاصةفي إدارة الرياضة.

تحليل البيانات الاستكشافي

التحليل الإحصائي

قبل أن نتعمقفي إجراء التنبؤات، دعنا نتعرف على بياناتنا. فكرفي هذا الأمر كأنك تتعرف على صديق جديد. يمكننا استخدام Pandas وNumPy، وهما أداتانفي بايثون، للنظرفي أشياء بسيطة مثل القيم المتوسطة، ومدى انتشار البيانات، وما إذا كانت بعض المعلومات تميل إلى التحرك معًا. إليك طريقة سريعة للقيام بذلك:

```
import pandas as pd
import numpy as np

df = pd.read_csv("data.csv")

# Average values
print(df.mean())

# How spread out the data is
print(df.std())

# If columns move together
print(df.corr())
```

يمنحنا هذا صورة أساسية، مثل كيفية أداء الفرق أو اللاعبين بشكل عام، وما إذا كان لعب المزيد من المباريات يعني تسجيل المزيد من النقاط على سبيل المثال.

يمكننا أيضًا النظرفي أرقام أخرى مثل القيمة المتوسطة middle value (المتوسط median)، وأعلى وأدنى الدرجات، وكيفية توزيع البيانات عبر فئات مختلفة، مثل الفرق teams.

يساعدنا فهم بياناتنا على اتخاذ خيارات أكثر ذكاءً لاحقًا عندما نتوقع النتائج.

التصور المرئي للبيانات

يمكن أن يساعدنا رؤية بياناتنافي تحديد الاتجاهات trends والأنماط patterns. يمكننا استخدام أدوات مثل Matplotlib وSeaborn لهذا الغرض. فيما يلي بعض الطرق البسيطة لتصور بياناتنا:

```
import matplotlib.pyplot as plt
import seaborn as sns

# Comparing two things
sns.scatterplot(x="games_played", y="points", data=df)

# Seeing how often something happens
plt.hist(df["assists"])

# Checking how things are related
sns.heatmap(df.corr(), annot=True)

# Comparing groups
sns.barplot(x="team", y="points", data=df, ci=None)

plt.show()
```

تتيح لنا هذه الأدوات إنشاء مخططات تشتت scatter plots لمقارنة الأشياء، ومخططات بيانية histograms لمعرفة مدى تكرار حدوث شيء ما، وخرائط حرارية heatmaps للتحقق من العلاقات، ومخططات شريطية bar plots لمقارنة المجموعات. يمكننا تغيير مظهر هذه المخططات لجعلها أكثر وضوحًا.

إن تصور البيانات Visualizing data أمر أساسي لملاحظة التفاصيل المهمة التي قد تساعدنافي التنبؤ.

هندسة الميزات

تعتبر هندسة الميزات Feature engineering جزءًا أساسيًا من إنشاء نماذج تنبؤية للرياضة. إنها تتعلق باختيار أفضل أجزاء من بياناتنا الخام وتغييرها حتى تتمكن نماذجنا من تخمين النتائج بدقة أكبر.

فيما يلي بعض الطرق البسيطة للقيام بهندسة الميزات باستخدام بيانات الرياضة sports data:

إنشاء ميزات مركبة

يمكننا مزج قطع مختلفة من البيانات لإنشاء ميزات جديدة وأكثر إفادة. على سبيل المثال، يمكننا إنشاء ميزة "قوة التهديف scoring power" من خلال إضافة الأهداف والتمريرات الحاسمة والتسديدات. وهذا يعطي صورة أكثر اكتمالاً.

```
df['scoring_power'] = df['goals'] + df['assists'] + df['shots']
```

ترميز البيانات التصنيفية

تعمل نماذجنا بشكل أفضل مع الأرقام مقارنة بالفئات categories مثل "أسماء الفرق team names". لذا، نقوم بتحويل هذه البيانات إلى أرقام باستخدام عملية تسمى الترميز واحد ساخن one-hot encoding:

```
team_encoded = pd.get_dummies(df['team'])
```

إضافة ميزات تعتمد على الوقت

قد يتغير أداء الفرق الرياضية أو اللاعبين بمرور الوقت. لذا، فإن إضافة تفاصيل مثل الشهر أو الموسم قد يُظهر لنا الأنماط:

```
df['month'] = df['date'].apply(lambda x: x.month)
```

إنشاء ميزات متأخرة

يمكن للأداء السابق أن يعطي أدلة حول النتائج المستقبلية. تتطلع الميزات المتأخرة Lagging features إلى البيانات السابقة للمساعدةفي التنبؤ بما سيأتي بعد ذلك:

```
df['rolling_avg'] = df['goals'].shift(1).rolling(window=3).mean()
```

تحويل السجل للبيانات المشوهة

إذا كانت بعض البياناتفي كل مكان، فإننا نستخدم تحويل السجل log transform لتسويتها. وهذا يمنع أي قطعة بيانات واحدة من أن يكون لها تأثير كبير:

```
df['salary'] = np.log(df['salary'])
```

هناك العديد من الطرق الذكية الأخرى للقيام بذلك، ولكن البدء بهذه الأساسيات يمكن أن يساعد نماذجك حقًافي الحصول على فهم أفضل لما يحدثفي الرياضة وإجراء تخمينات أكثر دقة حول الألعاب المستقبلية أو أداء اللاعبين. الحيلة هي التعرف حقًا على بياناتك.

بناء نماذج تنبؤية

تدريب النموذج

لنبدأ بكيفية تدريب نموذج يمكنه التنبؤ بنتائج الرياضة باستخدام بايثون. سنستخدم طريقة تسمى نموذج الغابة العشوائية random forest model. إليك طريقة بسيطة للقيام بذلك باستخدام Scikit-Learn، وهي أداةفي بايثون للتعلم الآلي:

```
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
```

تقسيم البيانات

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2)
```

النموذج

```
clf = RandomForestClassifier()
```

التدريب

```
clf.fit(X_train, y_train)
```

التنبؤ

```
y_pred = clf.predict(X_test)
```

```
This code means we're dividing our data into two parts, training the model
with one part, and then seeing how well it does with the other part. The
model learns from the training data and then tries to predict the outcomes
in the test data.

The same steps apply if we're using other models, like neural networks with
Keras or SVM models. The main tasks are preparing the data, choosing a
model, training it with our data, and then using it to guess outcomes.

### Model Comparison

Let's compare how two different models did when predicting the same sports
outcomes:

| Model | Accuracy | Precision | Recall |
|-|-|-|-|
| Random Forest | 0.82 | 0.81 | 0.83|
| Neural Network | 0.85 | 0.84 | 0.86 |

This table shows that the neural network model did a bit better than the
random forest model in terms of accuracy, precision, and recall. This means
it guessed the outcomes more correctly more often.

When we compare models, it's good to look at different scores like
accuracy, precision, and recall. Accuracy tells us how often the model's
guesses were right. Precision shows how reliable the model's 'yes' guesses
are. Recall tells us how good the model is at finding all the 'yes' cases.

We can also use tools like the ROC curve and confusion matrices to see if
some models are better with certain types of data. The best approach is to
try a few models and use cross-validation (a way to test the model's
ability to predict new data) to find the best one for our needs.

## Model Evaluation and Tuning
```

```
Checking and tweaking our machine learning models is a big part of making
sure they do a great job at predicting sports outcomes. Here's how to make
sure your model is as accurate as it can be:

### K-Fold Cross Validation

Cross validation is like a test run for your model to see how well it can
predict new, unseen data. It involves splitting your data into several
parts, training your model on most of these parts, and testing it on the
remaining part.

```python
from sklearn.model_selection import KFold

kf = KFold(n_splits=5)

for train, test in kf.split(X):
    model.fit(X[train], y[train])
    prediction = model.predict(X[test])
    # Evaluate predictions
```

من خلال تدوير الجزء المستخدمفي البيانات للاختبار، نحصل على صورة أوضح لكيفية أداء النموذج بشكل عام.

البحث الشبكي لضبط المعلمات الفائقة

تحتوي النماذج على إعدادات تسمى المعلمات الفائقة hyperparameters والتي يمكنها تغيير مدى نجاحها. يمكن العثور على أفضل الإعدادات باستخدام شيء يسمى البحث الشبكي grid search.

```
from sklearn.model_selection import GridSearchCV

params = {
    'n_estimators': [10, 50, 100],
    'max_depth': [None, 3, 5]
}

grid_search = GridSearchCV(RandomForestClassifier(), params, cv=5)
grid_search.fit(X_train, y_train)

print(grid_search.best_params_)
print(grid_search.best_score_)
```

تختبر هذه الطريقة مجموعات مختلفة من الإعدادات للعثور على أفضلها.

تحليل مصفوفة الارتباك

مصفوفة الارتباك confusion matrix عبارة عن جدول يوضح لنا التنبؤات الصحيحة وغير الصحيحة للنموذج:

```
Predicted Yes   |     Predicted No
Actual Yes      |     True Positive | False Negative
Actual No       |     False Positive | True Negative
```

إن رؤية الأخطاء التي يرتكبها النموذج تساعدنا على فهم كيفية تحسينه.
```

الموازنة بين الدقة والاستدعاء

في بعض الأحيان، قد يعني جعل النموذج أكثر موثوقية (الدقة precision) أنه قد يفوت بعض التخمينات الصحيحة (الاستدعاء recall). نحتاج إلى إيجاد توازن جيد اعتمادًا على ما نحاول التنبؤ به.

يعد التأكد من فحص نماذجنا وتعديلها بعناية أمرًا بالغ الأهمية للحصول على تنبؤات جيدة. يساعدنا استخدام أساليب مثل التحقق المتبادل cross-validation وإعدادات الضبطفي اختيار أفضل نموذج لبياناتنا الرياضية وإتقانه.

نشر النموذج

بمجرد حصولك على نموذج تعلُّم آلي يمكنه التنبؤ بنتائج الرياضة، فإن الخطوة التالية هي جعله متاحًا للاستخدام من قِبل الأشخاص. فيما يلي بعض الطرق للقيام بذلك باستخدام أدوات تعمل بشكل جيد مع بايثون.

Flask

Flask هي طريقة بسيطة لتحويل نماذجك إلى تطبيقات ويب.

إليك كيفية القيام بذلك:

1. احفظ نموذجك كملف pkl. حتى يمكن استخدامه لاحقًا.
2. استخدم Flask لإنشاء طرق للمستخدمين لإرسال البيانات إلى نموذجك واستعادة التوقعات.
3. ضع تطبيقك ونموذجك على منصة سحابية مثل AWS أو GCP أو Azure حتى يمكن الوصول إليها من أي مكان.

تتيح هذه الطريقة للمستخدمين التفاعل مع نموذجك عبر الويب.

Streamlit

Streamlit هي أداة تساعدك على إنشاء تطبيقات الويب ومشاركتها بسرعة، وخاصة لنماذج التعلم الآلي.

للبدء:

1. قم بتحميل نموذجكفي Streamlit.
2. قم بإنشاء عناصر سهلة الاستخدام مثل أشرطة التمرير للأشخاص لإدخال البيانات.
3. اعرض توقعات النموذج باستخدام الرسوم البيانية أو الأرقام.
4. شارك تطبيقك من خلال خدمة الاستضافة Streamlit.

يسهل Streamlit على الفرق رؤية كيفية تأثير تغيير المدخلات على التوقعات.

Heroku

Heroku عبارة عن منصة سهلة الاستخدام لنشر التطبيقات، بمافي ذلك تلك المبنية باستخدام بايثون ومكتبات التعلم الآلي.

للنشر على Heroku:

1. تأكد من أن تطبيقك ونموذجك يتضمنان جميع الملفات والمعلومات الضرورية.
2. قم بربط Heroku بمستودع GitHub الخاص بك حيث يوجد كود تطبيقك.
3. استخدم زر النشرفي Heroku لبناء تطبيقك وبدء تشغيله.
4. اضبط الإعدادات للتعامل مع المزيد من الزوار إذا لزم الأمر.

تُعد خطة Heroku المجانية مثالية للمشاريع الصغيرة أو العروض التوضيحية.

من خلال جعل نماذجكفي متناول الجميع بهذه الطريقة، يمكن للفرق والمنظمات الرياضية استخدام توقعاتك بسهولة لاتخاذ قرارات أفضل، سواء كان ذلك لاستراتيجية اللعبة أو إدارة اللاعبين.

الاستنتاج

لا يزال استخدام بايثون للتنبؤ بنتائج الرياضة أمرًا جديدًا إلى حد ما، لكنه يُظهِر بالفعل الكثير من النجاح. لقد أظهر لك هذا الدليل كيفية استخدام بايثون وأدواته للعمل مع بيانات الرياضة وإجراء التنبؤات.

فيما يلي النقاط الرئيسية التي يجب تذكرها:

- تنتج الرياضة الكثير من البيانات التي يمكن استخدامها لإجراء التنبؤات. يمكنك الحصول على هذه البيانات من مواقع الويب، وكشط الويب، والأدوات القابلة للارتداء، وغيرها من المصادر.
- تساعدك أدوات مثل Pandas وNumPy وScikit-Learnفي بايثون على جمع هذه البيانات وتنظيفها وتحليلها للعثور على رؤى مفيدة.
- هندسة الميزات هي تقنية تقوم فيها بتعديل البيانات الخام حتى يتمكن الكمبيوتر من فهمها والتعلم منها بشكل أفضل.
- يمكنك استخدام أنواع مختلفة من النماذج، مثل الغابات العشوائية والشبكات العصبية، لتخمين الأداء المستقبلي.
- من المهم مقارنة هذه النماذج لاختيار الأفضل. يتيح جعل تنبؤاتك متاحة من خلال تطبيقات الويب للفرق استخدام هذه المعلومات لتحسين تدريبها واستراتيجيتها وتخطيطها.

مع حصولنا على بيانات أفضل وتحسين نماذجنا، من المرجح أن تصبح التحليلات التنبؤيةفي الرياضة أداة رئيسية للفرق التي تتطلع إلى التقدم. وفي حين أن بناء نظام كامل يتطلب الكثير من العمل، فإن هذا الدليل من شأنه أن يمنحك فهمًا أساسيًا لكيفية فتح بايثون الباب أمام التنبؤات الرياضية.

بالنظر إلى المستقبل، قد نرى نماذج تجمع بين البيانات من التكنولوجيا القابلة للارتداء وتحليل الفيديو والإحصائيات التقليدية للتنبؤ بموعد إصابة اللاعب، بناءً على أشياء مثل مدى تعبه. يمكن أن يساعد هذا المدربينفي التخطيط لتدريبات أفضل واستراتيجيات للعبة للحفاظ على صحة اللاعبين وأدائهم الجيد.

الآن هو الوقت المناسب للبدءفي اللعب ببيانات الرياضة بنفسك ومعرفة ما يمكنك اكتشافه!

أسئلة ذات صلة

هل يمكن استخدام بايثون للتحليلات التنبؤية؟

نعم، بايثون هو الخيار الأفضل للتحليلات التنبؤية لأنه يحتوي على الكثير من الأدوات للعمل مع البيانات وإجراء التنبؤات. تساعدك المكتبات مثل Pandas وNumPy وScikit-Learn وKeras وPyTorch وTensorFlowفي تجهيز البيانات وبناء النماذج والتنبؤ بالنتائج. بايثون سهل الاستخدام، مما يجعله رائعًا لتجربة طرق التنبؤ المختلفة بسرعة.

هل يستخدم بايثون في تحليلات الرياضة؟

أصبحت بايثون أكثر شهرةفي تحليلات الرياضة لأنها جيدة حقًافي التعامل مع مجموعات البيانات الكبيرة والمعقدة. لديها أدوات مثل Pandas وNumPy لتنظيم البيانات وأدوات أخرى مثل Scikit-Learn وPyTorch وTensorFlow لإجراء تنبؤات حول أداء اللاعبين ونتائج المباريات ومخاطر الإصابة. بايثون مرنة، مما يجعلها مثالية للتحليل العميقفي الرياضة.

كيف تصنع نموذجًا تنبؤيًا في بايثون؟

لبناء نموذج تنبؤي في بايثون، اتبع الخطوات التالية:

1. استخدم Pandas لتنظيف بياناتك، مثل إصلاح القيم المفقودة.
2. انظر إلى بياناتك واعمل صورًا لها لفهمها بشكل أفضل.
3. جهّز بياناتك للنموذج.
4. قسّم بياناتك إلى أجزاء تدريب واختبار.
5. اختر نموذجًا لاستخدامه، مثل الانحدار الخطي أو الغابة العشوائية.
6. درّب نموذجك باستخدام بيانات التدريب.

7. تحقق من مدى أداء نموذجك مع بيانات الاختبار.
8. اضبط نموذجك لجعله أكثر دقة.
9. استخدم نموذجك.

ما لغة البرمجة المستخدمة في تحليلات الرياضة؟

تستخدم تحليلات الرياضة بشكل أساسي Python وR وSQL و Excel VBA. بايثون هي الأكثر شيوعًا لأنه رائع للتعلم الآلي وسهل الاستخدام. يستخدم R للتحليل الإحصائي الأعمق. يساعد SQL في إدارة مجموعات البيانات الضخمةفي قواعد البيانات. Excel سهل الوصول إليه ولكنه لا يمكنه التعامل مع قدر كبير من البيانات أو النماذج المعقدة مثل الآخرين. غالبًا ما يستخدم المحللون مزيجًا من هذه الأدوات للعمل مع بيانات الرياضة.

المصدر:

https://dataheadhunters.com/academy/how-to-build-a-predictive-analytics-tool-in-python-for-sports-management/

1) إطار عمل تحليلات الرياضة في بايثون Sports Analytics Framework in Python

كرة القدم football هي الرياضة الأكثر شعبيةفي جميع أنحاء العالم. إنها لعبة عالمية تربط كل شخص تقريبًا على هذا الكوكب. لقد كانت جزءًا من حياتي منذ أن كنت طفلاً. لقد شاهدت كل مباراة تقريبًا حتى الآن.

على مدار الشهرين الماضيين، كنت أعمل على مشروع يسمى تحليلات الرهان على مباريات الرياضة betting analytics on sports (كرة القدم). أريد تنفيذ إطار عمل Framework بسيط للمراهنةفي بايثون. يتكون من جزأين وهما معالجة البيانات مسبقًا data-preprocessing وتحليل البيانات الاستكشافي exploratory data analysis. في الجزء الاول، أريد أن أتناول كيفية معالجة بيانات كرة القدم مسبقًا وتحويلها إلى عمل.

ما هو تحليل الرياضة (المراهنة) وحالات الاستخدام؟

تحليل الرياضة Sports analytics هوفي الواقع مجال واسع وهناك العديد من الأنواع المختلفة من الأدوار التي يمكن أن يقوم بها محترف تحليل الرياضة. أريد أن أتطرق إلى الفرق بين تحليل البيانات data analysis وتحليل المراهنة betting analytics.

العثور على مجموعة بيانات

قد يكون من الصعب جدًا العثور على مجموعة بيانات محددة لمباريات كرة القدم، ولكنني وجدت موقعًا إلكترونيًا يوفر نتائج كرة قدم حديثة وتاريخية تتضمن 27 نتيجة موسم و20 إحصائية لمباريات الموسم "http://football-data.co.uk/data.php". تتوفر جميع الدوريات العالمية.

تحتوي مجموعات البيانات على 22 متغيرًا. فقط الدوري الإنجليزي الممتاز يحتوي على عمود للحكام referee column. يحتوي قاموس البيانات على "http://football-data.co.uk/notes.txt". فيما يلي بعض جداول قاموس المتغيرات variable dictionary table؛

- FTHG: أهداف الفريق المضيفي الوقت الكامل Full-Time Home Team Goals.
- FTAG: أهداف الفريق الزائرفي الوقت الكامل Full-Time Away Team Goals.
- FTR: النتيجةفي الوقت الكامل (H = فوز الفريق المضيف Home Win، D = تعادل Draw، A = فوز الفريق الزائر Away Win).
- HS: تسديدات الفريق المضيف Home Team Shots Home Team Shots.
- HST: تسديدات الفريق المضيف على المرمى Home Team Shots on Target.

	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
0	09/08/2019	20:00	Liverpool	Norwich	4	1	H	4	0	H	15	12	7	5	9	9	11	2	0	2	0	0
1	10/08/2019	12:30	West Ham	Man City	0	5	A	0	1	A	5	14	3	9	6	13	1	1	2	2	0	0
2	10/08/2019	15:00	Bournemouth	Sheffield United	1	1	D	0	0	D	13	8	3	3	10	19	3	4	2	1	0	0
3	10/08/2019	15:00	Burnley	Southampton	3	0	H	0	0	D	10	11	4	3	6	12	2	7	0	0	0	0
4	10/08/2019	15:00	Crystal Palace	Everton	0	0	D	0	0	D	6	10	2	3	16	14	6	2	2	1	0	1

مجموعة بيانات نموذجية من الدوري الإنجليزي الممتاز

النوع الأكثر شعبية من الرهانات على كرة القدم، والهدف منه هو التنبؤ بالنتيجة النهائية للمباراة، سواء فوز الفريق المضيف (1)، أو التعادل (X)، أو فوز الضيوف (2). ولكن، لديك الآن الكثير من الخيارات للعب. لقد قمت بعمل قائمة شهيرة أدناه. على سبيل المثال، يتم وضع رهان أكثر/أقل من 2.5 هدففي سوق 2.5 هدف. هنا يمكنك تحديد أحد أمرين ليحدث، أقل من 2.5 هدف يعني هدفين أو أقل، أكثر من 2.5 هدف يعني ثلاثة أهداف أو أكثر.

جزء الكود

أستطيع أن أعرض بعض الأمثلة حول كيفية برمجة خيارات الرهان مثل عمل إحصائيات أقل/أكثر من 2.5 هدففي الشوط الأول أو النهائي، والأهداف المتبادلة، وMS_1_under_1_5، والتغيير إلى العمود الرقمي numeric column لكل استراتيجية فريق.

```
##############################################################################
#################
def two_5_Under_Half(row):
    '''
    Returns the condition 2.5 under half or not
            Parameters:
                    row : each row
            Returns:
                    0 or 1: binary number
    '''
    if row['HTHG'] < 3 and row['HTAG'] < 3:
        return 1
    else:
        return 0
df['2.5_under_half'] = df.apply(lambda row : two_5_Under_Half(row),axis=1)
##############################################################################
#################
def two_5_Above_Final(score):
    '''
    Returns the condition 2.5 above final or not
            Parameters:
                    row : each row
            Returns:
                    0 or 1: binary number
    '''
    if score > 2:
        return 1 #
    else:
        return 0 #
df['2.5_above_final'] = df['final_result_sum'].apply(two_5_Above_Final)
##############################################################################
#################
def MS_1_under_1_5(row):
    '''
```

```
    Returns the condition MS_1_under_1_5 or not
            Parameters:
                    row : each row
            Returns:
                    0 or 1: binary number
    '''
    if row['FTR'] == 1 and row['final_result_sum'] < 2 :
        return 1
    else:
        return 0
df['MS_1_under_1_5'] = df.apply(lambda row : MS_1_under_1_5(row), axis=1)
##########################################################################
#################
#both teams to score
s1 = pd.to_numeric(df['FTHG'], errors='coerce')
s2 = pd.to_numeric(df['FTAG'], errors='coerce')

m1 = (s1 >0) & (s2 >0)
m2 = (s1 <0) & (s2 <0)

masks=[m1,m2]
vals = [1,0]

df['mutual_goal'] = np.select(masks,vals)
##########################################################################
#################
# Change to categoric column to numeric
df.loc[df['HTR']=='H','HTR']=1
df.loc[df['HTR']=='D','HTR']=0
df.loc[df['HTR']=='A','HTR']=2
##########################################################################
#################
```

يمكنك تطبيق نفس تنسيق البرمجة. يتم إنشاء 59 متغيرًا. بعد مجموعة البيانات الأولية، يتم إنشاء ملف GIF أدناه.

	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
0	2019-08-09	20:00	Liverpool	Norwich	4	1	1	4	0	1	15	12	7	5	9	9	11	2	0	2	0	0
1	2019-08-10	12:30	West Ham	Man City	0	5	2	0	1	2	5	14	3	9	6	13	1	1	2	2	0	0
2	2019-08-10	15:00	Bournemouth	Sheffield United	1	1	0	0	0	0	13	8	3	3	10	19	3	4	2	1	0	0
3	2019-08-10	15:00	Burnley	Southampton	3	0	1	0	0	0	10	11	4	3	6	12	2	7	0	0	0	0
4	2019-08-10	15:00	Crystal Palace	Everton	0	0	0	0	0	0	6	10	2	3	16	14	6	2	2	1	0	1

جزء تحليل الفريق

بعد الانتهاء من معالجة البيانات، يتم إنشاء دالة تسمى "الاحتمالية probability". وهي دالة تنشئ بطاقة معلومات لكل فريق. تعرض هذه الدوال النسبة المئوية للفوز والخسارة والتعادل وخيارات الرهان الأخرى بالتفصيل. يمكنك العثور على نسبة أنواع الرهانات بأسلوب مرتب.

Ipywidgets هي أدوات HTML تفاعلية لدفاتر Jupyter التي توفر سهولة الوصول إلى المعلومات. يكتسب المستخدمون التحكمفي بياناتهم ويمكنهم تصور التغييراتفي البيانات. لدي شريط تمرير slider وقائمة منسدلة dropdown menu بقائمة فريق يمكنك إدارتها بسهولة.

```
def probability(df,team,number):
    '''
    Returns dataframe that create an information card for each team.

            Parameters:
                    df     : dataframe
                    team   : string
                    number : integer

            Returns:
                    df_temp: binary number
    '''
    # home - away - general team analysis
    if (number == 1):
        df = df[(df['HomeTeam'] == team)]
    elif (number == 2):
        df = df[(df['AwayTeam'] == team)]
    else:
        df = df[(df['HomeTeam'] == team) | (df['AwayTeam'] == team)]

    #number of matches for team
    num_matches = df.shape[0]

################################################################################
####
    #bet options
    df['home_team_below_1_5'] = df.apply(lambda row :
home_team_below_1_5(row,team), axis=1)
    df['home_team_above_1_5'] = df.apply(lambda row :
home_team_above_1_5(row,team), axis=1)
    df['away_team_below_1_5'] = df.apply(lambda row :
away_team_below_1_5(row,team), axis=1)
    df['away_team_above_1_5'] = df.apply(lambda row :
away_team_above_1_5(row,team), axis=1)

################################################################################
####
    #bet options
    df['win_half_full_home'] = df.apply(lambda row :
win_half_full_home(row,team), axis=1)
    df['win_half_full_away'] = df.apply(lambda row :
win_half_full_away(row,team), axis=1)

################################################################################
####
    #bet options
    df['1_0'] = df.apply(lambda row : one_zero(row,team), axis=1)
    df['1_2'] = df.apply(lambda row : one_two(row,team), axis=1)
    df['0_2'] = df.apply(lambda row : zero_two(row,team), axis=1)

################################################################################
####
    #bet options
    df['win_half'] = df.apply(lambda row : win_half(row,team), axis=1)
    df['draw_half'] = df.apply(lambda row : draw_half(row,team), axis=1)
    df['lose_half'] = df.apply(lambda row : lose_half(row,team), axis=1)

################################################################################
####
    #bet options
    df['win'] = df.apply(lambda row : win(row,team), axis=1)
    df['draw'] = df.apply(lambda row : draw(row,team), axis=1)
```

```
    df['lose'] = df.apply(lambda row : lose(row,team), axis=1)

##########################################################################
####
    #bet options
    df['home_goal'] = df.apply(lambda row : home_goal(row,team), axis=1)
    df['away_goal'] = df.apply(lambda row : away_goal(row,team), axis=1)
    df['sum_goal'] = df['home_goal'] + df['away_goal']

##########################################################################
#####
    #bet options
    df['home_goal_concede'] = df.apply(lambda row :
home_goal_concede(row,team), axis=1)
    df['away_goal_concede'] = df.apply(lambda row :
away_goal_concede(row,team), axis=1)
    df['sum_concede'] = df['home_goal_concede'] + df['away_goal_concede']

##########################################################################
#####
    #display all team information
    display(df[(df['HomeTeam'] == team) | (df['AwayTeam'] == team)]) #query
    #bet options
    df['home_win_final'] = df.apply(lambda row : home_win_final(row,team),
axis=1)
    df['home_lose_final'] = df.apply(lambda row :
home_lose_final(row,team), axis=1)
    df['home_draw_final'] = df.apply(lambda row :
home_draw_final(row,team), axis=1)
    df['away_draw_final'] = df.apply(lambda row :
away_draw_final(row,team), axis=1)
    df['away_win_final'] = df.apply(lambda row : away_win_final(row,team),
axis=1)
    df['away_lose_final'] = df.apply(lambda row :
away_lose_final(row,team), axis=1)
    #bet options
    df['home_win_half'] = df.apply(lambda row : home_win_final(row,team),
axis=1)
    df['home_lose_half'] = df.apply(lambda row : home_lose_final(row,team),
axis=1)
    df['home_draw_half'] = df.apply(lambda row : home_draw_final(row,team),
axis=1)
    df['away_draw_half'] = df.apply(lambda row : away_draw_final(row,team),
axis=1)
    df['away_win_half'] = df.apply(lambda row : away_win_final(row,team),
axis=1)
    df['away_lose_half'] = df.apply(lambda row : away_lose_final(row,team),
axis=1)
    #bet options
    df = df.iloc[:,33:].reset_index(drop=True)
    #display(df)
    #creating information card
    df = df.sum(axis=0).to_frame().T
    df.insert(0, 'Team', team)
    df.insert(1, 'Oynadigi_Mac', num_matches)

    home = df['home_goal']
    df.drop(labels=['home_goal'], axis=1,inplace = True)
    df.insert(2, 'home_goal', home)

    away = df['away_goal']
    df.drop(labels=['away_goal'], axis=1,inplace = True)
```

```
    df.insert(3, 'away_goal', away)

    home_concede = df['home_goal_concede']
    df.drop(labels=['home_goal_concede'], axis=1,inplace = True)
    df.insert(4, 'home_goal_concede', home_concede)

    away_concede = df['away_goal_concede']
    df.drop(labels=['away_goal_concede'], axis=1,inplace = True)
    df.insert(5, 'away_goal_concede', away_concede)

    sum = df['sum_goal']
    df.drop(labels=['sum_goal'], axis=1,inplace = True)
    df.insert(6, 'attigi_goal_ortalama', sum)

    sum_concede = df['sum_concede']
    df.drop(labels=['sum_concede'], axis=1,inplace = True)
    df.insert(7, 'concede_goal_ortalama', sum_concede)

    cols_to_keep = df.columns[6:].values.tolist()
    appended_data = []
    # proportion
    for i in df.columns[6:8]:
        appended_data.append((df[i].loc[0]/num_matches))
    #give percentage
    for i in df.columns[8:]:
        appended_data.append((df[i].loc[0]/num_matches)*100)

    df = pd.DataFrame(appended_data).T
    temp = df.shape[1]

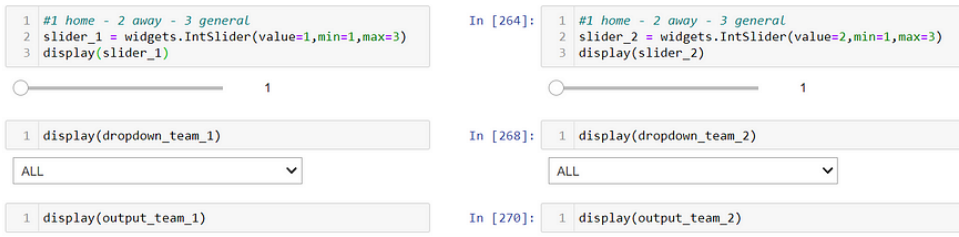
    df.columns = cols_to_keep
    df.insert(0, 'team', team)
    df.insert(1, 'match_played', num_matches)
    df.insert(2, 'home_goal', home)
    df.insert(3, 'away_goal', away)
    df.insert(4, 'sum_goal', sum)
    df.insert(5, 'concede_goal_home', home_concede)
    df.insert(6, 'concede_goal_away', away_concede)
    df.insert(7, 'concede_sum', sum_concede)

    df_temp = df.iloc[:,10:temp].T
    df_temp = df_temp.sort_values(by=0,ascending = False).T

    df_temp.insert(0, 'team', team)
    df_temp.insert(1, 'match_played', num_matches)
    df_temp.insert(2, 'home_goal', home)
    df_temp.insert(3, 'away_goal', away)
    df_temp.insert(4, 'sum_goal', sum)
    df_temp.insert(5, 'concede_goal_home', home_concede)
    df_temp.insert(6, 'concede_goal_away', away_concede)
    df_temp.insert(7, 'concede_sum', sum_concede)

    df_temp = df_temp.T
    df_temp = df_temp.rename(columns={0: 'Information_Card'})

    return df_temp
```



عند تشغيل الكود، سترى ipywidgetsفي نهاية الكود؛

- الخيار الأول هو التحليل 1 (المضيف home) 2 (الخارج away) 3 (جميع المباريات all matches).
- اختر تحليل الفريق الأول.
- اختر تحليل الفريق الثاني.
- وأخيرًا، سترى نتائج الفريقين لتحليلهما وتحديد الخيار الأفضل للمراهنة. يوضح ملف gif أدناه كيفية عمل النظام.

ستشاهد لوحة معلومات العينة على هذا النحو. (ملاحظة: يمكنك استخدام ملحق تقسيم الخلايا لتقسيم دفتر الملاحظات رأسياً لمنع وجود مساحة بيضاء إضافية على الجانب الأيمن من دفاتر Jupyter الخاصة بك والتي تساعد nbextensions).

في هذه المقالة، تناولنا كيفية تنفيذ تحليلات الرهان المستندة إلى البياناتفي بايثون. كل التعليمات البرمجية موجودةفي حسابي على Github. شكرًا لك على القراءة. يرجى إخباري إذا كان لديك أي تعليقات.

بالنسبة لأولئك المهتمين، تابع قراءة الجزء الثانيفي هذه السلسلة باتباع الرابط أعلاه حيث أتعمقفي تحليل البيانات الاستكشافي باستخدام تصورات البيانات لمعرفة المتغيرات التي تحلل بيانات كرة القدم.

في الجزء الأول، قمنا بمعالجة مسبقة لمجموعة بيانات كرة القدم.في هذا الجزء، نقوم بإجراء تحليل استكشافي للبيانات. تحتوي مجموعة البيانات على 79 متغيرًا توضيحيًا تتضمن مجموعة كبيرة من سمات الرهان. يوجد ملف GIF الخاص بمجموعة البيانات أدناه.

	Date	Time	HomeTeam	AwayTeam	FTHG	FTAG	FTR	HTHG	HTAG	HTR	HS	AS	HST	AST	HF	AF	HC	AC	HY	AY	HR	AR
0	2019-08-09	20:00	Liverpool	Norwich	4	1	1	4	0	1	15	12	7	5	9	9	11	2	0	2	0	0
1	2019-08-10	12:30	West Ham	Man City	0	5	2	0	1	2	5	14	3	9	6	13	1	1	2	2	0	0
2	2019-08-10	15:00	Bournemouth	Sheffield United	1	1	0	0	0	0	13	8	3	3	10	19	3	4	2	1	0	0
3	2019-08-10	15:00	Burnley	Southampton	3	0	1	0	0	0	10	11	4	3	6	12	2	7	0	0	0	0
4	2019-08-10	15:00	Crystal Palace	Everton	0	0	0	0	0	0	6	10	2	3	16	14	6	2	2	1	0	1

سنقوم بتحليل فريق ليفربول الذي كان بطل الدوري الإنجليزي الممتاز 2019ـ2020. سأقوم ببعض التجميعات على مجموعة البيانات لتقديمهافي شكل موجز. فيما يلي بعض منها؛

- التجميع الزمني Time aggregation / التجميع حسب الفترة الزمنية Time Slot aggregation.
- التجميع اليومي Day aggregation.
- التجميع الشهري Month aggregation.
- التجميع حسب يوم الأسبوع Day of Week/ تجميع نهاية الأسبوع .

جزء الرسم

```
def show_chart_2_2d(df, x, y, is_bar_chart,
is_sorted,plot_title,x_axis,y_axis):
    '''
    Returns the  plots optional bar or line chart with any given dimension.

    Parameters:
        df (dataframe) : all dataframe
        x : x axis of data set
        y : y axis of data set
        is_bar_chart : boolean
        is_sorted : boolean
        plot_title : string
        x_axis : string
        y axis : string

    Returns:
        return: return multi dimensional line chart or bar chart on plotly
    '''
    chart = go.Bar if is_bar_chart else go.Scatter
    marker = {'size': 15, 'opacity': 0.5, 'line': {'width': 0.5, 'color':
'white'}}
    fig = go.Figure()
    df = df.sort_values(by=x, ascending=True)
    for _y in y:
        if not is_bar_chart:
            fig.add_trace(chart(x=df[x], y=df[_y], mode= 'lines+markers',
name=_y, marker=marker))
```

```
        else:
            fig.add_trace(chart(x=df[x], y=df[_y], name= _y))

    fig.update_layout(
        title=plot_title,
        xaxis_title=x_axis,
        yaxis_title=y_axis)

    fig.show()
#sample function call
#show_chart_2_2d(df_1, 'Time', ['2.5_above_final', '2.5_under_final'],
False, True,'Time Analysis','Time','Count')
```

أستخدم Plotly مع بايثون. يوضح الكود أعلاه كيفية إنشاء المخططات الخطية line plots. سأقارن استراتيجية 2.5 هدف أقل/أعلى لكل التجميعات.

```
def time_agg(df):
    '''
    Returns the aggregate time data

    Parameters:
        df (dataframe) : all frame

    Returns:
        df (dataframe): return the dataframe with agg data
    '''
    df = df.pivot_table(index='Time',

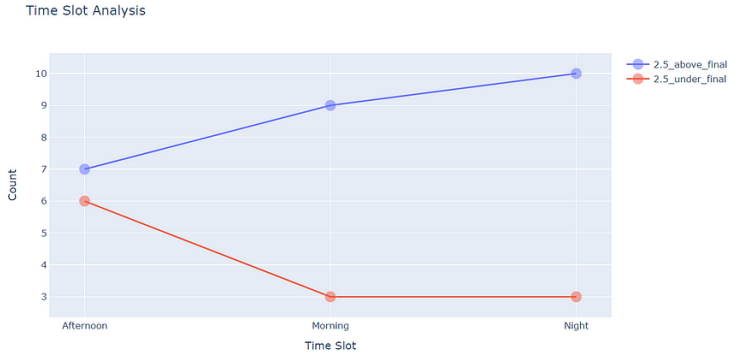
aggfunc={'2.5_above_final':'sum','2.5_under_final':'sum'}
                        ).reset_index()
    return df
```

تجميع الوقت/الفواصل الزمنية

التجميع الأول هو ميزة "الوقت/الفواصل الزمنية Time/Time slot". من الواضح أن الأداء خلال الصباح أو الليل يؤثر على مباراة كرة القدم. يوضح الرسم البياني الأول أي وقت انطلاق 15:00 أو 20:15 من المرجح أن يكون أعلى من النتيجة النهائية بمقدار 2.5. أقوم بتصنيف الأوقات التي تكون في الصباح وبعد الظهر والليل. الرسم البياني الثاني يعطي المزيد من الحدس حول كيفية تأثير وقت انطلاق المباراة على هدف المباراة.

	Time	2.5_above_final	2.5_under_final
0	12:30	2	2
1	15:00	7	1
2	16:00	1	0
3	16:30	3	4
4	17:30	3	2
5	19:00	0	1
6	19:45	0	1
7	20:00	4	1
8	20:15	6	0

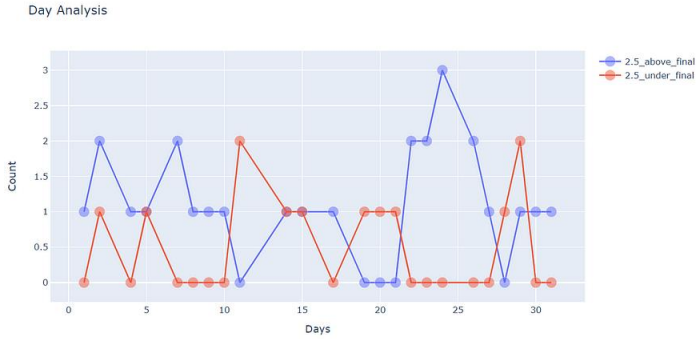
مخطط الوقت



مخطط الفترة الزمنية

التجميع اليومي

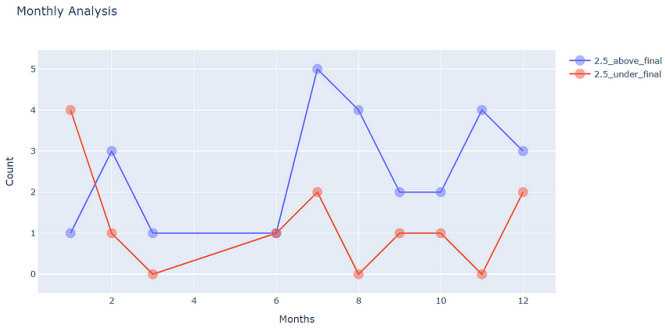
التجميع الثاني هو سمة "اليوم day". من الواضح من الرسم البياني أن نهاية الشهر من المرجح أن تكون أعلى من الأهداف بمقدار 2.5. بشكل عام، الأيام الأخرى متماثلة نوعًا ما.



مخطط الايام

التجميع الشهري

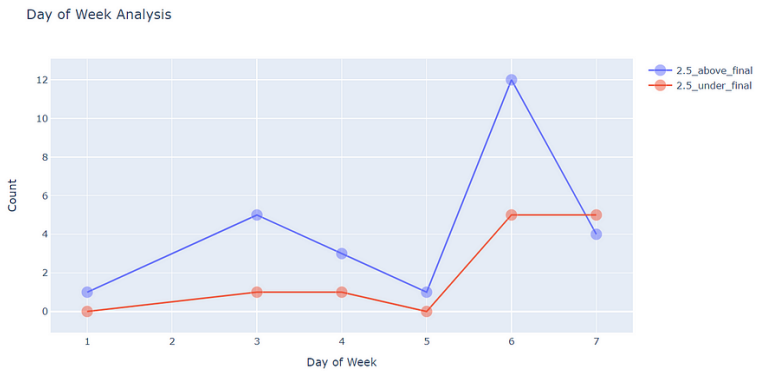
التجميع الثالث هو عمود "الشهر month". يوضح الشكل رسمًا بيانيًا خطيًا يشير إلى أن الفترة الشتوية من المرجح أن تكون 2.5 أقل من النهائي. وعلى النقيض من الأرقام الخاصة بأنواع الرهان الأخرى، فإن مباريات الفترة الصيفية أكثر أهدافًا.



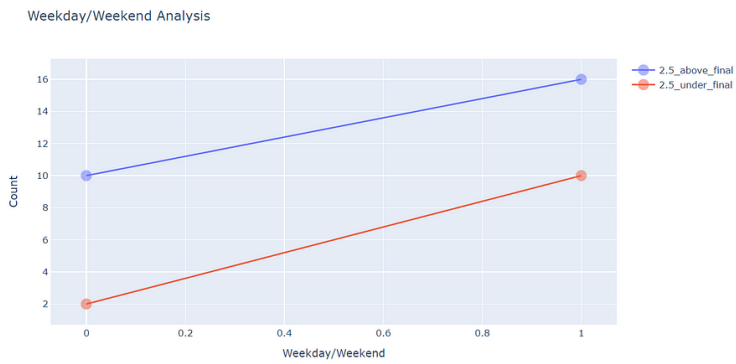
مخطط الشهر

تجميع يوم الأسبوع/نهاية الأسبوع نهاية الاسبوع

التجميع النهائي هو "يوم الأسبوع / نهاية الأسبوع". يشير الرقم الخاص بأول يوم من الأسبوع إلى أن احتمالية تجاوز الأهداف بـ 2.5 هدفًافي يومي الأربعاء والسبت أكبر. وفي أيام أخرى، يكون النمط مشابهًا تقريبًا.



خطط يوم من الاسبوع



مخطط نهاية الاسبوع نهاية الاسبوع

في النهاية، أجرينا بعض تحليلات البيانات الاستكشافية على مجموعة بيانات كرة القدم لاكتشاف الأنماط والاتجاهاتفي ميزات الرهان المحددة. كل التعليمات البرمجيةفي حسابي على Github.

.

المصدر:

https://medium.com/analytics-vidhya/sports-analytics-in-python-part-1-12e4907da227

https://medium.com/analytics-vidhya/exploratory-data-analysis-in-sports-analytics-part-2-5ba6aa50cd5

2) تحليل الألعاب الأولمبية باستخدام بايثون Olympic Games Analysis with Python

اليوم سوف نستكشف مجموعة بيانات عن الألعاب الأولمبية الحديثة، بمافي ذلك جميع الألعاب من أثينا 1896 إلى ريو 2016.

تم جمع البيانات من http://www.sports-reference.com/ في مايو 2018.

تنزيل مجموعات البيانات

- athlete_events
- datasets_31029_40943_noc_regions

استيراد المكتبات

لنبدأ استيراد المكتبات

```
import numpy as np
import pandas as pd
import seaborn as sns
from matplotlib import pyplot as plt
```

جمع المعلومات حول كل من مجموعتي البيانات

سنقوم بما يلي:

1. مراجعة الأسطر الأولى من البيانات؛
2. استخدام دالتي الوصف describe والمعلومات info لجمع المعلومات الإحصائية وأنواع البيانات وأسماء الأعمدة والمعلومات الأخرى.

```
data = pd.read_csv('athlete_events.csv')
data.head()
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	M
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	

```
data.describe()
```

	ID	Age	Height	Weight	Year
count	271116.000000	261642.000000	210945.000000	208241.000000	271116.000000
mean	68248.954396	25.556898	175.338970	70.702393	1978.378480
std	39022.286345	6.393561	10.518462	14.348020	29.877632
min	1.000000	10.000000	127.000000	25.000000	1896.000000
25%	34643.000000	21.000000	168.000000	60.000000	1960.000000
50%	68205.000000	24.000000	175.000000	70.000000	1988.000000
75%	102097.250000	28.000000	183.000000	79.000000	2002.000000
max	135571.000000	97.000000	226.000000	214.000000	2016.000000

```
data.info()
```

```
#Output
>class 'pandas.core.frame.DataFrame<'
RangeIndex: 271116 entries, 0 to 271115
Data columns (total 15 columns):
   # Column  Non-Null Count   Dtype
  -----   --------------  ------  ---
   0 ID      271116 non-null  int64
   1 Name    271116 non-null  object
   2 Sex     271116 non-null  object
   3 Age     261642 non-null  float64
   4 Height  210945 non-null  float64
   5 Weight  208241 non-null  float64
   6 Team    271116 non-null  object
   7 NOC     271116 non-null  object
   8 Games   271116 non-null  object
   9 Year    271116 non-null  int64
  10 Season  271116 non-null  object
  11 City    271116 non-null  object
  12 Sport   271116 non-null  object
  13 Event   271116 non-null  object
  14 Medal   39783 non-null   object
dtypes: float64(3), int64(2), object(10)
memory usage: 31.0+ MB
```

```
regions = pd.read_csv('datasets_31029_40943_noc_regions.csv')
regions.head()
```

	NOC	region	notes
0	AFG	Afghanistan	NaN
1	AHO	Curacao	Netherlands Antilles
2	ALB	Albania	NaN
3	ALG	Algeria	NaN
4	AND	Andorra	NaN

ربط إطارات البيانات

```
merged = pd.merge(data, regions, on='NOC', how='left')
merged.head()
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
4	5	Christine Jacoba Aaftink	F	21.0	185.0	82.0	Netherlands	NED	1988 Winter	1988	Winter	Calgary	Speed Skating	Speed Skating Women's 500 metres	NaN

توزيع أعمار الحائزين على الميداليات الذهبية

لنبدأ بإنشاء إطار بيانات جديد يتضمن فقط الحائزين على الميداليات الذهبية gold medalists.

```
goldMedals = merged[(merged.Medal == 'Gold')]
goldMedals.head()
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
42	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Team All-Around	Gold
44	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Horse Vault	Gold
48	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Pommelled Horse	Gold
60	20	Kjetil Andr Aamodt	M	20.0	176.0	85.0	Norway	NOR	1992 Winter	1992	Winter	Albertville	Alpine Skiing	Alpine Skiing Men's Super G	Gold

أرغب في الحصول على رسم بياني للعمر لرؤية التوزيع ولكنني بحاجة إلى التحقق أولاً مما إذا كان عمود العمر يحتوي على قيم NaN.

```
goldMedals.isnull().any()
```

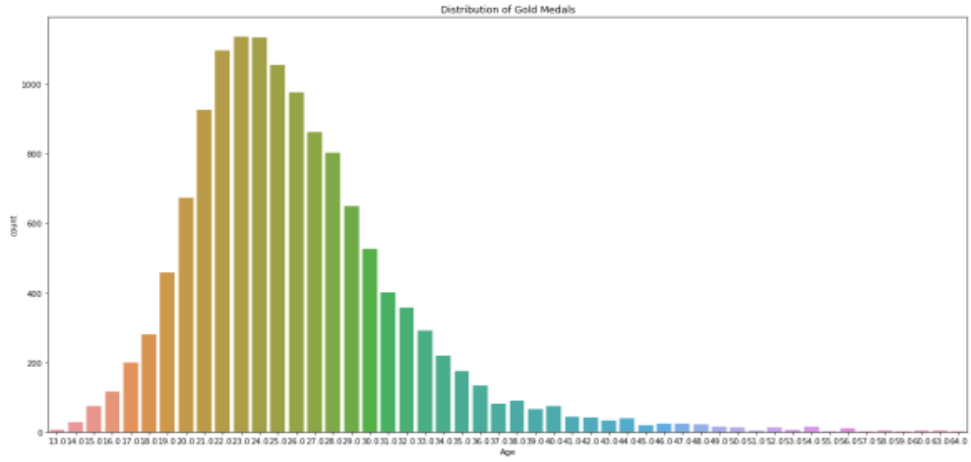
```
#Output
ID        False
Name      False
Sex       False
Age        True
Height     True
Weight     True
Team      False
NOC       False
Games     False
Year      False
Season    False
City      False
Sport     False
Event     False
Medal     False
region     True
notes      True
dtype: bool
```

دعونا نأخذ فقط القيم التي تختلف عن NaN.

```
goldMedals = goldMedals[np.isfinite(goldMedals['Age'])]
```

يمكننا الآن إنشاء مخطط عددي countplot لرؤية نتيجة عملنا:

```
plt.figure(figsize=(20, 10))
plt.tight_layout()
sns.countplot(goldMedals['Age'])
plt.title('Distribution of Gold Medals')
```



يبدو أن لدينا أشخاصًا تبلغ أعمارهم أكثر من 50 عامًا ويحملون ميدالية ذهبية: دعونا نعرف المزيد عن هؤلاء الأشخاص!

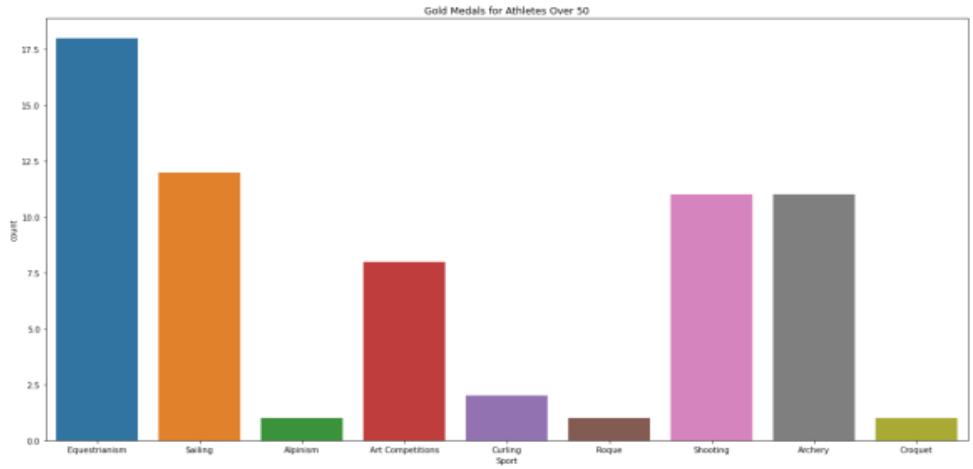
```
goldMedals['ID'][goldMedals['Age'] > 50].count()
```

```
#Output
65
```

65 شخصًا: رائع! ولكن ما هي التخصصات disciplines التي تسمح لك بالحصول على ميدالية ذهبية بعد بلوغك الخمسين من عمرك؟

سنقوم الآن بإنشاء إطار بيانات جديد يسمى masterDisciplines حيث سنقوم بإدراج هذه المجموعة الجديدة من الأشخاص ثم إنشاء تصور باستخدامها.

```
masterDisciplines = goldMedals['Sport'][goldMedals['Age'] > 50]
plt.figure(figsize=(20, 10))
plt.tight_layout()
sns.countplot(masterDisciplines)
plt.title('Gold Medals for Athletes Over 50')
```



يبدو أن الفائزين بالميداليات الذهبية من كبار الرياضيين هم من الرماة shooters والرماة بالقوس والسهم archers والبحارة sailors ، وقبل كل شيء من راكبي الخيل horse riders!

الأمر منطقي: لا أستطيع أن أتخيل عداءة تقطع مسافة 100 مترفي 10 ثوانٍ وهيفي سن 55 عامًا، ولكن من يدري!

النساء في ألعاب القوى

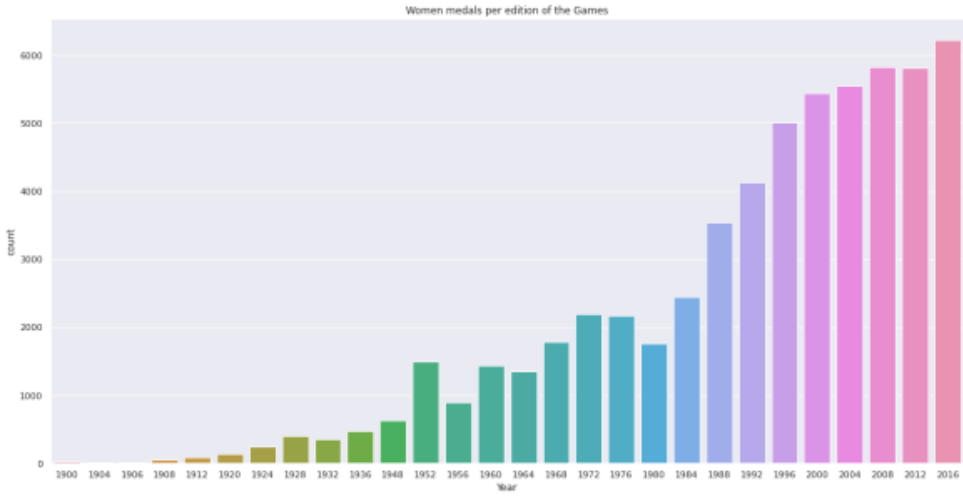
من خلال دراسة البيانات، يمكننا محاولة فهم عدد الميداليات التي حصلت عليها النساء فقطفي تاريخ الألعاب الصيفية الحديث. دعنا ننشئ مجموعة بيانات مفلترة:

```
womenInOlympics = merged[(merged.Sex == 'F') & (merged.Season == 'Summer')]
womenInOlympics.head(10)
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
26	8	Cornelia "Cor" Aalten (-Strannood)	F	18.0	168.0	NaN	Netherlands	NED	1932 Summer	1932	Summer	Los Angeles	Athletics	Athletics Women's 100 metres	NaN
27	8	Cornelia "Cor" Aalten (-Strannood)	F	18.0	168.0	NaN	Netherlands	NED	1932 Summer	1932	Summer	Los Angeles	Athletics	Athletics Women's 4 x 100 metres Relay	NaN
32	13	Minna Maarit Aalto	F	30.0	159.0	55.5	Finland	FIN	1996 Summer	1996	Summer	Atlanta	Sailing	Sailing Women's Windsurfer	NaN
33	13	Minna Maarit Aalto	F	34.0	159.0	55.5	Finland	FIN	2000 Summer	2000	Summer	Sydney	Sailing	Sailing Women's Windsurfer	NaN
79	21	Ragnhild Margrethe Aamodt	F	27.0	163.0	NaN	Norway	NOR	2008 Summer	2008	Summer	Beijing	Handball	Handball Women's Handball	Gold

لرسم المنحنى على مدار الوقت، دعنا ننشئ رسمًا بيانيًا نضع فيه السنة (على المحور السيني) وعدد الميداليات لكل نسخة من الألعاب (مع الأخذفي الاعتبار أنه سيكون لدينا المزيد من الميداليات لنفس الرياضي).

```
sns.set(style="darkgrid")
plt.figure(figsize=(20, 10))
sns.countplot(x='Year', data=womenInOlympics)
plt.title('Women medals per edition of the Games')
```



عادة ما أقوم بمراجعة البيانات:في الأسفل حاولت مراجعة الفائزين بالميداليات فقطفي طبعة صيف عام 1900 لمعرفة ما إذاكان التصور صحيحًا.

```
womenInOlympics.loc[womenInOlympics['Year'] == 1900].head(10)
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal	region	n
283	150	Margaret Ives Abbott (-Dunne)	F	23.0	NaN	NaN	United States	USA	1900 Summer	1900	Summer	Paris	Golf	Golf Women's Individual	Gold	USA	
284	151	Mary Perkins Ives Abbott (Perkins-)	F	42.0	NaN	NaN	United States	USA	1900 Summer	1900	Summer	Paris	Golf	Golf Women's Individual	NaN	USA	
30535	15740	A. Brun	F	NaN	NaN	NaN	France	FRA	1900 Summer	1900	Summer	Paris	Golf	Golf Women's Individual	NaN	France	
44448	22925	Charlotte Reinagle Cooper (-Sterry)	F	29.0	NaN	NaN	Great Britain	GBR	1900 Summer	1900	Summer	Paris	Tennis	Tennis Women's Singles	Gold	UK	
44449	22925	Charlotte Reinagle Cooper (-Sterry)	F	29.0	NaN	NaN	Great Britain	GBR	1900 Summer	1900	Summer	Paris	Tennis	Tennis Mixed Doubles	Gold	UK	

حسنًا، دعنا نحسب الصفوف (نفس الكود المذكور أعلاه مع إضافة دالة count() والتصفية حسب المعرف ID فقط).

```
womenInOlympics['ID'].loc[womenInOlympics['Year'] == 1900].count()
```

```
#Output
33
```

لذا لدينا 33 سجلاً (مع التكرارات، على سبيل المثال، فازت "ماريون جونز Marion Jones (ـ فاركوهار Farquhar)" بميداليةفي كل من فردي السيداتفي التنس وتنس الزوجي المختلط Tennis Women's Singles and Tennis Mixed Doubles ـ للتأكد من ذلك، قمت أيضًا بالتحقق من ذلك باستخدام ويكيبيديا ويبدو أن النتيجة صحيحة).

الميداليات حسب كل دولة

دعونا الآن نستعرض الدول الخمس الأولى الحائزة على الميداليات الذهبية:

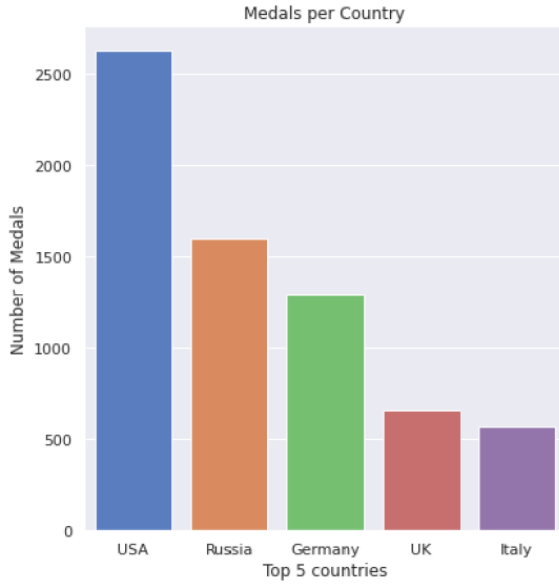
```
goldMedals.region.value_counts().reset_index(name='Medal').head()
```

```
#Output
index      Medal
0   USA     2627
1   Russia  1599
2   Germany 1293
3   UK      657
4   Italy   567
```

دعونا نرسم هذا:

```
totalGoldMedals =
goldMedals.region.value_counts().reset_index(name='Medal').head(5)
g = sns.catplot(x="index", y="Medal", data=totalGoldMedals,
```

```
                height=6, kind="bar", palette="muted)"
g.despine(left=True)
g.set_xlabels("Top 5 countries")
g.set_ylabels("Number of Medals")
plt.title('Medals per Country')
```



يبدو أن الولايات المتحدة هي الدولة الأكثر فوزًا.

ولكن ما هي التخصصات التي حاز فيها الرياضيون الأمريكيون على أكبر عدد من الميداليات الذهبية؟

التخصصات التي حازت على أكبر عدد من الميداليات الذهبية

لنقم بإنشاء إطار بيانات لتصفية الميداليات الذهبية للولايات المتحدة فقط.

```
goldMedalsUSA = goldMedals.loc[goldMedals['NOC'] == 'USA']
```

تم! الآن، يمكننا حساب الميداليات لكل تخصص:

```
goldMedalsUSA.Event.value_counts().reset_index(name='Medal').head(20)
```

دعونا نقوم بتقطيع إطار البيانات باستخدام بيانات الرياضيين الذكور فقط لمراجعته بشكل أفضل:

```
basketballGoldUSA = goldMedalsUSA.loc[(goldMedalsUSA['Sport'] ==
'Basketball') & (goldMedalsUSA['Sex'] == 'M')].sort_values(['Year'])
basketballGoldUSA.head(15)
```

ما افترضناه صحيح: لم يتم تجميع الميداليات حسب الإصدار Edition/الفريق Team ، لكننا كنا نحسب الميداليات الذهبية لكل عضوفي الفريق!

لنبدأفي تجميع الرياضيين حسب السنة ـ الفكرة هي إنشاء إطار بيانات جديد لإنشاء مرشح مسبق pre- filter باستخدام السجل الأول فقط لكل عضوفي الفريق.

```
groupedBasketUSA = basketballGoldUSA.groupby(['Year']).first()
groupedBasketUSA
groupedBasketUSA['ID'].count()
```

ما هو متوسط طول/ وزن الحائز على ميدالية أوليمبية؟

لنحاول رسم مخطط تشتت scatterplot للطول مقابل الوزن لنرى توزيع القيم (بدون التجميع حسب التخصص).

أولاً وقبل كل شيء، علينا أن نأخذ مرة أخرى إطار البيانات goldMedals.

```
goldMedals.head()
```

	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
42	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Team All-Around	Gold
44	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Horse Vault	Gold
48	17	Paavo Johannes Aaltonen	M	28.0	175.0	64.0	Finland	FIN	1948 Summer	1948	Summer	London	Gymnastics	Gymnastics Men's Pommelled Horse	Gold
60	20	Kjetil Andr Aamodt	M	20.0	176.0	85.0	Norway	NOR	1992 Winter	1992	Winter	Albertville	Alpine Skiing	Alpine Skiing Men's	Gold

يمكننا أن نرى أن لدينا قيم NaN في عمودي الارتفاع والوزن.

في هذه المرحلة، يمكننا التصرف على النحو التالي:

1. استخدام الصفوف التي تحتوي على قيمة في عمودي الارتفاع والوزن فقط؛
2. استبدال القيمة بمتوسط العمود.

في رأيي، الحل 2 ليس هو الحل الأفضل: نحن نتحدث عن بيانات الرياضيين من مختلف الأعمار والتخصصات المختلفة (الذين خضعوا لتدريبات مختلفة).

لننتقل إلى الحل 1.

أول شيء يجب القيام به هو جمع معلومات عامة حول إطار البيانات الذي يتعين علينا استخدامه: goldMedals.

```
goldMedals.info()
```

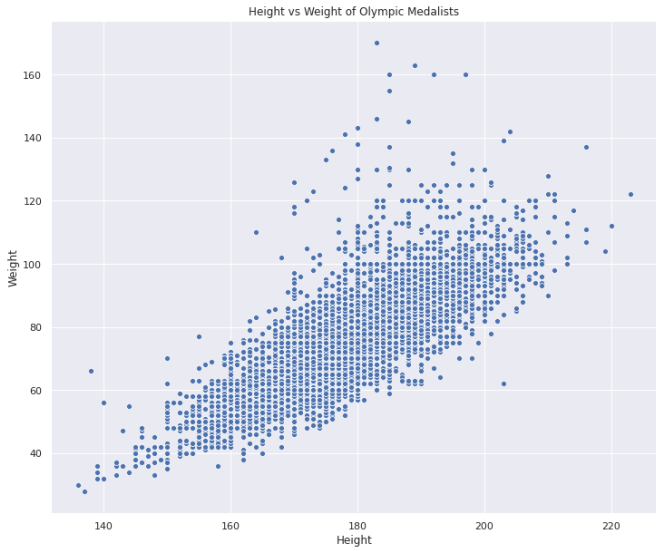
```
#Output
<class 'pandas.core.frame.DataFrame'>
Int64Index: 13224 entries, 3 to 271076
Data columns (total 17 columns):
 #   Column  Non-Null Count  Dtype
---  ------  --------------  -----
 0   ID      13224 non-null  int64
 1   Name    13224 non-null  object
 2   Sex     13224 non-null  object
 3   Age     13224 non-null  float64
 4   Height  10532 non-null  float64
 5   Weight  10248 non-null  float64
 6   Team    13224 non-null  object
 7   NOC     13224 non-null  object
 8   Games   13224 non-null  object
 9   Year    13224 non-null  int64
 10  Season  13224 non-null  object
 11  City    13224 non-null  object
 12  Sport   13224 non-null  object
 13  Event   13224 non-null  object
 14  Medal   13224 non-null  object
 15  region  13223 non-null  object
 16  notes   171 non-null    object
dtypes: float64(3), int64(2), object(12)
memory usage: 2.4+ MB
```

حسنًا، لدينا أكثر من 13000 صف.

سنقوم الآن بإنشاء إطار بيانات يقوم بتصفية الصفوف التي تحتوي فقط على العمودين Height وWeight.

```
notNullMedals = goldMedals[(goldMedals['Height'].notnull()) &
(goldMedals['Weight'].notnull())]
```

```
plt.figure(figsize=(12, 10))
ax = sns.scatterplot(x="Height", y="Weight", data=notNullMedals)
plt.title('Height vs Weight of Olympic Medalists')
```



تظهر الغالبية العظمى من العينات علاقة خطية بين الطول والوزن (كلما زاد الوزن، زاد الطول).

لدينا استثناءات وأنا على استعداد لمعرفة المزيد!

على سبيل المثال، دعنا نرى من هو الرياضي الذي يزيد وزنه عن 160 كيلوجرامًا.

```
notNullMedals.loc[notNullMedals['Weight'] > 160]
```

تطور الألعاب الأولمبية عبر الزمن

سنحاول الآن الإجابة على الأسئلة التالية:

- كيف اختلف عدد الرياضيين/الدول مع مرور الوقت؟
- كيف اختلف عدد الرجال/النساء مع مرور الوقت؟
- ماذا عن متوسط العمر والوزن والطول مع مرور الوقت؟

تباين الرياضيين الذكور/ الإناث على مر الزمن (الألعاب الصيفية)

سنقوم الآن بإنشاء إطارين للبيانات لتقسيم مجموعة البيانات الخاصة بنا باستخدام الجنس Sex والموسم Season (نود مراجعة الألعاب الصيفية summer games فقط).

```
MenOverTime = merged[(merged.Sex == 'M') & (merged.Season == 'Summer')]
WomenOverTime = merged[(merged.Sex == 'F') & (merged.Season == 'Summer')]
```

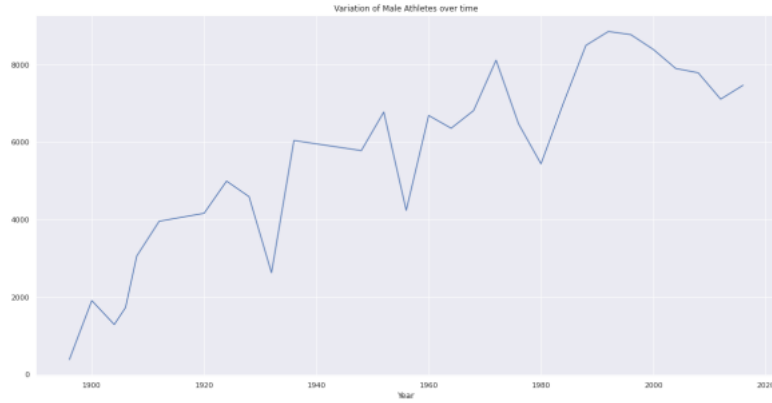
تم، دعنا نتحقق من رأس أحد إطارات البيانات الجديدة لرؤية النتيجة:

```
MenOverTime.head()
```

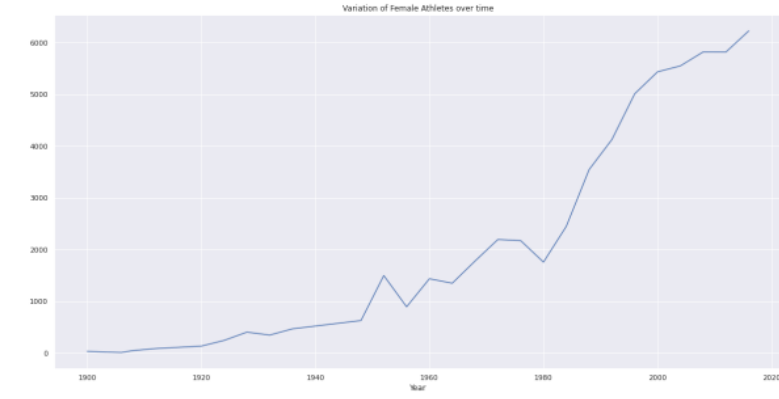
	ID	Name	Sex	Age	Height	Weight	Team	NOC	Games	Year	Season	City	Sport	Event	Medal
0	1	A Dijiang	M	24.0	180.0	80.0	China	CHN	1992 Summer	1992	Summer	Barcelona	Basketball	Basketball Men's Basketball	NaN
1	2	A Lamusi	M	23.0	170.0	60.0	China	CHN	2012 Summer	2012	Summer	London	Judo	Judo Men's Extra-Lightweight	NaN
2	3	Gunnar Nielsen Aaby	M	24.0	NaN	NaN	Denmark	DEN	1920 Summer	1920	Summer	Antwerpen	Football	Football Men's Football	NaN
3	4	Edgar Lindenau Aabye	M	34.0	NaN	NaN	Denmark/Sweden	DEN	1900 Summer	1900	Summer	Paris	Tug-Of-War	Tug-Of-War Men's Tug-Of-War	Gold
29	10	Einar Ferdinand "Einari" Aalto	M	26.0	NaN	NaN	Finland	FIN	1952 Summer	1952	Summer	Helsinki	Swimming	Swimming Men's 400 metres Freestyle	NaN

حسنًا،في هذا الوقت نحن مستعدون لإنشاء المخططات. المخطط الأول للرجال، والثاني للنساء:

```
part = MenOverTime.groupby('Year')['Sex'].value_counts()
plt.figure(figsize=(20, 10))
part.loc[:,'M'].plot()
plt.title('Variation of Male Athletes over time')
```




```
part = WomenOverTime.groupby('Year')['Sex'].value_counts()
plt.figure(figsize=(20, 10))
part.loc[:,'F'].plot()
plt.title('Variation of Female Athletes over time')
```



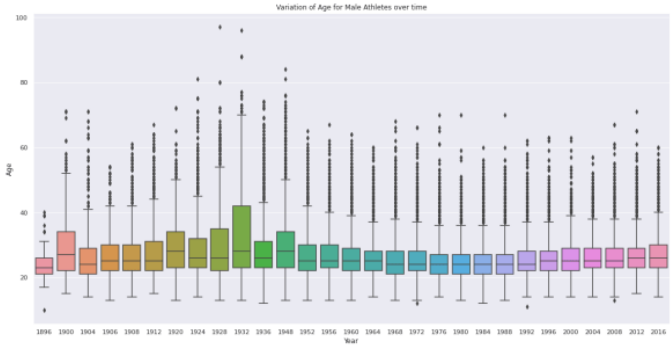
ما رأيته على الفور هو أنه بالنسبة للنساء:

1. لدينا زيادة حادةفي عدد السكان؛
2. النمو مستمر.

من ناحية أخرى، يبدو أن النمو بالنسبة للرجال أقل قوة:

1. بعد عام 1990، يمكننا أن نرى انخفاضًا كبيرًافي عدد الرياضيين الذكورفي الألعاب الصيفية؛
2. لقد بدأ النمو ببطء مؤخرًا.

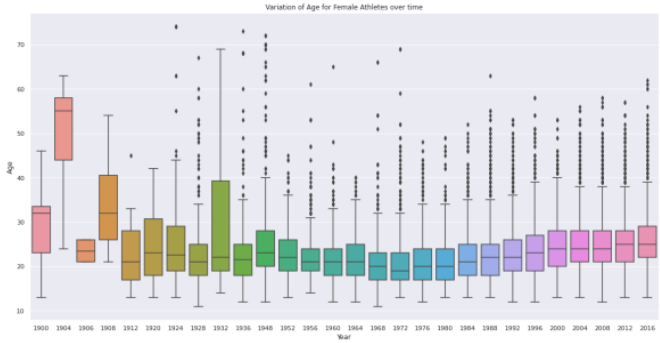
```
plt.figure(figsize=(20, 10))
sns.boxplot('Year', 'Age', data=MenOverTime)
plt.title('Variation of Age for Male Athletes over time')
```



ما هو غريب بالنسبة لي هو أعمار بعض الرياضيينفي الألعاب بين عامي 1924 و1948: دعونا نلقي نظرة على جميع الأشخاص الذين تزيد أعمارهم عن 80 عامًا.

```
MenOverTime.loc[MenOverTime['Age'] > 80].head(10)
```

```
plt.figure(figsize=(20, 10))
sns.boxplot('Year', 'Age', data=WomenOverTime)
plt.title('Variation of Age for Female Athletes over time')
```



نقاط مثيرة للاهتمام بالنسبة لي:

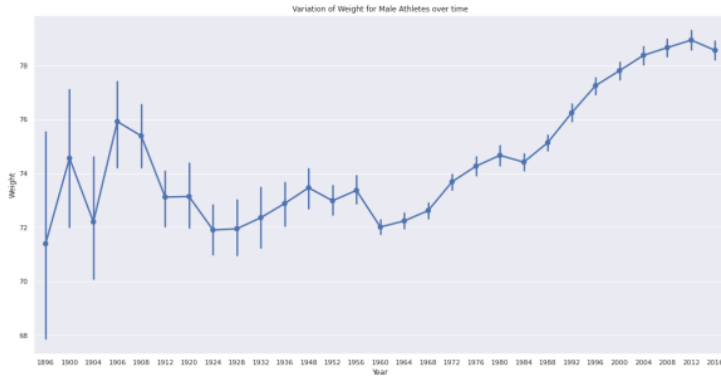
- بشكل عام، يبدأ توزيع الأعمار بحد أدنى أقل وحد أقصى أقل؛
- في عام 1904 كان توزيع الأعمار مختلفًا تمامًا عن الألعاب الأولمبية الأخرى: دعنا نتعرف على المزيد حول هذه النقطة:

```
WomenOverTime.loc[WomenOverTime['Year'] == 1904]
```

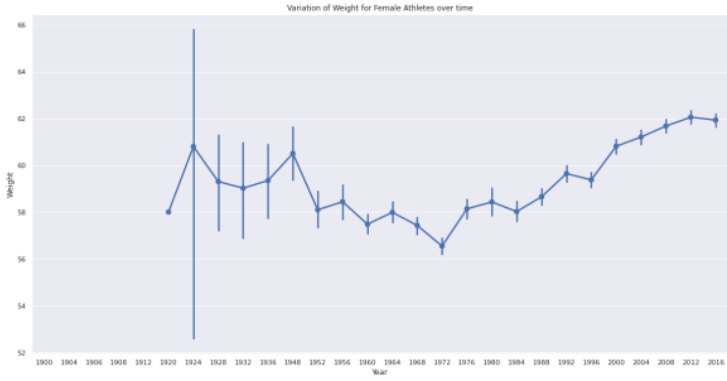
سنحاول الآن استخدام المخطط النقطي pointplot لتوضيح التباينفي الوزن بين الرياضيين.

سيعرض الرسم البياني الأول بيانات الرجال، والثاني بيانات النساء:

```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Weight', data=MenOverTime)
plt.title('Variation of Weight for Male Athletes over time')
```




```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Weight', data=WomenOverTime)
plt.title('Variation of Weight for Female Athletes over time')
```



ما يمكننا رؤيته هو أنه يبدو أنه ليس لدينا بيانات عن النساء قبل عام 1924. دعونا نحاول تصفية جميع الرياضياتفي تلك الفترة لمراجعة هذه النقطة:

```
womenInOlympics.loc[womenInOlympics['Year'] < 1924].head(20)
```

باستخدام نفس المخطط النقطي (باستخدام لوحة مختلفة)، يمكننا رسم تغير الوزن على مدار الوقت.

سيوضح الرسم البياني الأول المعلومات الخاصة بالرجال، والثاني للنساء:

```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Height', data=MenOverTime, palette='Set2')
plt.title('Variation of Height for Male Athletes over time')
```


```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Height', data=WomenOverTime, palette='Set2')
plt.title('Variation of Height for Female Athletes over time')
```

ما قد نراه:

- بالنسبة للرجال والنساء، يتزايد الطول بمرور الوقت ولكنه يتناقص بين عامي 2012 و2016.
- بالنسبة للنساء، لدينا ذروة بين عامي 1928 و1948، دعونا نتعمقفي هذه النقطة:

```
WomenOverTime.loc[(WomenOverTime['Year'] > 1924) & (WomenOverTime['Year'] <
1952)].head(10)
```

دعونا نرى التقدمفي العمر مع مرور الوقت بالنسبة للرياضيين الإيطاليين.

سأبدأ بمراجعة مجموعة البيانات MenOverTime لتحديث الأعمدة:

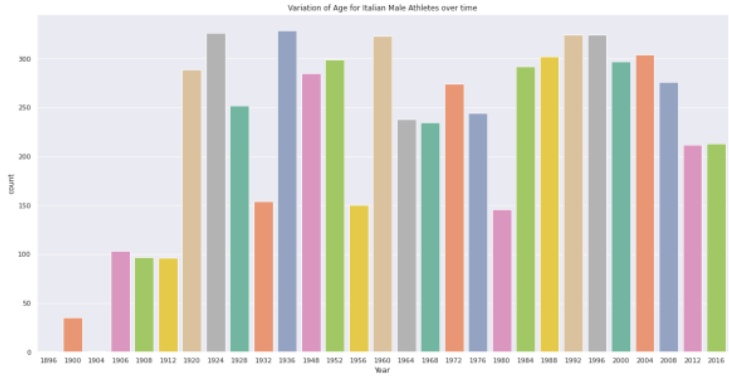
```
MenOverTime.head()
```

دعنا ننشئ إطار بيانات مقسمًا يتضمن الرياضيين الذكور فقط من إيطاليا.

```
itMenOverTime = MenOverTime.loc[MenOverTime['region'] == 'Italy']
```

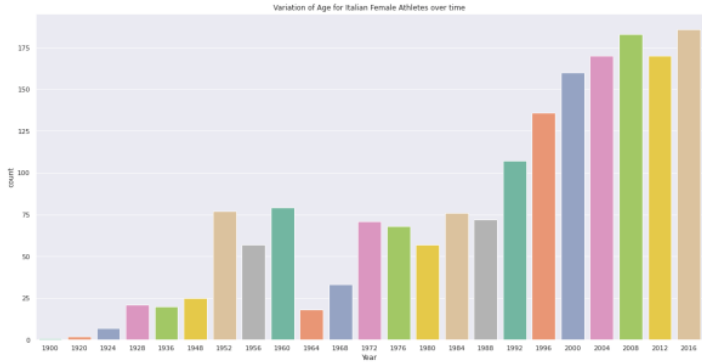
حسنًا، يمكننا الآن رسم التغيير بمرور الوقت:

```
sns.set(style="darkgrid")
plt.figure(figsize=(20, 10))
sns.countplot(x='Year', data=itMenOverTime, palette='Set2')
plt.title('Variation of Age for Italian Male Athletes over time')
```



حسنًا، يمكننا إجراء نفس العملية بسرعة للنساء:

```
itWomenOverTime = WomenOverTime.loc[WomenOverTime['region'] == 'Italy']
sns.set(style="darkgrid")
plt.figure(figsize=(20, 10))
sns.countplot(x='Year', data=itWomenOverTime, palette='Set2')
plt.title('Variation of Age for Italian Female Athletes over time')
```



دعونا أولاً نعزل كل تخصصات إطار بيانات الألعاب الأولمبية.

فكرتي هي معرفة ما إذا كانت الجمباز Gymnastics تسمى بشكل مختلف أو ما إذا كان هناك أي نوع.

```
#Output
']Basketball,'
```

```
' Judo,'
' Football,'
' Tug-Of-War,'
' Swimming,'
' Badminton,'
' Gymnastics,'
' Athletics,'
' Art Competitions,'
' Wrestling,'
' Water Polo,'
' Sailing,'
' Rowing,'
' Fencing,'
' Equestrianism,'
' Shooting,'
' Boxing,'
' Taekwondo,'
' Cycling,'
' Weightlifting,'
' Diving,'
' Canoeing,'
' Handball,'
' Tennis,'
' Modern Pentathlon,'
' Hockey,'
' Volleyball,'
' Baseball,'
' Table Tennis,'
' Archery,'
' Trampolining,'
' Beach Volleyball,'
' Golf,'
' Rugby Sevens,'
' Triathlon,'
' Rugby,'
' Lacrosse,'
' Polo,'
' Cricket,'
' Ice Hockey,'
' Racquets,'
' Motorboating,'
' Croquet,'
' Figure Skating,'
' Jeu De Paume,'
' Roque,'
' Basque Pelota,'
' Alpinism,'
' Aeronautics['
```

الجمباز

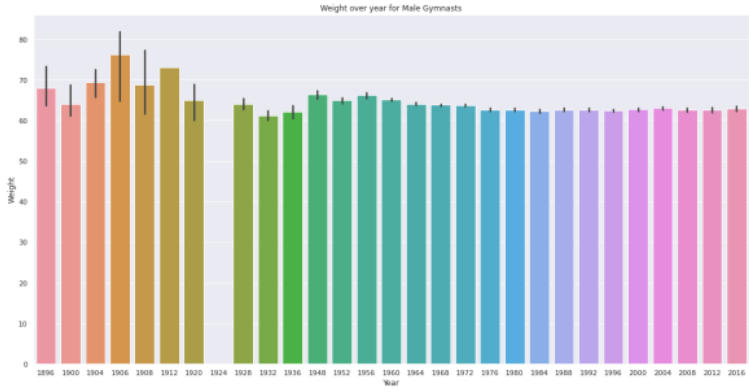
حسنًا، السلسلة التي يجب استخدامها للتصفية هي "الجمباز Gymnastics": دعنا ننشئ إطارين بيانات جديدين للرجال والنساء.

```
gymMenOverTime = MenOverTime.loc[MenOverTime['Sport'] == 'Gymnastics']
gymWomenOverTime = WomenOverTime.loc[WomenOverTime['Sport'] ==
'Gymnastics']
```

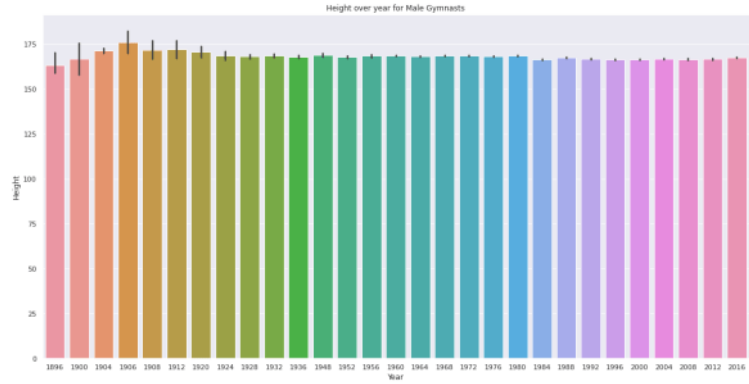
حسنًا: لنبدأ الآنفي إنشاء مخططنا للرياضيين الذكور والإناث ومن ثم يمكننا إجراء ملاحظاتنا.

```
plt.figure(figsize=(20, 10))
```

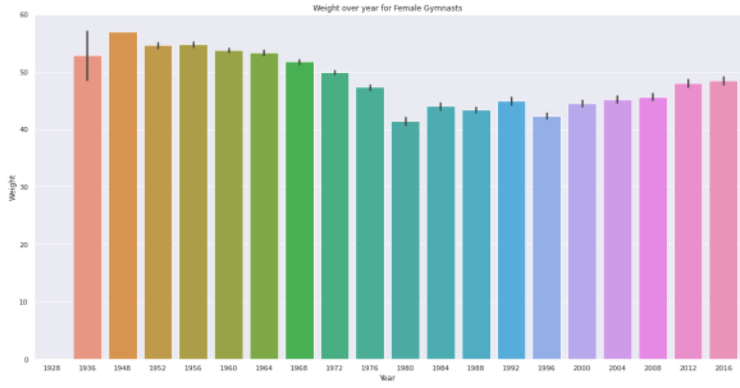
```
sns.barplot('Year', 'Weight', data=gymMenOverTime)
plt.title('Weight over year for Male Gymnasts')
```




```
plt.figure(figsize=(20, 10))
sns.barplot('Year', 'Height', data=gymMenOverTime)
plt.title('Height over year for Male Gymnasts')
```

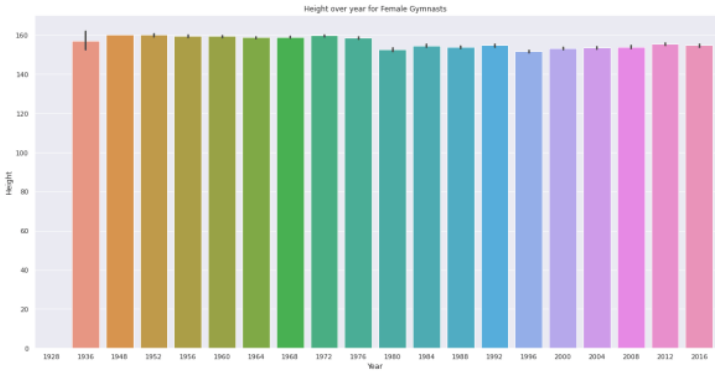



```
plt.figure(figsize=(20, 10))
sns.barplot('Year', 'Weight', data=gymWomenOverTime)
plt.title('Weight over year for Female Gymnasts')
```




```
plt.figure(figsize=(20, 10))
sns.barplot('Year', 'Height', data=gymWomenOverTime)
```

```
plt.title('Height over year for Female Gymnasts')
```



لقد لاحظت بعض الأشياء:

- لقد انخفض وزن لاعبات الجمباز الإناث بمقدار 60 إلى 50 كيلوغرامًافي المتوسط؛
- لقد ظل وزن الرجال مستقرًا إلى حد ما منذ عام 1964؛
- لقد أصبح الطول أكثر استقرارًا لكل من الرجال والنساء.

كما يبدو أن بيانات وزن الرجال من عام 1924 مفقودة: فلنتحقق من ذلك.

رفع الأثقال

دعونا نعمل على تحليل مماثل لما قمنا به للجمباز وكذلك لرافعي الأثقال.

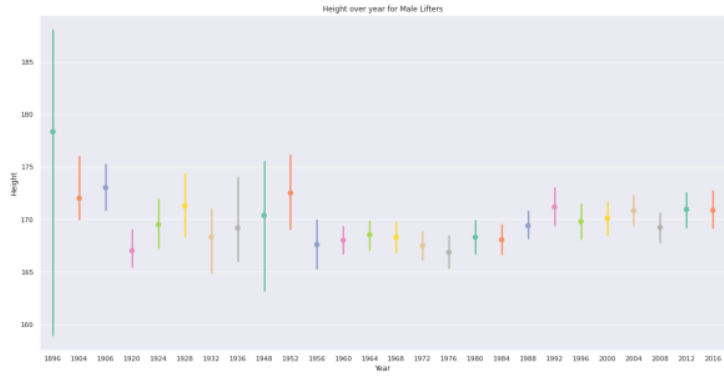
يمكننا البدءفي إنشاء إطار بيانات جديد مخصص.

```
wlMenOverTime = MenOverTime.loc[MenOverTime['Sport'] == 'Weightlifting']
wlWomenOverTime = WomenOverTime.loc[WomenOverTime['Sport'] ==
'Weightlifting']
```

حسنًا: لنبدأ الآنفي إنشاء مخططنا للرياضيين الذكور والإناث ومن ثم يمكننا إجراء ملاحظاتنا.

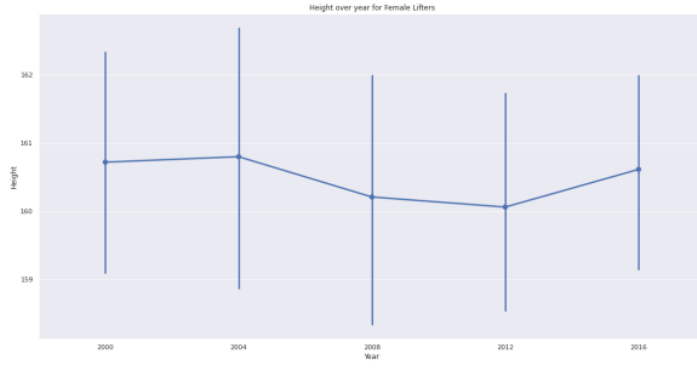
```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Weight', data=wlMenOverTime, palette='Set2')
plt.title('Weight over year for Male Lifters')
```

```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Height', data=wlMenOverTime, palette='Set2')
plt.title('Height over year for Male Lifters')
```




```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Weight', data=wlWomenOverTime)
plt.title('Weight over year for Female Lifters')
```


```
plt.figure(figsize=(20, 10))
sns.pointplot('Year', 'Height', data=wlWomenOverTime)
plt.title('Height over year for Female Lifters')
```



المصدر:

https://thecleverprogrammer.com/2020/05/27/data-science-project-olympic-/games-data-exploration

3) التنبؤ بالإصابات لدى العدائين التنافسيين باستخدام التعلم الآلي Injury Prediction in Competitive Runners Using Machine Learning

الجري Running رياضة شعبيةفي جميع أنحاء العالم، حيث يشارك عدد كبير من الأفرادفي أنشطة مثل الركض أو الجري أو الجري على الطرق الوعرة.في الولايات المتحدة وحدها، شارك حوالي 60 مليون شخصفي هذه الأنشطةفي عام 2017. ومع ذلك، تكشف إحصائية مثيرة للقلق أن حوالي نصف العدائين يعانون من إصابات سنويًا. يمكن أن يكون التعامل مع هذه الإصابات عملية صعبة وتستغرق وقتًا طويلاً، مما يدفع العدائين إلى تبني استراتيجيات مختلفة لتقليل فرص الإصابة. تشمل بعض هذه التدابير الوقائية استخدام بكرات rollers، والحصول على تدليك، وطلب التوجيه من المدربين المحترفين. لسوء الحظ، لا يستطيع الجميع تحمل تكاليف هذه الموارد، مما يجعل الوقاية من الإصابات مصدر قلق كبير للعديد من العدائين.

استجابة لهذا القلق، يتبنى الخبراء وعلماء البيانات بشكل متزايد قدرات التعلم الآلي machine learning للتنبؤ بالإصابات وتجنبها لدى العدائين التنافسيين. من خلال الاستفادة من الخوارزميات المتطورة ومجموعات البيانات الواسعة، تمتلك نماذج التعلم الآلي القدرة على تحويل منهجيات الوقاية من الإصابات، مما يؤديفي النهاية إلى رفع مستوى الرفاهية العامة للرياضيين.

فوائد التنبؤ بالإصابات لدى العدائين التنافسيين باستخدام التعلم الآلي

- يمكن للتعلم الآلي تحليل مجموعات بيانات واسعة النطاق، بمافي ذلك سجلات الإصابات وأنماط التدريب والبيانات الحيوية والعوامل البيئية، للكشف عن الأنماط والارتباطات الخفية، مما يساعدفي اتخاذ قرارات أكثر استنارة من قبل المدربين والرياضيين.
- يمكن لهذه النماذج التعلم باستمرار من البيانات الجديدة، وتحسين دقتها وفعاليتها بمرور الوقت والتكيف مع عوامل خطر الإصابة المتغيرة مع تقدم مسيرة الرياضي.
- يتم توفير تقييمات مخاطر الإصابةفي الوقت الفعلي من خلال التعلم الآلي، مما يسمح بإجراء تعديلات فورية على أحمال التدريب والممارسات، مما يقلل من فرص الإصابات أثناء التدريب المكثف أو المسابقات.

تحديات التنبؤ بالإصابات لدى العدائين التنافسيين باستخدام التعلم الآلي

في حين أن إمكانات التعلم الآليفي التنبؤ بالإصابات واعدة، إلا أن هناك أيضًا تحديات كبيرة يجب معالجتها:

- يعد الحصول على بيانات عالية الجودة ومتنوعة وموثوقة تحديًا كبيرًا حيث تعتمد نماذج التعلم الآلي بشكل كبير على البيانات للتدريب. قد تؤدي البيانات المتحيزة أو غير المكتملة أو غير التمثيلية إلى تنبؤات غير دقيقة.
- إن تعقيد وظائف الجسم البشرية والطبيعة المتعددة العوامل للإصابات تشكل تحدياتفي إنشاء نماذج يمكنها التقاط جميع العوامل المساهمة بشكل شامل. يعد تحديد التفاعلات بين عوامل الخطر المختلفة والنظر فيها أمرًا بالغ الأهمية لفعالية النموذج.
- يعد تفسير مخرجات نماذج التعلم الآلي أمرًا صعبًا.في حين أنها يمكن أن توفر تنبؤات دقيقة، إلا أن فهم الأسباب الكامنة وراء هذه التنبؤات قد يكون صعبًا. تعد النماذج الشفافة والقابلة للتفسير أمرًا حيويًا لكسب الثقة والقبول من المدربين والرياضيين ومحترفي الطب الرياضي.

التنبؤ بالإصابات باستخدام التعلم الآلي باستخدام بايثون

مجموعة البيانات

تتكون مجموعة البيانات dataset من سجل تدريب شامل من فريق هولندي مشهور للجري على مستوى عالٍ يمتد لسبع سنوات (2012–2019). وهي تشمل عدائي المسافات المتوسطة والطويلة الذين يتنافسونفي سباقات تتراوح بين 800 متر والماراثون. وهذا الاختيار مبرر بأنظمة التدريب القائمة على التحمل القابلة للمقارنة. والجدير بالذكر أن مدرب الفريق ظل ثابتًا طوال فترة جمع البيانات.

توجد سجلات من 74 عداءًافي مجموعة البيانات، منهم 27 امرأة و47 رجلاً. وفي المتوسط، كان الرياضيون جزءًا من الفريق لمدة 3.7 عامًا تقريبًا. تنافست غالبية العدائين على المستوى الوطني، بينما شارك البعضفي مسابقات دولية. والتزمت الدراسة بشكل صارم بإرشادات إعلان هلسنكي وحصلت على موافقة من لجنة الأخلاقياتفي مؤسسة المؤلف الثاني.

سنحاول هنا التنبؤ بالإصابة لدى المتسابق والبحث عن أفضل نموذج للتنبؤ بالإصابة.

استيراد المكتبات

```
import numpy as np # linear algebra
import pandas as pd # data processing, .csv file I/O
import sklearn as sk
import seaborn as sns
import matplotlib.pyplot as plt
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn.preprocessing import StandardScaler
from sklearn.svm import SVC
from sklearn.metrics import classification_report, confusion_matrix, accura
cy_score
from sklearn import datasets, metrics
```

قراءة مجموعة البيانات

```
dataframe = pd.read_csv("/kaggle/input/injury-prediction-for-competitive-
runners/week_approach_maskedID_timeseries.csv")
np.random.seed(0)
```

تحليل البيانات الاستكشافي (EDA)

سنبدأ باستكشاف البيانات الأساسية لتحديد ما إذا كانت مجموعة البيانات تحتوي على أي شذوذ anomalies أو تناقضات inconsistencies.

```
missing_Values = dataframe.isnull().sum()
missing_Values
```

```
nr. sessions                0
nr. rest days               0
total kms                   0
max km one day              0
total km Z3-Z4-Z5-T1-T2     0
                           ..
injury                      0
rel total kms week 0_1      0
rel total kms week 0_2      0
rel total kms week 1_2      0
Date                        0
Length: 72, dtype: int64
```

```
dataframe.describe()
```

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3-Z4-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3-4 one day	total km Z5-T1-T2	...
count	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	42798.000000	...
mean	5.809337	1.874667	49.543911	14.009255	9.433621	0.930184	1.672531	4.859398	3.456888	4.063970	...
std	2.484234	1.853287	36.715017	9.071678	8.887120	1.040631	1.263528	6.984670	4.577423	5.645305	...
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
25%	5.000000	1.000000	22.800000	9.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000	...
50%	6.000000	1.000000	44.800000	13.400000	8.000000	1.000000	2.000000	0.000000	0.000000	1.500000	...
75%	7.000000	3.000000	70.100000	18.300000	14.600000	2.000000	3.000000	8.000000	6.300000	6.300000	...
max	14.000000	7.000000	242.000000	131.000000	100.000000	6.000000	7.000000	79.800000	75.000000	80.000000	...

8 rows × 72 columns

```
dataframe.head(10)
```

	nr. sessions	nr. rest days	total kms	max km one day	total km Z3-Z4-Z5-T1-T2	nr. tough sessions (effort in Z5, T1 or T2)	nr. days with interval session	total km Z3-4	max km Z3-4 one day	total km Z5-T1-T2	...	max training success.2	avg recovery.2	min recovery.2	max recovery.2	Athlete ID	injury	rel total kms week 0_1	rel total kms week 0_2
0	5.0	2.0	22.2	16.4	11.8	1.0	2.0	10.0	10.0	0.6	...	0.0	0.18	0.16	0.20	0	0	0.718447	1.378882
1	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	0.0	0.18	0.16	0.20	0	0	0.683544	1.018868
2	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	0.0	0.17	0.16	0.18	0	0	0.683544	1.018868
3	5.0	2.0	21.6	16.4	11.7	1.0	2.0	10.0	10.0	0.5	...	0.0	0.18	0.16	0.18	0	0	0.683544	1.018868
4	6.0	1.0	39.2	17.6	18.9	1.0	3.0	17.2	10.0	0.5	...	0.0	0.17	0.16	0.18	0	0	2.202247	1.361111
5	6.0	1.0	39.2	17.6	18.9	1.0	3.0	17.2	10.0	0.5	...	0.0	0.17	0.16	0.18	0	0	2.202247	1.361111
6	6.0	1.0	33.3	17.6	15.4	1.0	3.0	13.7	7.2	0.5	...	0.0	0.17	0.16	0.18	0	0	1.500000	1.077670
7	5.0	2.0	33.3	17.6	15.4	1.0	3.0	13.7	7.2	0.5	...	0.0	0.17	0.16	0.18	0	0	1.500000	1.077670
8	5.0	2.0	32.9	17.6	15.5	0.0	3.0	14.3	7.2	0.0	...	0.0	0.17	0.16	0.18	0	0	1.523148	1.041139
9	5.0	2.0	32.9	17.6	15.5	0.0	3.0	14.3	7.2	0.0	...	0.0	0.17	0.16	0.18	0	0	1.523148	1.041139

10 rows × 72 columns

إن مجموعة البيانات التي لدينا عالية الأبعاد إلى حد كبير، مما يجعل من الصعب البدءفي تحليل البيانات. لتبسيط العملية، قررنا إسقاط سمات معينة بناءً على التحليل التجريبي empirical analysis. على وجه التحديد، قمنا بإزالة السمات المتعلقة بكيفية شعور الشخص person feels، حيث أردنا

التركيز على التنبؤ بالبيانات بناءً على جودة الجري الكمية فقط. على الرغم من أن السمات المتعلقة بالتعافي recovery يمكن أن توفر رؤى قيمة، إلا أن الاعتماد فقط على أسئلة الاستطلاع لقياس الحالة البدنية للعداء أمر صعب وقد لا يؤدي إلى نتائج دقيقة. للحفاظ على الموضوعية وتبني نهج قائم على البياناتفي تحليلنا، اخترنا استبعاد السمات الذاتية والتركيز على عوامل أكثر واقعية وقابلية للقياس.

```
dataframe.info()
dataframe = dataframe.drop(['avg training success.2', 'max training success
.2', 'min training success.2', 'avg exertion', 'min exertion', 'max exertio
n'], axis = 1)
dataframe = dataframe.drop(['avg exertion.1', 'min exertion.1', 'max exerti
on.1', 'avg exertion.2', 'min exertion.2', 'max exertion.2', 'max km one da
y'], axis = 1)
dataframe = dataframe.drop(['avg recovery', 'min recovery', 'max recovery',
 'avg recovery.1', 'min recovery.1', 'max recovery.1', 'avg recovery.2', 'm
in recovery.2', 'max recovery.2'], axis = 1)
dataframe = dataframe.drop(['avg training success', 'min training success',
 'max training success', 'avg training success.1', 'max training success.1'
, 'min training success.1'], axis = 1)
dataframe = dataframe.drop(['rel total kms week 0_1', 'rel total kms week 0
_2', 'rel total kms week 1_2'], axis = 1)
dataframe.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42798 entries, 0 to 42797
Data columns (total 72 columns):
 #   Column                                    Non-Null Count  Dtype
---  ------                                    --------------  -----
 0   nr. sessions                              42798 non-null  float64
 1   nr. rest days                             42798 non-null  float64
 2   total kms                                 42798 non-null  float64
 3   max km one day                            42798 non-null  float64
 4   total km Z3-Z4-Z5-T1-T2                   42798 non-null  float64
 5   nr. tough sessions (effort in Z5, T1 or T2)  42798 non-null  float64
 6   nr. days with interval session            42798 non-null  float64
 7   total km Z3-4                             42798 non-null  float64
 8   max km Z3-4 one day                       42798 non-null  float64
 9   total km Z5-T1-T2                         42798 non-null  float64
 10  max km Z5-T1-T2 one day                   42798 non-null  float64
 11  total hours alternative training          42798 non-null  float64
 12  nr. strength trainings                    42798 non-null  float64
 13  avg exertion                              42798 non-null  float64
 14  min exertion                              42798 non-null  float64
 15  max exertion                              42798 non-null  float64
 16  avg training success                      42798 non-null  float64
 17  min training success                      42798 non-null  float64
 18  max training success                      42798 non-null  float64
 19  avg recovery                              42798 non-null  float64
 20  min recovery                              42798 non-null  float64
 21  max recovery                              42798 non-null  float64
 22  nr. sessions.1                            42798 non-null  float64
 23  nr. rest days.1                           42798 non-null  float64
 24  total kms.1                               42798 non-null  float64
 25  max km one day.1                          42798 non-null  float64
 26  total km Z3-Z4-Z5-T1-T2.1                 42798 non-null  float64
 27  nr. tough sessions (effort in Z5, T1 or T2).1  42798 non-null  float64
 28  nr. days with interval session.1          42798 non-null  float64
 29  total km Z3-4.1                           42798 non-null  float64
 30  max km Z3-4 one day.1                     42798 non-null  float64
 31  total km Z5-T1-T2.1                       42798 non-null  float64
 32  max km Z5-T1-T2 one day.1                 42798 non-null  float64
```

```
 33  total hours alternative training.1             42798 non-null  float64
 34  nr. strength trainings.1                       42798 non-null  float64
 35  avg exertion.1                                 42798 non-null  float64
 36  min exertion.1                                 42798 non-null  float64
 37  max exertion.1                                 42798 non-null  float64
 38  avg training success.1                         42798 non-null  float64
 39  min training success.1                         42798 non-null  float64
 40  max training success.1                         42798 non-null  float64
 41  avg recovery.1                                 42798 non-null  float64
 42  min recovery.1                                 42798 non-null  float64
 43  max recovery.1                                 42798 non-null  float64
 44  nr. sessions.2                                 42798 non-null  float64
 45  nr. rest days.2                                42798 non-null  float64
 46  total kms.2                                    42798 non-null  float64
 47  max km one day.2                               42798 non-null  float64
 48  total km Z3-Z4-Z5-T1-T2.2                      42798 non-null  float64
 49  nr. tough sessions (effort in Z5, T1 or T2).2  42798 non-null  float64
 50  nr. days with interval session.2               42798 non-null  float64
 51  total km Z3-4.2                                42798 non-null  float64
```

```
 52  max km Z3-4 one day.2                          42798 non-null  float64
 53  total km Z5-T1-T2.2                            42798 non-null  float64
 54  max km Z5-T1-T2 one day.2                      42798 non-null  float64
 55  total hours alternative training.2             42798 non-null  float64
 56  nr. strength trainings.2                       42798 non-null  float64
 57  avg exertion.2                                 42798 non-null  float64
 58  min exertion.2                                 42798 non-null  float64
 59  max exertion.2                                 42798 non-null  float64
 60  avg training success.2                         42798 non-null  float64
 61  min training success.2                         42798 non-null  float64
 62  max training success.2                         42798 non-null  float64
 63  avg recovery.2                                 42798 non-null  float64
 64  min recovery.2                                 42798 non-null  float64
 65  max recovery.2                                 42798 non-null  float64
 66  Athlete ID                                     42798 non-null  int64
 67  injury                                         42798 non-null  int64
 68  rel total kms week 0_1                         42798 non-null  float64
 69  rel total kms week 0_2                         42798 non-null  float64
 70  rel total kms week 1_2                         42798 non-null  float64
```

```
 71  Date                                           42798 non-null  int64
dtypes: float64(69), int64(3)
memory usage: 23.5 MB
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 42798 entries, 0 to 42797
Data columns (total 41 columns):
 #   Column                                         Non-Null Count  Dtype
---  ------                                         --------------  -----
 0   nr. sessions                                   42798 non-null  float64
 1   nr. rest days                                  42798 non-null  float64
 2   total kms                                      42798 non-null  float64
 3   total km Z3-Z4-Z5-T1-T2                        42798 non-null  float64
 4   nr. tough sessions (effort in Z5, T1 or T2)    42798 non-null  float64
 5   nr. days with interval session                 42798 non-null  float64
 6   total km Z3-4                                  42798 non-null  float64
 7   max km Z3-4 one day                            42798 non-null  float64
 8   total km Z5-T1-T2                              42798 non-null  float64
 9   max km Z5-T1-T2 one day                        42798 non-null  float64
 10  total hours alternative training               42798 non-null  float64
```

```
 11  nr. strength trainings                       42798 non-null  float64
 12  nr. sessions.1                               42798 non-null  float64
 13  nr. rest days.1                              42798 non-null  float64
 14  total kms.1                                  42798 non-null  float64
 15  max km one day.1                             42798 non-null  float64
 16  total km Z3-Z4-Z5-T1-T2.1                    42798 non-null  float64
 17  nr. tough sessions (effort in Z5, T1 or T2).1 42798 non-null  float64
 18  nr. days with interval session.1             42798 non-null  float64
 19  total km Z3-4.1                              42798 non-null  float64
 20  max km Z3-4 one day.1                        42798 non-null  float64
 21  total km Z5-T1-T2.1                          42798 non-null  float64
 22  max km Z5-T1-T2 one day.1                    42798 non-null  float64
 23  total hours alternative training.1           42798 non-null  float64
 24  nr. strength trainings.1                     42798 non-null  float64
 25  nr. sessions.2                               42798 non-null  float64
 26  nr. rest days.2                              42798 non-null  float64
 27  total kms.2                                  42798 non-null  float64
 28  max km one day.2                             42798 non-null  float64
 29  total km Z3-Z4-Z5-T1-T2.2                    42798 non-null  float64
 29  total km Z3-Z4-Z5-T1-T2.2                    42798 non-null  float64
 30  nr. tough sessions (effort in Z5, T1 or T2).2 42798 non-null  float64
 31  nr. days with interval session.2             42798 non-null  float64
 32  total km Z3-4.2                              42798 non-null  float64
 33  max km Z3-4 one day.2                        42798 non-null  float64
 34  total km Z5-T1-T2.2                          42798 non-null  float64
 35  max km Z5-T1-T2 one day.2                    42798 non-null  float64
 36  total hours alternative training.2           42798 non-null  float64
 37  nr. strength trainings.2                     42798 non-null  float64
 38  Athlete ID                                   42798 non-null  int64
 39  injury                                       42798 non-null  int64
 40  Date                                         42798 non-null  int64
dtypes: float64(38), int64(3)
memory usage: 13.4 MB
```

بعد تحليل دقيق، نجحنافي تقليل عدد السماتفي مجموعة البيانات من 72 إلى 41. وفي حين أن هذا التقدم واعد، إلا أن مجموعة البيانات لا تزال عالية الأبعاد، مما يشكل تحديًا لمزيد من التحليل والنمذجة.

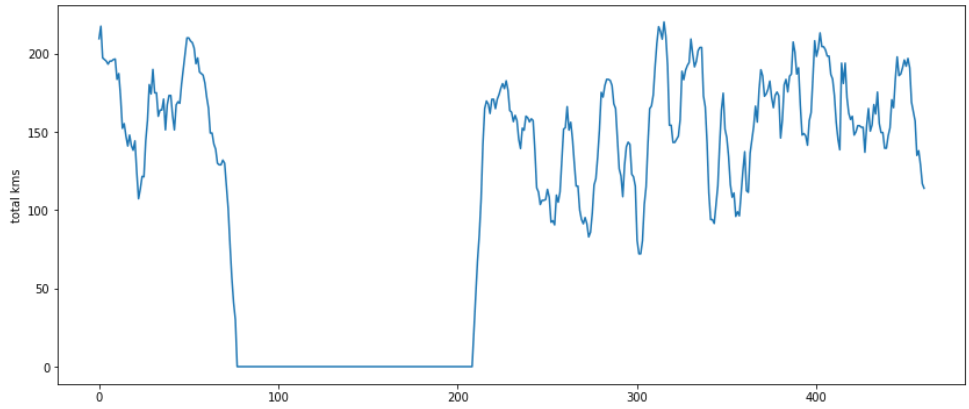
```
dataframe['Athlete ID'].unique()
```

```
array([ 0,  1,  2,  3,  4,  5,  6,  7,  8,  9, 10, 11, 12, 13, 14, 15, 16,
       17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33,
       34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50,
       51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67,
       68, 69, 70, 71, 72, 73])
```

في المجمل، هناك 74 رياضيًافي مجموعة البيانات. الآن، دعنا نركز على فحص بيانات التدريب لأول رياضي للحصول على رؤى حول أنماط تدريبه.

```
def indexIndividualData(id):
  df_0 = dataframe[dataframe['Athlete ID'] == id]
  index1 = df_0.index[0]
  indexLast = df_0.index[-1]
  y = indexLast - len(df_0[df_0['injury']==0]) - len(df_0[df_0['injury']==1
])
  df_0 = df_0.rename(index = lambda x: x - y - 1 if x > indexLast - len(df_
0[df_0['injury']==1]) else x - index1)
  df_0 = df_0.sort_values(by = 'Date')
  return df_0
def plotIndividualData(id, column):
  df_0 = indexIndividualData(id)
  plt.figure(figsize = (14,6))
  sns.lineplot(data=df_0[column])
```

```
plotIndividualData(1, "total kms")
```



يثير هذا الرسم البياني بعض الأسئلة المثيرة للاهتمام. فنحن نتساءل لماذا هناك انخفاض كبيرفي التدريب لفترة طويلة كهذه بالنسبة لهذا الرياضي بعينه. فإذا كانت كل نقطة بيانات تمثل أسبوعًا، فإن الإصابة لأكثر من مائة أسبوع تبدو غير عادية. ومن ناحية أخرى، إذا كانت النقاط تشير إلى أيام، فإن مائة يوم من الإصابة لا تزال كبيرة جدًا وتستحق المزيد من التحقيق.

إن السمات التي تحمل '1.' و'2.' و'3.'في مجموعة البيانات محيرة. ونجد صعوبةفي فهم أهميتها فيما يتعلق بسمة التواريخ، حتى بعد الرجوع إلى ملف البيانات الوصفية Metadata file. ولا يزال معنى هذه السمات وأهميتها غير واضحين ويستحقان مزيدًا من التوضيح.

```
def dateInjurySubset(id, column1, column2, column3):
  df_0 = indexIndividualData(id)
  return df_0[[column1,column2,column3]]



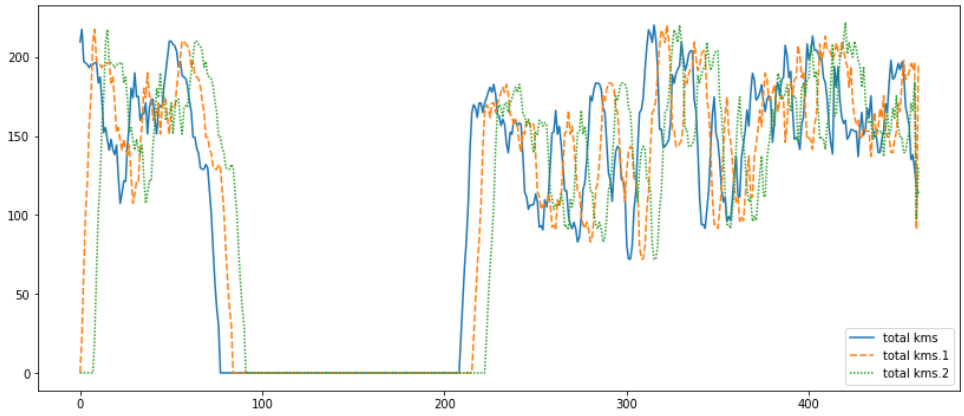
def plotIndividualDataDuoColumn(id, column1, column2, column3):
  df_0 = dateInjurySubset(id,column1,column2,column3)
  plt.figure(figsize = (14,6))
  sns.lineplot(data=df_0)

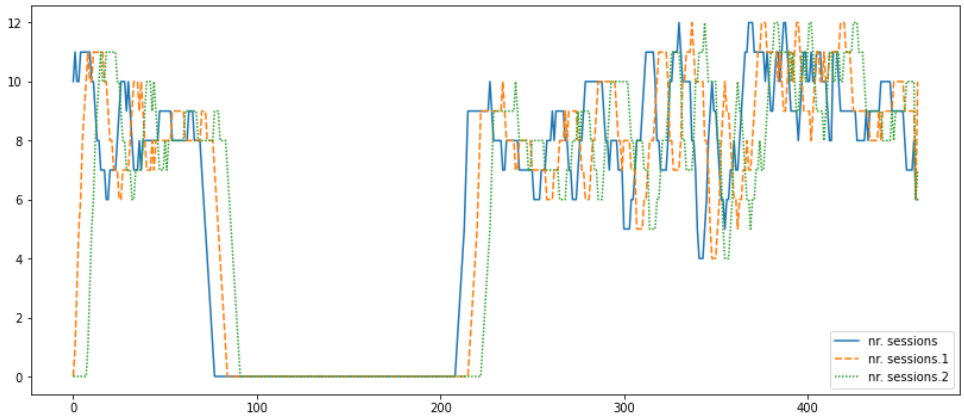
print(dateInjurySubset(1,"total kms", "total kms.1","total kms.2").mean())

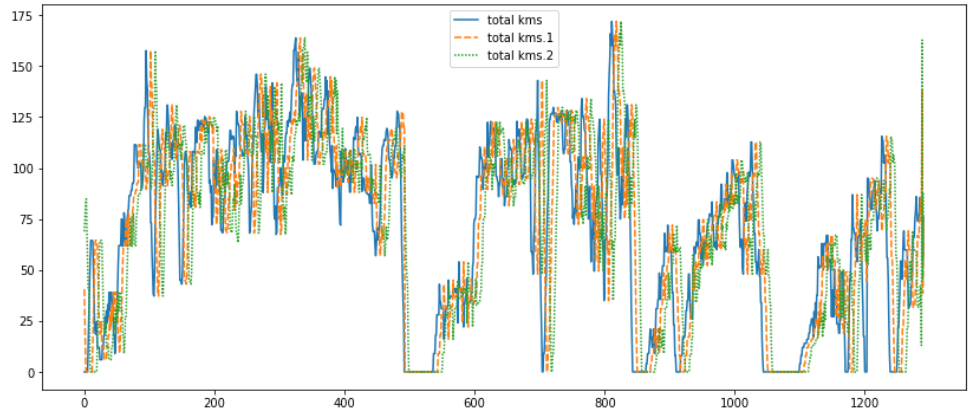
plotIndividualDataDuoColumn(1,"total kms", "total kms.1", "total kms.2")
print(dateInjurySubset(1,"nr. sessions", "nr. sessions.1","nr. sessions.2")
.mean())
plotIndividualDataDuoColumn(1,"nr. sessions", "nr. sessions.1","nr. session
s.2")
print(dateInjurySubset(2,"total kms", "total kms.1","total kms.2").mean())

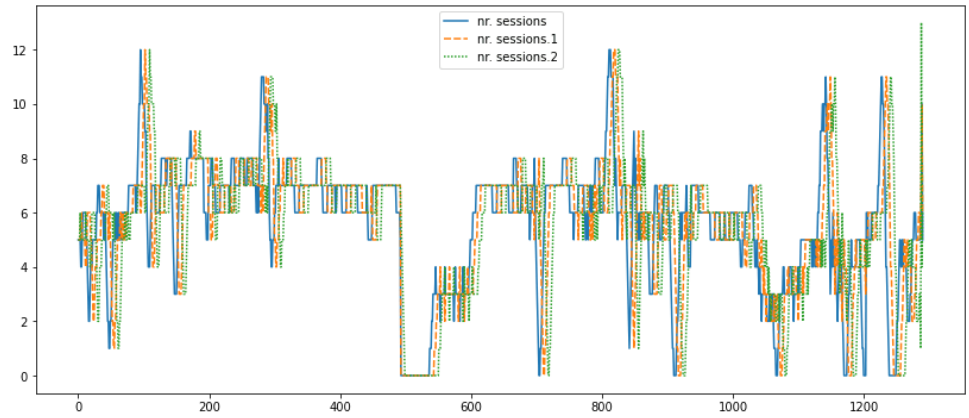
plotIndividualDataDuoColumn(2,"total kms", "total kms.1", "total kms.2")
print(dateInjurySubset(2,"nr. sessions", "nr. sessions.1","nr. sessions.2")
.mean())
plotIndividualDataDuoColumn(2,"nr. sessions", "nr. sessions.1","nr. session
s.2")
```

```
total kms       110.078308
total kms.1     109.772017
total kms.2     106.549675
dtype: float64
nr. sessions      6.017354
nr. sessions.1    6.010846
nr. sessions.2    5.865510
dtype: float64
total kms       71.518745
total kms.1     71.278621
total kms.2     70.798683
dtype: float64
nr. sessions      5.557707
nr. sessions.1    5.557707
nr. sessions.2    5.525174
dtype: float64
```










```
dfQ2 = dataframe[['Athlete ID', 'total km Z3-Z4-Z5-T1-T2', 'total km Z3-Z4-
Z5-T1-T2.1', 'total km Z3-Z4-Z5-T1-
T2.2', 'injury', 'nr. tough sessions (effort in Z5, T1 or T2)', 'nr. tough
sessions (effort in Z5, T1 or T2).1', 'nr. tough sessions (effort in Z5, T1
 or T2).2', 'total km Z5-T1-T2', 'total km Z5-T1-T2.1', 'total km Z5-T1-
T2.2', 'total km Z3-4', 'total km Z3-4.1', 'total km Z3-4.2']]
dfArray = []
for i in dataframe['Athlete ID'].unique():
  dfArray.append(indexIndividualData(i))
```

إن سمة "التواريخ dates"في هذه المجموعة من البيانات محيرة أيضًا. وللحصول على فهم أكثر وضوحًا لأهميتها، سنحاول تصورها واستكشاف أنماطها.

```
df_0 = dfArray[1]
injury = df_0[df_0['injury'] == 1]
notInjured = df_0[df_0['injury'] == 0]
print("INJURED DATES ID 1")
for i in injury['Date']:
  print(i)
print("NOT INJURED DATES ID 1:\n")
for i in notInjured['Date']:
  print(i)
```

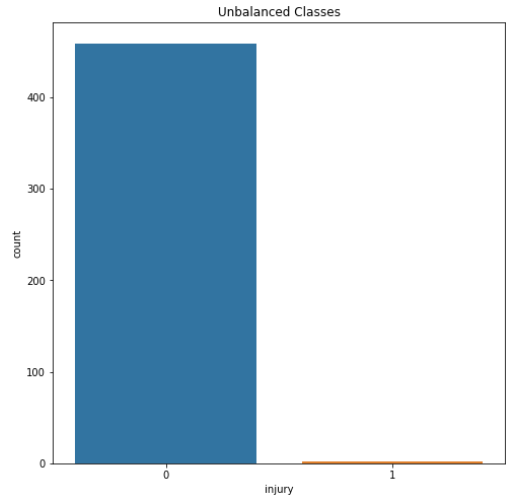
```
INJURED DATES ID 1
672
765
NOT INJURED DATES ID 1:

0
1
2
3
4
5
6
7
8
9
10
11
12
13
```

```
725
726
727
728
729
730
731
732
733
734
735
736
737
738
739
740
741
742
743
```

إن القفزة المفاجئةفي سمة "dates" من 400 إلى 700 لرياضي ما أمر محير ويثير تساؤلات. ولإجراء تنبؤات دقيقة وتحليل البيانات بشكل فعال، نحتاج إلى تواريخ متتالية ومتتالية. وعلى الرغم من بعض عدم اليقين والرؤى المفقودة المحتملة، سنظل نحاول تصنيف بيانات الجري. وعلاوة على ذلك، كشف استكشافنا للبيانات عن تحيز كبير تجاه نقاط البيانات غير المصابةفي مجموعة البيانات. وقد يؤثر هذا الخلل على أداء النموذج وتوقعاته.

```
plt.figure(figsize=(8, 8))
sns.countplot(x=dataframe["injury"])
plt.title('Unbalanced Classes')
plt.show()
```



في هذا التصور، أصبح واضحًا مدى انحراف مجموعة البيانات بشكل كبير تجاه الحالات غير المصابة.

```
df_1 = df_0.sort_values(by = 'Athlete ID');

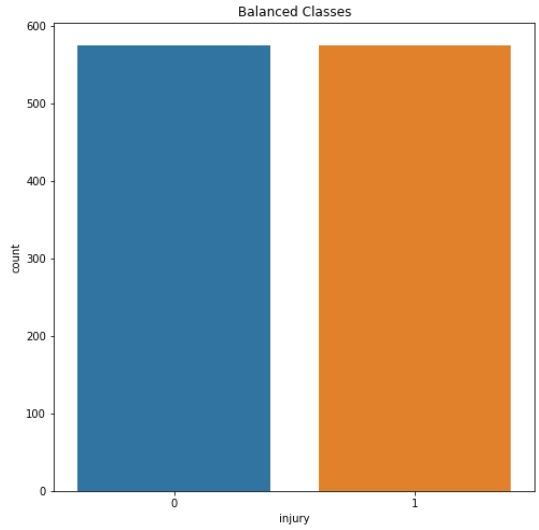
shuffled_df_1 = dataframe.sample(frac=1,random_state=4)

# Put all the fraud class in a separate dataset.
injury_df_1 = shuffled_df_1.loc[shuffled_df_1['injury'] == 1]

#Randomly select 492 observations from the non-fraud (majority class)
non_injured_df_1 = shuffled_df_1.loc[shuffled_df_1['injury'] == 0].sample(n
=575)

# Concatenate both dataframes again
normalized_df = pd.concat([injury_df_1, non_injured_df_1])

#plot the dataset after the undersampling
plt.figure(figsize=(8, 8))
sns.countplot(x=normalized_df['injury'])
plt.title('Balanced Classes')
plt.show()
```



من أجل منع الضبط الزائد overfitting في نماذجنا التنبؤية، قمنا بموازنة مجموعة البيانات باستخدام تقنيات أخذ العينات. وهذا يضمن تمثيل الحالات المصابة وغير المصابة بالتساويفي بيانات التدريب.

النمذجة

سنستخدم هنا خوارزميات التعلم الآلي المختلفة جنبًا إلى جنب مع دقتها ومصفوفة الارتباك confusion matrix الخاصة بها.

لنبدأ بإظهار تأثير مجموعة البيانات المنحرفة skewed dataset.في مجموعة البيانات غير المتوازنة unbalanced dataset، يبدو أن دقة المصنف عالية جدًا لأنه يحدد ببساطة جميع نقاط البيانات على

أنها غير مصابة. ومع ذلك، فإن هذا النهج غير مفيد لأهداف هذا المشروع، لأنه يفشل في التنبؤ بدقة بحالات الإصابة.

```
y = df_0['injury']
X = df_0.drop('injury', axis=1)
X = df_0.drop('Athlete ID', axis=1)
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.3, random_state = 0)

K = []
training = []
test = []
scores = {}

for k in range(2, 21):
    clf = KNeighborsClassifier(n_neighbors = k)
    clf.fit(X_train, y_train)

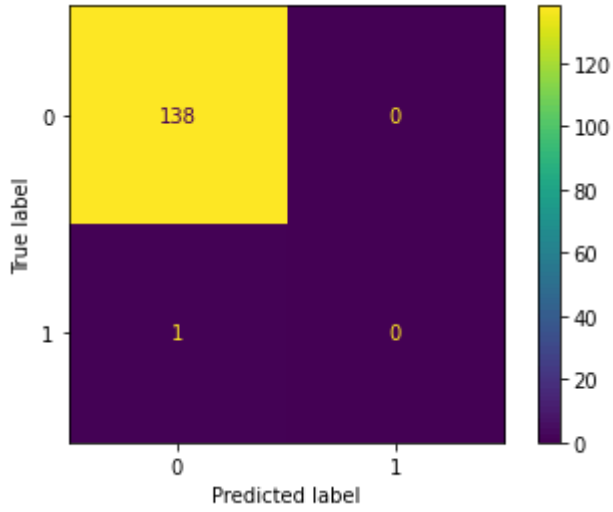
    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

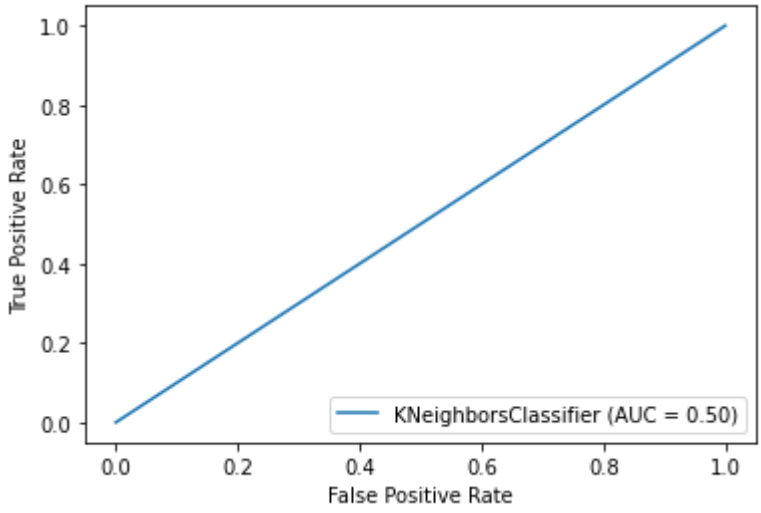
    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]

for keys, values in scores.items():
    print(keys, ':', values)
```

```
2 : [0.9968944099378882, 0.9928057553956835]
3 : [0.9968944099378882, 0.9928057553956835]
4 : [0.9968944099378882, 0.9928057553956835]
5 : [0.9968944099378882, 0.9928057553956835]
6 : [0.9968944099378882, 0.9928057553956835]
7 : [0.9968944099378882, 0.9928057553956835]
8 : [0.9968944099378882, 0.9928057553956835]
9 : [0.9968944099378882, 0.9928057553956835]
10 : [0.9968944099378882, 0.9928057553956835]
11 : [0.9968944099378882, 0.9928057553956835]
12 : [0.9968944099378882, 0.9928057553956835]
13 : [0.9968944099378882, 0.9928057553956835]
14 : [0.9968944099378882, 0.9928057553956835]
15 : [0.9968944099378882, 0.9928057553956835]
16 : [0.9968944099378882, 0.9928057553956835]
17 : [0.9968944099378882, 0.9928057553956835]
18 : [0.9968944099378882, 0.9928057553956835]
19 : [0.9968944099378882, 0.9928057553956835]
20 : [0.9968944099378882, 0.9928057553956835]
```

```
from sklearn.metrics import classification_report, plot_confusion_matrix
clf.fit(X_train, y_train)
plot confusion matrix(clf, X test, y test)
metrics.plot_roc_curve(clf, X_test, y_test)
plt.show()
```





في مصفوفة الارتباك الموضحة أعلاه، يتم تصنيف جميع نقاط البيانات على أنها غير مصابة. ولمعالجة هذه المشكلة، سننتقل الآن إلى استخدام مجموعة البيانات المتوازنة.

```
y = normalized_df['injury']
X = normalized_df.drop('injury', axis=1)
X = normalized_df.drop('Athlete ID', axis=1)
X_train, X_test, y_train, y_test = train_test_split(
             X, y, test_size = 0.3, random_state = 0)

K = []
training = []
test = []
scores = {}
```

```
for k in range(2, 21):
    clf = KNeighborsClassifier(n_neighbors = k)
    clf.fit(X_train, y_train)

    training_score = clf.score(X_train, y_train)
    test_score = clf.score(X_test, y_test)
    K.append(k)

    training.append(training_score)
    test.append(test_score)
    scores[k] = [training_score, test_score]

for keys, values in scores.items():
    print(keys, ':', values)
```

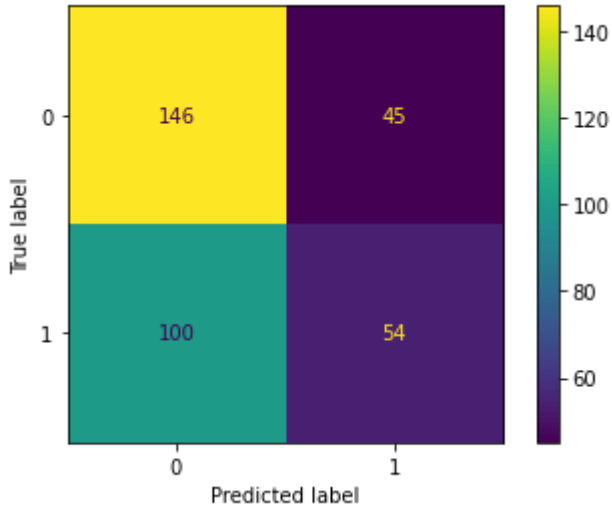
```
    2 : [0.7987577639751553, 0.5797101449275363]
    3 : [0.7937888198757764, 0.5855072463768116]
    4 : [0.7515527950310559, 0.5855072463768116]
    5 : [0.7552795031055901, 0.5826086956521739]
    6 : [0.7453416149068323, 0.5565217391304348]
    7 : [0.7192546583850932, 0.5652173913043478]
    8 : [0.7105590062111802, 0.5681159420289855]
    9 : [0.6993788819875777, 0.5710144927536231]
    10 : [0.7080745341614907, 0.591304347826087]
    11 : [0.684472049689441, 0.6028985507246377]
    12 : [0.6906832298136646, 0.5942028985507246]
    13 : [0.6708074534161491, 0.591304347826087]
    14 : [0.6745341614906832, 0.5855072463768116]
    15 : [0.653416149068323, 0.6028985507246377]
    16 : [0.6459627329192547, 0.5797101449275363]
    17 : [0.6447204968944099, 0.6057971014492753]
    18 : [0.6484472049689441, 0.6173913043478261]
    19 : [0.6546583850931676, 0.6057971014492753]
    20 : [0.6571428571428571, 0.6057971014492753]
```

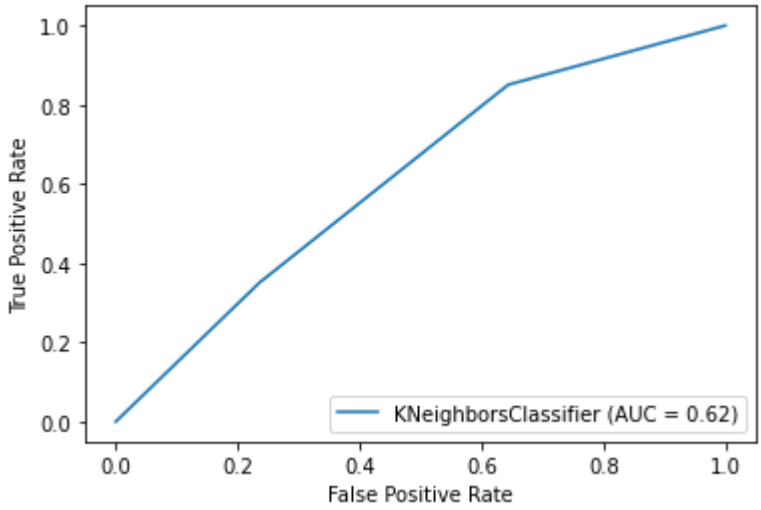
في هذا التحليل، نلاحظ أنه مع زيادة المعامل k، تنخفض دقة التدريب بشكل ملحوظ (من حوالي 80% إلى 65%)، بينما تتحسن دقة الاختبار قليلاً (من حوالي 58% إلى 60%). لتقييم أداء النموذج في التعامل مع البيانات المصابة وغير المصابة، سنستخدم مصفوفة الارتباك.

```
clf = KNeighborsClassifier(n_neighbors = 2)
 clf.fit(X_train, y_train)

 training_score = clf.score(X_train, y_train)
 test_score = clf.score(X_test, y_test)
 K.append(k)

 training.append(training_score)
 test.append(test_score)
 scores[k] = [training_score, test_score]
clf.fit(X_train, y_train)
plot_confusion_matrix(clf, X_test, y_test)
metrics.plot_roc_curve(clf, X_test, y_test)
plt.show()
```





مع قيمة k تساوي 2، يظهر المصنف دقة أعلىفي التنبؤ بالبيانات غير المصابة.

```
clf = KNeighborsClassifier(n_neighbors = 12)
 clf.fit(X_train, y_train)

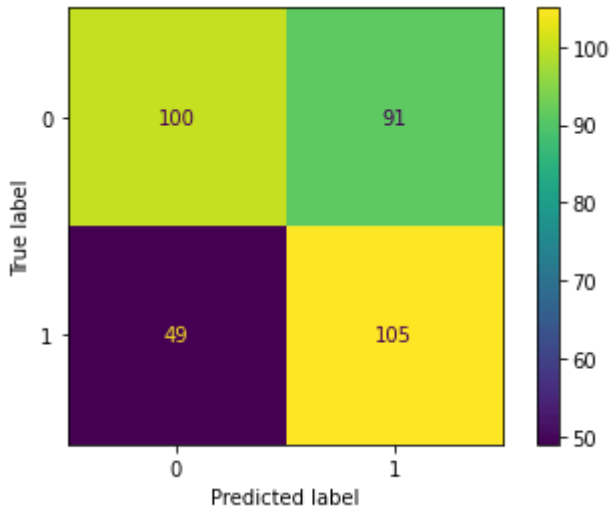
 training_score = clf.score(X_train, y_train)
 test_score = clf.score(X_test, y_test)
 K.append(k)

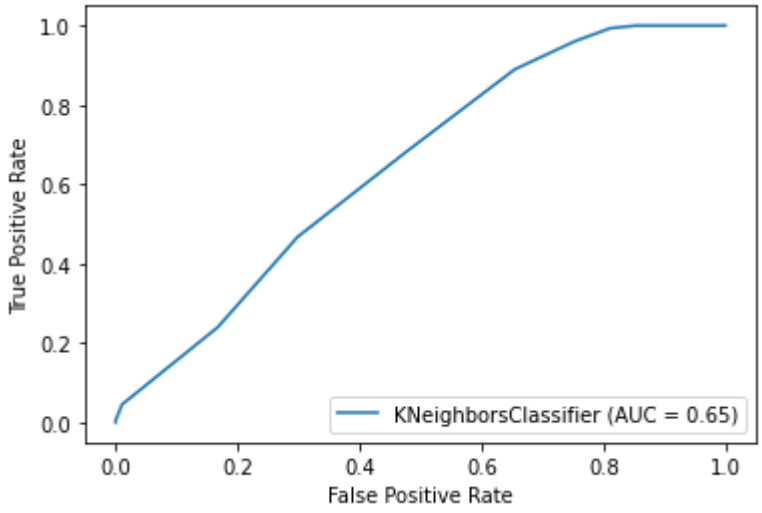
 training.append(training_score)
 test.append(test_score)
 scores[k] = [training_score, test_score]

clf.fit(X_train, y_train)
print("Accuracy:",accuracy_score(Y_test, Y_pred))
```

```
plot_confusion_matrix(clf, X_test, y_test)
metrics.plot_roc_curve(clf, X_test, y_test)
plt.show()
```

```
Accuracy: 0.5738317757009346
```





عند استخدام قيمة k عالية تبلغ 21، يصبح التنبؤ بالإصابة أكثر دقة. ومع ذلك، يؤدي هذا أيضًا إلى زيادة كبيرةفي الإيجابيات الخاطئة false positives للبيانات غير المصابة، مما يعني أن المصنف يحدد عن طريق الخطأ المزيد من الأشخاص غير المصابين على أنهم مصابون.

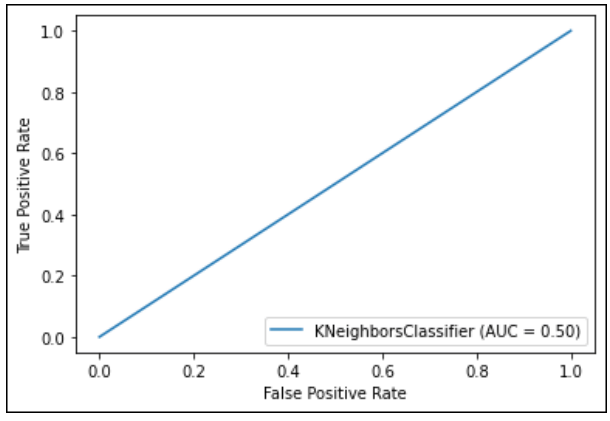
بعد إجراء تجارب مكثفة بقيم معلمات مختلفة، حددنا نقطة التوازن المثلى بقيمة k تبلغ 12. حقق المصنف معدل دقة إجماليًا بنسبة 60%، حيث أظهر دقة بنسبة 52%في التنبؤ بنقاط البيانات غير المصابة ودقة بنسبة 68%في التنبؤ بنقاط البيانات المصابة. ونظرًا للتحيز المتأصل في مجموعة البيانات، فإن هذا المستوى من الدقة جدير بالثناء. ومع ذلك، نواصل استكشاف المصنفات الثنائية البديلة وطرق الموازنة المختلفة لتعزيز دقة تنبؤاتنا بشكل أكبر.

يوضح الكود أدناه استخدام مصنف آلة متجه الدعم support vector machine classifier لإجراء تنبؤات بالإصابة لنقاط البيانات. للتعامل مع الطبيعة غير المتوازنة لمجموعة البيانات الخاصة بنا، نعتمد تقنية أخذ العينات غير الكافية undersampling technique. من خلال تحقيق التوازن بين تمثيل الحالات المصابة وغير المصابة، فإننا نهدف إلى تحسين دقة المصنف في تقديم التوقعات لكلا الفئتين.

```
#SVM classifier using undersampling
from imblearn.under_sampling import RandomUnderSampler
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
rus = RandomUnderSampler(random_state=0)
X_train, Y_train =rus.fit_resample(X_train,Y_train)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print("Accuracy:",accuracy_score(Y_test, Y_pred))
print(confusion_matrix(Y_test, Y_pred))


metrics.plot_roc_curve(clf, X_test, Y_pred)
plt.show()
```

```
Accuracy: 0.5924299065420561
[[6253 4303]
 [  58   86]]
```

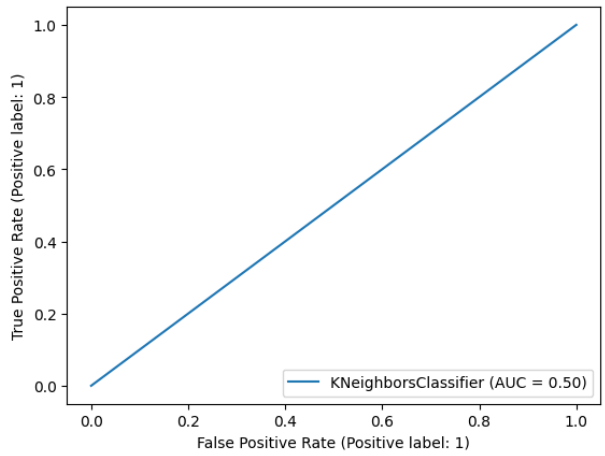


في الكود أدناه، نستخدم مصنفًا لآلة متجه الدعم للتنبؤ بما إذا كانت نقطة البيانات تتوافق مع إصابة أم لا. لمعالجة مشكلة مجموعة البيانات غير المتوازنة لدينا، نستخدم تقنية أخذ العينات الزائدة oversampling technique. يهدف هذا النهج إلى إنشاء تمثيل أكثر توازناً للبيانات من خلال تكرار أو إنشاء حالات إضافية من فئة الأقلية minority class (بيانات المصابين injured data)، مما يسمح للمصنف بتحقيق دقة أفضل في التنبؤ بحالات المصابين وغير المصابين.

```
#SVM Classifier using oversampling
from imblearn.over_sampling import SMOTE
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
sm = SMOTE(random_state = 0)
X_train, Y_train = sm.fit_resample(X_train,Y_train)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
classifier = SVC(kernel = 'rbf', random_state = 0)
classifier.fit(X_train, Y_train)
Y_pred = classifier.predict(X_test)
print("Accuracy:",accuracy_score(Y_test, Y_pred))


print(confusion_matrix(Y_test, Y_pred))
metrics.plot_roc_curve(clf, X_test, Y_pred)
plt.show()
```

```
Accuracy: 0.9376635514018692
[[10011   545]
 [  122    22]]
```



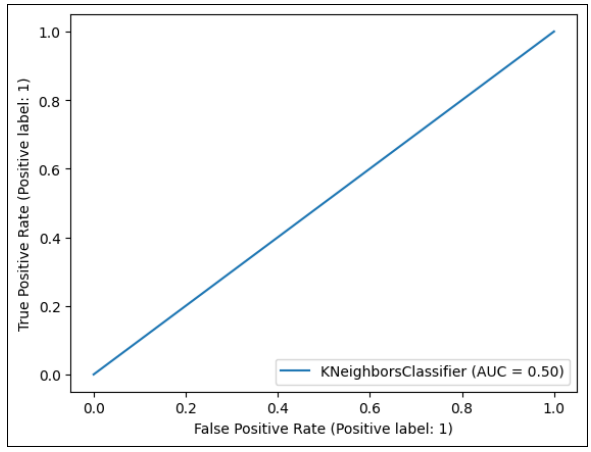
في الكود أدناه، نقوم بتنفيذ مصنف التعبئة bagging classifier مع أخذ عينات أقل من المطلوب للتنبؤ بما إذا كانت نقطة البيانات تتوافق مع إصابة أم لا.

```
#Bagging Classifier With Undersampling
```

```
import sklearn.ensemble
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
rus = RandomUnderSampler(random_state=0)
X_train, Y_train =rus.fit_resample(X_train,Y_train)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
bag = sklearn.ensemble.BaggingClassifier(n_estimators = 35)
bag.fit(X_train, Y_train)
Y_pred = bag.predict(X_test)
print("Accuracy:",accuracy_score(Y_test, Y_pred))


print(confusion_matrix(Y_test, Y_pred))
metrics.plot_roc_curve(clf, X_test, Y_test)
plt.show()
```

```
Accuracy: 0.5786915887850468
[[6099 4457]
 [  51   93]]
```



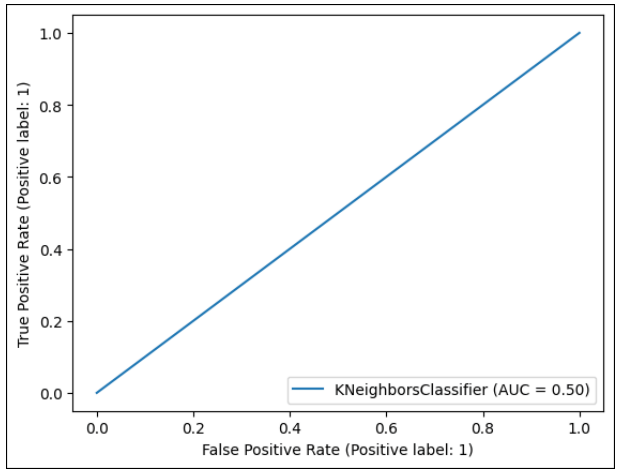
في الكود أدناه، نقوم بتنفيذ مصنف التعبئة مع الإفراطفي أخذ العينات oversampling للتنبؤ بما إذا كانت نقطة البيانات تتوافق مع إصابة أم لا.

```
#Bagging Classifier With Oversampling
import sklearn.ensemble
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
sm = SMOTE(random_state = 0)
X_train, Y_train = sm.fit_resample(X_train,Y_train)
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
bag = sklearn.ensemble.BaggingClassifier(n_estimators = 30)
```

```
bag.fit(X_train, Y_train)
Y_pred = bag.predict(X_test)
print("Accuracy:",accuracy_score(Y_test, Y_pred))

print(confusion_matrix(Y_test, Y_pred))
metrics.plot_roc_curve(clf, X_test, Y_test)
plt.show()
```

```
Accuracy: 0.9865420560747663
[[10556     0]
 [  144     0]]
```

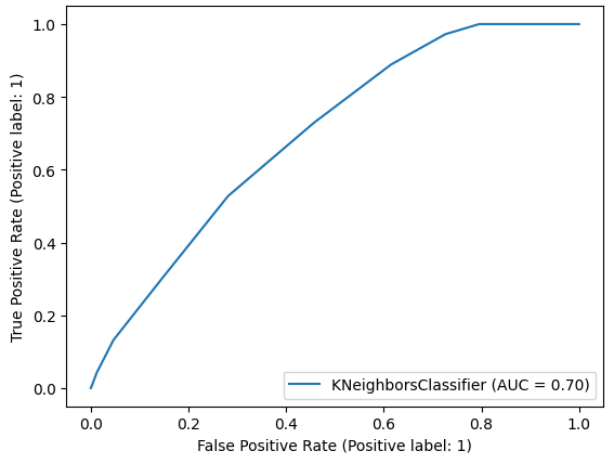


هنا، نستخدم مصنف التعبئة للتنبؤ بما إذا كانت نقطة البيانات قد تعرضت لضرر أو نستخدم تقنية أخذ العينات غير الكافية لمواجهة مجموعة البيانات غير المتوازنة لدينا.

```
#XGBooster model with undersampling
from xgboost import XGBClassifier
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
rus = RandomUnderSampler(random_state=0)
X_train, Y_train =rus.fit_resample(X_train,Y_train)
boost = XGBClassifier(max_depth = 3, n_estimators = 30)
boost.fit(X_train, Y_train)
Y_pred = boost.predict(X_test)
print("Accuracy:",accuracy_score(Y_test, Y_pred))

print(confusion_matrix(Y_test, Y_pred))
metrics.plot_roc_curve(clf, X_test, Y_test)
plt.show()
```

```
Accuracy: 0.5701869158878504
[[6009 4547]
 [  52   92]]
```

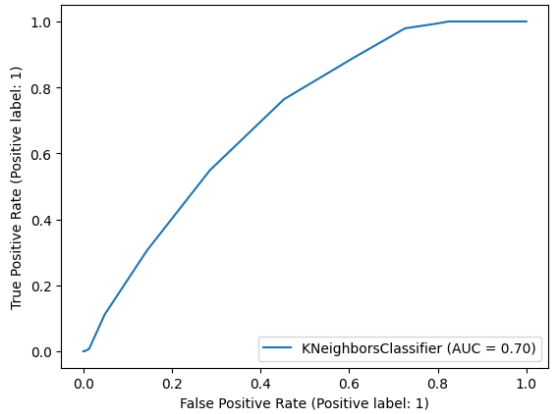


نحن نستخدم مصنف التعبئة bagging classifier لتحديد ما إذا كانت نقطة البيانات تتوافق مع إصابة، ومعالجة الخلل في مجموعة البيانات لدينا من خلال استخدام تقنية أخذ العينات الزائدة.

```
#XGBooster model with Oversampling
from xgboost import XGBClassifier
X = dataframe.drop('injury', axis = 1)
Y = dataframe['injury']
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size = 0.25,
 stratify = Y)
sm = SMOTE(random_state = 0)
X_train, Y_train = sm.fit_resample(X_train,Y_train)
boost = XGBClassifier(max_depth = 2, n_estimators = 30)
boost.fit(X_train, Y_train)
Y_pred = boost.predict(X_test)
print("Accuracy:",accuracy score(Y test, Y pred))

print(confusion_matrix(Y_test, Y_pred))
metrics.plot_roc_curve(clf, X_test, Y_test)
plt.show()
```

```
Accuracy: 0.9864485981308411
[[10555     1]
 [  144     0]]
```



تأثرت دقة النموذج بشكل أساسي بطريقة أخذ العينات المستخدمة. كان أداء نهج أخذ العينات الزائدة أفضل في تصنيف نقاط البيانات غير المصابة، بينما تفوقت طريقة أخذ العينات الناقصة في تحديد نقاط البيانات المصابة بدقة.

لموازنة مجموعة البيانات، طبقنا استراتيجيات مختلفة مثل أخذ العينات sampling، وأخذ العينات الزائدة oversampling، وأخذ العينات الناقصة. بعد ذلك، قمنا بتقييم مجموعات البيانات المتوازنة المختلفة باستخدام مصنفات ثنائية متعددة، بما في ذلك KNN وSVM وBagging وXGBooster. تم تحقيق أفضل أداء باستخدام XGBooster وBagging، مع معدل دقة مذهل يبلغ حوالي 99٪.

الجوانب المستقبلية للتنبؤ بالإصابة لدى العدائين التنافسيين باستخدام التعلم الآلي

يحمل التعلم الآلي وعدًا كبيرًا للطب الرياضي ورعاية الرياضيين. مع التقدم في تكنولوجيا الأجهزة القابلة للارتداء وطرق جمع البيانات، يمكن لنماذج التعلم الآلي تحليل مجموعات البيانات الشاملة، مما يوفر رؤى أعمق حول أداء الرياضيين والميكانيكا الحيوية والمعلمات الفسيولوجية. إن دمج البيانات متعددة الوسائط، مثل علم الوراثة والظروف البيئية، قد يكشف عن أنماط جديدة تساهم في خطر الإصابة. وستكون الدراسات الطولية حاسمة في بناء نماذج تنبؤ دقيقة من خلال مراقبة الرياضيين بمرور الوقت.

وعلاوة على ذلك، يمكن للتعلم الآلي أن يقدم خططًا مخصصة للوقاية من الإصابات بناءً على الخصائص الفردية. ومن خلال تبني هذه التطورات، يمكن أن تساعد في تقليل خطر الإصابة.

الاستنتاج

إن التنبؤ بالإصابات Injury prediction لدى العدائين التنافسيين باستخدام التعلم الآلي يحمل وعدًا هائلاً لتحويل الطب الرياضي وتعزيز رفاهية الرياضيين. وبمساعدة الخوارزميات المتقدمة ومجموعات البيانات الشاملة، تقدم نماذج التعلم الآلي تقييمات مخاطر الإصابة في الوقت الفعلي، وإرشادات تدريبية مخصصة، واستراتيجيات الوقاية من الإصابات القائمة على البيانات.

ومع ذلك، يمكن تحقيق الفوائد الكاملة للتعلم الآلي في التنبؤ بالإصابات من خلال معالجة التحديات مثل جودة البيانات، وقابلية تفسير النموذج، والمحاسبة عن تفاعلات عوامل الخطر المعقدة. سيكون التعاون بين علماء البيانات وخبراء الطب الرياضي والمدربين والرياضيين مفتاحًا للتغلب على هذه العقبات وإطلاق العنان للإمكانات الكاملة للتعلم الآلي في الجري التنافسي.

وبينما يستكشف الباحثون والممارسون الاحتمالات، يمكن للتنبؤ بالإصابات من خلال التعلم الآلي أن يحدث ثورة في كيفية تعامل العدائين التنافسيين مع التدريب والتعافي والوقاية من الإصابات، مما يؤدي في النهاية إلى مجتمع رياضي أكثر صحة ونجاحًا.

المصدر:

https://www.javatpoint.com/injury-prediction-in-competitive-runners-using-machine-learning

4) تحليل بيانات لاعبي كرة القدم باستخدام بايثون والتعلم الآلي Data Analysis on Football Players using Python & Machine Learning

لقد كنت أتعلم البرمجة باستخدام بايثون بشكل مستقل وقمت بإجراء تحليل استكشافي للبيانات على مجموعة بيانات للاعبي كرة القدم. يتضمن ذلك معالجة البيانات باستخدام pandas ورسم الرسوم البيانية باستخدام matplotlib وإنشاء نموذج تعلُّم آلي باستخدام scikit-learn.

هذا رابط للبيانات المستخدمة. (تم التقاط البياناتفي منتصف الموسم لذا لن تكون الأرقام محدثة).

إليك ما وجدته وكيف استخدمت بايثون للمساعدةفي الإجابة على أسئلتي:

استيراد مجموعة البيانات

بعد استيراد المكتبات ذات الصلة، قمت باستيراد مجموعة البيانات باستخدام الدالة ()read.

```
#importing libraries
import numpy as np
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt

#creating a dataframe from .csv file
df = pd.read_csv('C:\\Users\\Rahul\\Desktop\\2022-2023 Football Player
Stats.csv', encoding='latin-1')

#checking data for errors
df.info()
df.head()
```

	Rk	Player	Nation	Pos	Squad	Comp	Age	Born	MP	Starts	...	Off	Crs	TklW	PKwon	PKcon	OG	Recov	AerWon	AerLost	AerWon%
0	1	Brenden Aaronson	USA	MFFW	Leeds United	Premier League	22	2000	20	19	...	0.17	2.54	0.51	0.0	0.0	0.00	4.86	0.34	1.19	22.2
1	2	Yunis Abdelhamid	MAR	DF	Reims	Ligue 1	35	1987	22	22	...	0.05	0.18	1.59	0.0	0.0	0.00	6.64	2.18	1.23	64.0
2	3	Himad Abdelli	FRA	MFFW	Angers	Ligue 1	23	1999	14	8	...	0.00	1.05	1.40	0.0	0.0	0.00	8.14	0.93	1.05	47.1
3	4	Salis Abdul Samed	GHA	MF	Lens	Ligue 1	22	2000	20	20	...	0.00	0.35	0.80	0.0	0.0	0.05	6.60	0.50	0.50	50.0
4	5	Laurent Abergel	FRA	MF	Lorient	Ligue 1	30	1993	15	15	...	0.00	0.23	2.02	0.0	0.0	0.00	6.51	0.31	0.39	44.4

5 rows × 124 columns

الآن بعد استيراد البيانات، يمكننا البدءفي الإجابة عن بعض الأسئلة حول البيانات.

من سجل أكبر عدد من الأهداف في الدوري الإنجليزي الممتاز؟

باستخدام الدالة ()query. ، قمت بإنشاء إطار بيانات يحتوي فقط على لاعبي الدوري الإنجليزي الممتاز. ثم قمت بإنشاء إطار بيانات dataframe آخر لفرز اللاعبين حسب أكثر الأهداف المسجلة، وإظهار الأعمدة ذات الصلة فقط.

```
#querying data to create dataframe filtered to Premier League players
PL= df.query('Comp == "Premier League"')
```

```
#creating a view of the top 10 scorers
Top_Scorers =
PL.sort_values(by='Goals',ascending=False)[['Player','Nation','Pos','Squad'
,'Comp','Age','Goals']]

Top_Scorers.head(3)
```

	Player	Nation	Pos	Squad	Comp	Age	Goals
1057	Erling Haaland	NOR	FW	Manchester City	Premier League	22	25
1260	Harry Kane	ENG	FW	Tottenham	Premier League	29	17
2451	Ivan Toney	ENG	FW	Brentford	Premier League	26	14

كما تظهر البيانات، كان الهداف الأول هو إيرلينج هالاند (25) يليه هاري كين (17) وإيفان توني (14).

أي فريق سجل جماعيًا أكبر عدد من الأهداف في الدوري الإنجليزي الممتاز؟

للإجابة على هذا السؤال، احتجت إلى تجميع اللاعبين بناءً على فريقهم وتلخيص الأهداف المسجلة. يتم تحقيق ذلك باستخدام دالة ()groupby.

```
#grouping teams together
Top_Scoring_Teams =
PL.groupby(['Squad'])['Goals'].sum().reset_index().rename(columns={'Goals':
'Total Goals'})

#sorting by goals scored
Top_Scoring_Teams.sort_values(by='Total
Goals',ascending=False,inplace=True)

Top_Scoring_Teams.head(3)
```

	Squad	Total Goals
12	Manchester City	52
0	Arsenal	44
17	Tottenham	40

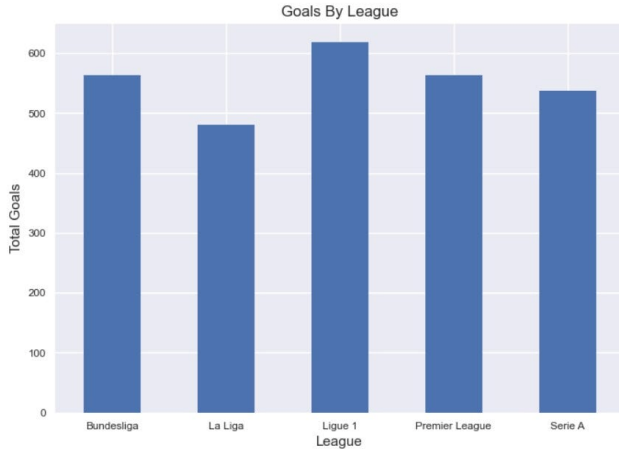
تظهر البيانات أن مانشستر سيتي (52) سجل أكبر عدد من الأهداف يليه آرسنال (44) وتوتنهام (40).

يشاهد صديقي جريج فقط مباريات كرة القدم التي تحتوي على الكثير من الأهداف، فأي الدوريات يجب على جريج مشاهدتها/تجنبها لرؤية أكبر عدد من الأهداف؟

يمكن الإجابة على هذا السؤال باستخدام نفس دوال التجميع grouping والفرز sorting من السؤال الأخير. لقد رسمت هذافي مخطط شريطي bar chart لتوضيح البيانات.

```
#grouping leagues together
League_Goals = df.groupby(['Comp'])['Goals'].sum()
```

```
#creating bar chart
matplotlib.style.use('seaborn-v0_8')
League_Goals.plot.bar(rot=0,title='Goals By
League',xlabel='League',ylabel='Total Goals',fontsize='small')
```



بناءً على هذه النتائج، يجب على جريج تجنب الدوري الإسباني لأنه الأقل تسجيلاً للأهداف، ويجب عليه مشاهدة الدوري الفرنسي لأنه الأكثر تسجيلاً للأهداف.

ما هو متوسط عدد الدقائق التي لعبها جميع اللاعبين في الموسم؟

يمكن استخدام الدالة ()mean. هنا لأخذ متوسط عدد الدقائق التي لعبها اللاعبونفي عمود الدقائق التي لعبت.

```
#calculating average minutes played
avg mins= df['Min'].mean()

#printing and rounding number to 2 decimal places
print(f'The average minutes played was {round(avg_mins,2)}')
```

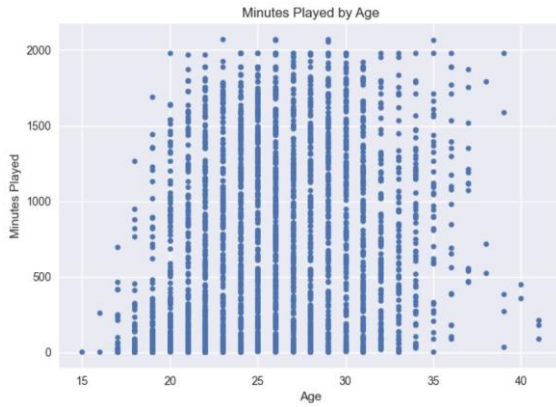
بلغ متوسط الدقائق التي لعبها جميع اللاعبين 760.45 دقيقة.

هل هناك علاقة بين العمر والدقائق التي لعبها اللاعبون؟

بدأت الإجابة على هذا السؤال بإنشاء مخطط تشتت scatterplot لإظهار الدقائق التي لعبها اللاعبون حسب العمر، على أمل أن أرى على الفور ارتباطًا واضحًا.

```
#creating a dataframe just showing players, age and minutes played
age_mins= df[['Player','Age','Min']].rename(columns={'Min':'Minutes
Played'})

#using the new dataframe to create a scatter plot
age_mins.plot.scatter(x='Age',y='Minutes Played',title='Minutes Played by
Age')
```



كان هناك ببساطة عدد كبير جدًا من نقاط البيانات بحيث لم أتمكن من رؤية أي اتجاه واضح، كما أنني لا أستطيع رؤية المناطق الأكثر كثافة بسهولة.

ولعلاج هذه المشكلة، أخذت متوسطات كل عمر ورسمت مخططًا آخر للنقاط.

```
#grouping by age and averaging the number of minutes played
avg_age_mins= age_mins.groupby(['Age'])['Minutes
Played'].mean().reset_index().rename(columns={'Minutes Played':'Average
Minutes Played'})

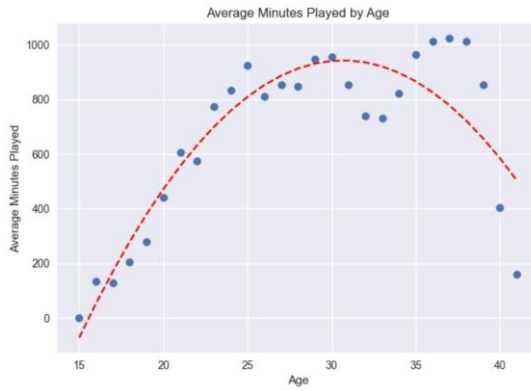
#creating scatter plot
x= avg_age_mins['Age']
y= avg_age_mins['Average Minutes Played']

plt.scatter(x,y)
plt.title("Average Minutes Played by Age")
plt.xlabel("Age")
plt.ylabel("Average Minutes Played")

#fitting a trendline
z= np.polyfit(x,y,2)
p= np.poly1d(z)

plt.plot(x,p(x),"r--")

plt.show()
```



إنشاء نموذج تعلُّم آلي للتنبؤ بالدقائق التي لعبها اللاعب بناءً على العمر

باستخدام Sci-Kit Learn، قمت بإنشاء نموذج تعلُّم آلي قمت بتدريبه باستخدام مجموعة البيانات هذه. يعتمد النموذج على دالة انحدار regression function، ويمكننا استخدام ناتج الانحدار للتنبؤ بالنتائج بناءً على قيمة الإدخال الخاصة بنا.

في هذه الحالة، يكون الإدخال هو عمر اللاعب ويخرج النموذج دقائق اللعب، وهو عدد الدقائق التي يتوقع النموذج أن يلعبها فريق عمري

نظرًا لوجود اتجاه عكسي على شكل حرف "U"في البيانات، فقد استخدمت مكتبة PolynomialFeatures لتطبيق هذا التحويل على نموذج الانحدار.

```
#setting variables:
age_mins.sort_values(by='Age',inplace=True)
X= age_mins['Age'].values.reshape(-1,1)
Y= age mins['Minutes Played']

#creating machine learning model
from sklearn.model_selection import train_test_split
X_train, X_test, Y_train, Y_test = train_test_split(X,Y, test_size=0.2)

#adding polynomial features to model
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(degree= 2,include_bias=True)
x_poly = poly.fit_transform(X_train)
x_test_trans = poly.transform(X_test)
poly.fit(X_train,Y_train)

#running machine learning model
from sklearn.linear_model import LinearRegression
linear_regression_model = LinearRegression()
linear_regression_model.fit(x_poly,Y_train)
y_pred_test = linear_regression_model.predict(x_test_trans)

#generating output from model for set ages
prediction_array = np.array([[18],[24],[28],[32],[41]])
poly_array = poly.transform(prediction_array)
prediction= linear_regression_model.predict(poly_array)

ML_Result = pd.DataFrame(prediction)
Test_ages = [18,24,28,32,41]
ML_Result.insert(0,"Age",[18,24,28,32,41],True)
ML_Result.rename(columns={0: 'ML Predicted Minutes Played'},inplace=True)
ML_Result.head()
```

	Age	ML Predicted Minutes Played
0	18	264.081652
1	24	765.705950
2	28	908.291795
3	32	897.413356
4	41	311.833083

بعد إنشاء النموذج وتدريبه، أعطيت النموذج بعض قيم الإدخال لمعرفة عدد الدقائق التي يتوقع أن تلعبها هذه الفئات العمرية. أخذت هذه الأرقام لطباعة جدول للأعمار 18 و24 و28 و32 و41.

يظهر النموذج علاقة عكسية على شكل حرف "U" كمافي السابق. يمكننا أن نرى أنهفي سن 18 عامًا يكون هناك مقدار أقل من وقت اللعب الذي يرتفع ويستقر عند سن 28، وينخفض عند سن 41 حيث سيكون لدى أي لاعب متبقيفي هذا العمر ولم يتقاعد مقدارًا منخفضًا من وقت اللعب.

المصدر:

https://medium.com/@rahul-mohan-data-portfolio/data-analysis-on-football-players-using-python-machine-learning-faf32fe57cc0

5) التنبؤ بالإصابات في كرة القدم باستخدام التعلم الآلي
Injury Prediction in football using machine learning

ولمعالجة المخاوف المتزايدة بشأن سلامة اللاعبين والوقاية من الإصاباتفي الرياضات التنافسية، تم إنشاء هذه البيانات الاصطناعية synthetic data، والتي تلتقط سمات مهمة مثل التركيبة السكانية للاعبين player demographics، وكثافة التدريب training intensities، وأوقات التعافي recovery times، وتواريخ الإصابات السابقة previous injury histories. وقد تم تحديد الارتباطات بين هذه السمات واحتمالية الإصابات المستقبلية لمحاكاة السيناريوهاتفي العالم الحقيقي بدقة.

الهدف الرئيسي من هذا النوتبوك هو تطوير نموذج تصنيف يساهمفي الحفاظ على صحة الرياضي وحالته البدنية من خلال التنبؤ بإمكانية الإصابات بناءً على عوامل مختلفة مثل الطول والوزن وتاريخ الإصابات السابقة ووقت التعافي وكثافة التدريب وغيرها.

استيراد المكتبات والبيانات

```
import pandas as pd
import matplotlib
import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
import sklearn
from platform import python_version
from sklearn.preprocessing import OneHotEncoder
from sklearn.model_selection import train_test_split
from sklearn.ensemble import ExtraTreesClassifier, AdaBoostClassifier
from sklearn.svm import NuSVC
from sklearn.tree import ExtraTreeClassifier
from lightgbm import LGBMClassifier
from sklearn.metrics import classification_report, accuracy_score,
recall_score
from sklearn.metrics import confusion_matrix
from sklearn.metrics import roc_curve, roc_auc_score


# Libraries and Python version | Versão das bibliotecas e do Python
library = {
    "Pandas": pd,
    "Matplotlib": matplotlib,
    "Seaborn": sns,
    "NumPy": np,
    "Scikit-Learn": sklearn,
    }

# Libraries version
print("Library Version:\n")
print(f"{'':-^20} | {'':-^10}")
print(f"{'Library':^20} | {'Version':^10}")
print(f"{'':-^20} | {'':-^10}")
```

```
for nome, library in sorted(library.items()):
    print(f"{nome:<20} | {library.__version__:>10}")

# Python Version
print()
print(f"Python Version: {python_version()}")
```

```
                    Library Version:

-------------------- | ----------
      Library        |  Version
-------------------- | ----------
Matplotlib           |      3.7.5
NumPy                |     1.26.4
Pandas               |      2.2.0
Scikit-Learn         |      1.2.2
Seaborn              |     0.12.2

Python Version: 3.10.13
```

```
# Importing df | Importando df
df = pd.read_csv('/kaggle/input/injury-prediction-dataset/injury_data.csv')


df['Player_Weight'] = df['Player_Weight'].round(2)
df['Player_Height'] = df['Player_Height'].round(2)
df['Training_Intensity'] = df['Training_Intensity'].round(2)

# View df vg | Visualizar df vg
df.head(5)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury
0	24	66.25	175.73	1	0.46	5	0
1	37	71.00	174.58	0	0.23	6	1
2	32	80.09	186.33	0	0.61	2	1
3	28	87.47	175.50	1	0.25	4	1
4	25	84.66	190.18	0	0.58	1	1

التحقق من البيانات

لقد تمت معالجة كافة البيانات ويمكننا الانتقال إلى الخطوة التالية.

```
# Creating DataFrame with Dtype, Unique, and Null information | Criando Df
com informações Dtype, Unique e Null
df_info = pd.DataFrame(df.dtypes, columns=['Dtype'])
df_info['Unique'] = df.nunique().values
df_info['Null'] = df.isnull().sum().values
df_info
```

	Dtype	Unique	Null
Player_Age	int64	22	0
Player_Weight	float64	863	0
Player_Height	float64	875	0
Previous_Injuries	int64	2	0
Training_Intensity	float64	101	0
Recovery_Time	int64	6	0
Likelihood_of_Injury	int64	2	0

```
# Df Describe
with pd.option_context(
    "display.float_format",
    "{:.2f}".format,
    "display.max_columns",
    None,
):
    display(df.describe())
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury
count	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00	1000.00
mean	28.23	74.79	179.75	0.52	0.49	3.47	0.50
std	6.54	9.89	9.89	0.50	0.29	1.70	0.50
min	18.00	40.19	145.29	0.00	0.00	1.00	0.00
25%	22.00	67.95	173.03	0.00	0.24	2.00	0.00
50%	28.00	75.02	180.03	1.00	0.48	4.00	0.50
75%	34.00	81.30	186.56	1.00	0.73	5.00	1.00
max	39.00	104.65	207.31	1.00	1.00	6.00	1.00

إنشاء أعمدة جديدة

سيتم إنشاء عمودين جديدين: يحتوي العمود الأول على مؤشر كتلة الجسم Body Mass Index (BMI)، ويحتوي العمود الثاني على الفئات العمرية Age Categories.

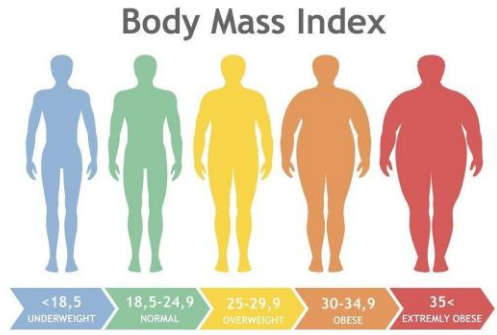
مؤشر كتلة الجسم

مؤشر كتلة الجسم (BMI) هو مقياس يستخدم لتقييم ما إذا كان الشخص يتمتع بوزن صحي نسبة إلى طوله. يتم حسابه عن طريق قسمة الوزن (بالكيلوجرام) على الطول مربعًا (بالأمتار).

في سيناريو واقعي، لن يكون مؤشر كتلة الجسم هو أفضل مؤشر لتقييم ما إذا كان الرياضيفي أفضل حالة بدنية، حيث لا يأخذ هذا المؤشرفي الاعتبار تكوين جسم الرياضي. قد يتم تقييم شخص يتمتع ببنية بدنية أكثر قوة، مع كتلة عضلية وكثافة عظام أكبر، على أنه يعاني من زيادة الوزن، على الرغم من أنه يتمتع بصحة جيدة ولديه خطر أقل للإصابة. من الناحية المثالية، سيكون لدينا بيانات حول تكوين جسم كل رياضي، بالإضافة إلى الرياضة التي يمارسها. نظرًا لعدم وجود هذه البيانات، فسيتم استخدام مؤشر كتلة الجسم لأنه مؤشر عام.

تظهر الصيغة الرياضية لحساب مؤشر كتلة الجسم وتصنيفاتهفي الصورة أدناه:

$$BMI = \frac{\text{Weight in kilogram}}{(\text{Height in meter})^2}$$




```
# Calculate the Body Mass Index (BMI) | Calcular o Índice de Massa Corporal
(IMC)
df['BMI'] = df['Player Weight'] / (df['Player Height'] / 100) ** 2

# Defining gaps for BMI classification | Definir os intervalos para
classificação do IMC
gaps = [-float('inf'), 18.5, 24.9, 29.9, 34.9, 39.9, float('inf')]
categories = ['Underweight', 'Normal', 'Overweight', 'Obesity I', 'Obesity
II', 'Obesity III']

# Create "BMI_Classification" column | Criar a coluna "Classificação_IMC"
df['BMI_Classification'] = pd.cut(df['BMI'], bins=gaps, labels=categories,
right=False)

df.head(1)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury	BMI
0	24	66.25	175.73	1	0.46	5	0	21.453298

الفئة العمرية

لقد اخترت تقسيم اللاعبين البالغ عددهم 1000 لاعبفي هذه المجموعة من البيانات إلى 5 مجموعات عمرية، بهدف تصور هذه المجموعات واختبار إضافة هذه الأعمدة لتقييم أداء النماذج.

```
# Finding the youngest and oldest age among athletes | Descobrindo a idade
mais nova e mais velha entre os atletas
print('Player Age Min: {}'.format(df.Player Age.min()))
print('Player Age Max: {}'.format(df.Player_Age.max()))
```

```
Player Age Min: 18
Player Age Max: 39
```

```
# Creating columns with grouping | Criando colunas com agrupamento
df["Age_Group"] = pd.cut(
    df["Player_Age"],
    bins=[18, 22, 26, 30, 34, df["Player_Age"].max()],
    labels=["18-22", "23-26", "27-30", "31-34", "35+"],
    include_lowest=True,
)
```

```
df.head(5)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury	BMI
0	24	66.25	175.73	1	0.46	5	0	21.453298
1	37	71.00	174.58	0	0.23	6	1	23.295357
2	32	80.09	186.33	0	0.61	2	1	23.068148
3	28	87.47	175.50	1	0.25	4	1	28.399120
4	25	84.66	190.18	0	0.58	1	1	23.407152

تحليل البيانات الاستكشافي

نظرًا لأن إطار البيانات هذا يحتوي على عدد قليل من الأعمدة، فلنقم بتحليلها جميعًا.

```
def plot_histogram_kde_and_boxplot(dataframe, column, color_column):
    fig, axs = plt.subplots(1, 3, figsize=(18, 6))

    # Remove grid and spines | Remover o grid e as bordas
    for ax in axs:
        ax.grid(False)
        for spine in ax.spines.values():
            spine.set_visible(False)

    # Plot histogram (subplot 1) | Plotar histograma (subplot 1)
    sns.histplot(data=dataframe, x=column, bins=20, color='skyblue',
edgecolor='black', kde=True, ax=axs[0])

    # Add labels | Adicionar rótulos
    axs[0].set_xlabel('')
    axs[0].set_ylabel('')
    axs[0].set_title(f'{column} Histogram', weight='bold', size=13)

    # Plot KDE (subplot 2) | Plotar KDE (subplot 2)
    sns.kdeplot(data=dataframe, x=column, color='skyblue', fill=True,
hue=color_column, palette={0: 'green', 1: 'red'}, ax=axs[1])
    axs[1].set_xlabel('')
    axs[1].set_ylabel('')
    axs[1].set_title(f'{column} Density', weight='bold', size=13)

    # Plot boxplot (subplot 3) | Plotar boxplot (subplot 3)
    sns.boxplot(data=dataframe[column], orient='h', ax=axs[2])

    # Add labels | Adicionar rótulos
    axs[2].set_xlabel('')
    axs[2].set_ylabel('')
    axs[2].set_title(f'{column} Boxplot', weight='bold', size=13)

    # Adjust layout | Ajustar o layout
    plt.tight_layout()

    # Display figure | Exibir a figura
    plt.show()


def plot_dual_chart(dataframe, column1, column2, cat_order=None,
y_limit1=None, y_limit2=None):
    fig, axs = plt.subplots(1, 2, figsize=(18, 6))

    # Remove grid and spines | Remover o grid e as bordas
    for ax in axs:
```

```
        ax.grid(False)
        for spine in ax.spines.values():
            spine.set_visible(False)

    # Plot histogram | Plotar histograma
    sns.histplot(data=dataframe, x=column1, bins=20, color='skyblue',
edgecolor='black', kde=True, ax=axs[0])
    axs[0].set_title(f'{column1} Histogram', weight='bold', size=13)
    axs[0].set_xlabel('')
    axs[0].set_ylabel('')

    # Define y limit | Definir limite y
    if y_limit1 is None:
        y_limit1 = dataframe[column1].max() * 1.1
    axs[0].set_ylim(top=y_limit1)

    # Plot two sets of bars | Plotar os dois conjuntos de barras
    ax = sns.countplot(data=dataframe, x=column2,
hue='Likelihood_of_Injury', palette={0: 'green', 1: 'red'}, ax=axs[1],
linewidth=2, order=cat_order)

    # Add labels | Adicionar rótulos
    axs[1].set_xlabel('')
    axs[1].set_ylabel('')
    axs[1].set_title(f'{column2} x Likelihood_of_Injury', weight='bold',
size=13)

    # Rotate x-axis labels | Rotacionar os rótulos do eixo x
    axs[1].tick_params(axis='x', rotation=0)

    # Remove background grid | Remover a grade de fundo
    axs[1].grid(False)

    # Add legend | Adicionar legenda
    axs[1].legend()

    # Define upper limit | Definir limite superior
    if y_limit2 is None:
        y_limit2 = dataframe[column2].value_counts().max() * 1.1  # Max
value multiplied by 1.1 to ensure a margin
    axs[1].set_ylim(top=y_limit2)

    # Add values on top of each bar | Adicionar valores em cima de cada
barra
    for p in axs[1].patches:
        height = p.get_height()
        if not np.isnan(height):
            axs[1].annotate(str(int(height)), (p.get_x() + p.get_width() /
2., height),
                            ha='center', va='center', xytext=(0, 5),
textcoords='offset points', color='black', weight='bold', size=13)
        else:
            axs[1].annotate("0", (p.get_x() + p.get_width() / 2., 0),
                            ha='center', va='center', xytext=(0, 5),
textcoords='offset points', color='black', weight='bold', size=13)

    # Adjust layout | Ajustar layout
    plt.tight_layout()

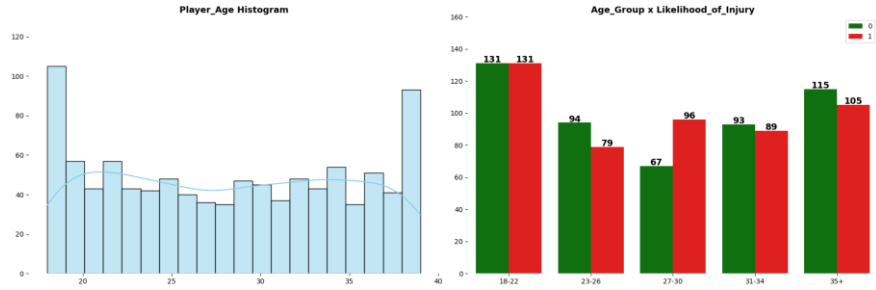
    # Display figure | Exibir a figura
```

```
    plt.show()
```

Age_Group

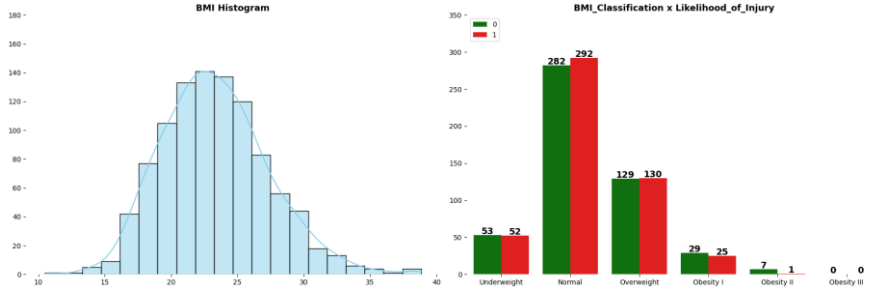
يمكننا أن نلاحظ زيادةفي نسبة اللاعبين الذين من المحتمل تعرضهم للإصابةفي الفئة العمرية من 27 إلى 30 عامًا، مقارنة بالفئات الأخرى.

```
plot dual chart(df, 'Player Age', 'Age Group', cat order=["18-22", "23-26",
"27-30", "31-34", "35+"], y_limit1=130, y_limit2=160)
```



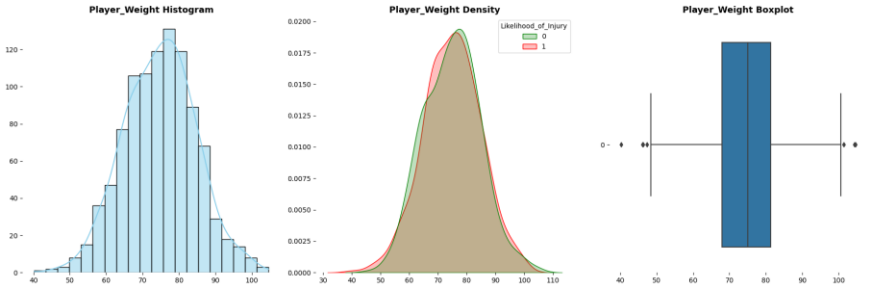
BMI_Classification

```
plot_dual_chart(df, 'BMI', 'BMI_Classification', y_limit1=180,
y_limit2=350)
```



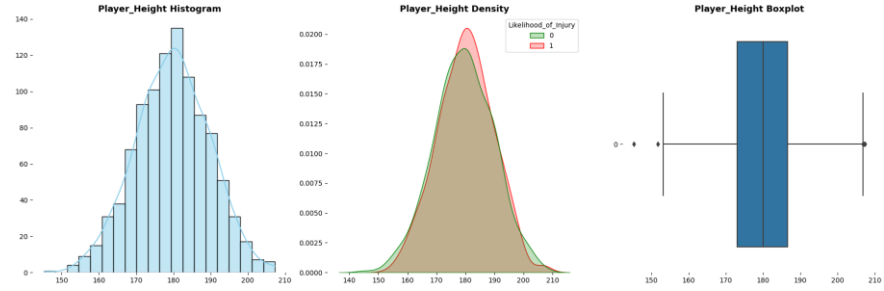
Player_Weight

```
plot_histogram_kde_and_boxplot(df, 'Player_Weight', 'Likelihood_of_Injury')
```



Player_Height

```
plot_histogram_kde_and_boxplot(df, 'Player_Height', 'Likelihood_of_Injury')
```



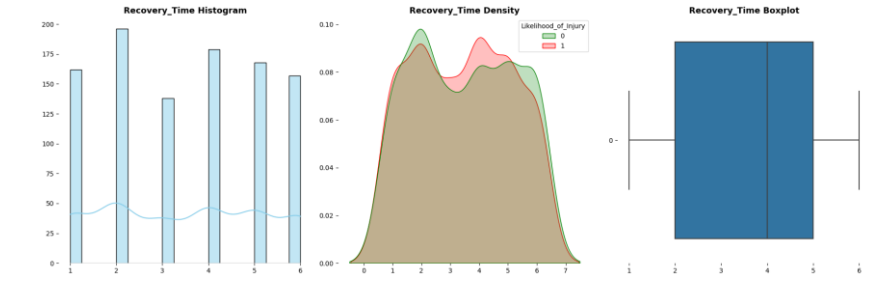
Training_Intensity

يمكننا ملاحظة أن أنماط احتمالات الإصابة تنعكسفي مخطط الكثافة. ففي جلسات التدريب ذات الكثافة المنخفضة، يفوق عدد اللاعبين الذين لا يتعرضون لخطر الإصابة عدد اللاعبين المعرضين للخطر، بينمافي جلسات التدريب ذات الكثافة الأعلى، يفوق عدد اللاعبين المعرضين لخطر الإصابة عدد اللاعبين غير المعرضين لخطر الإصابة.

```
plot_histogram_kde_and_boxplot(df, 'Training_Intensity',
'Likelihood_of_Injury')
```

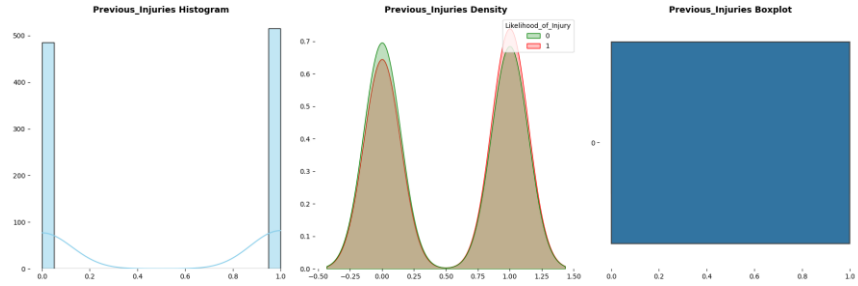
Recovery_Time

```
plot_histogram_kde_and_boxplot(df, 'Recovery_Time', 'Likelihood_of_Injury')
```



Previous_Injuries

```
plot_histogram_kde_and_boxplot(df, 'Previous_Injuries',
'Likelihood_of_Injury')
```

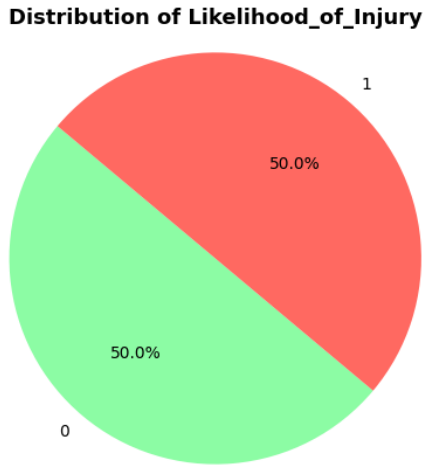


Likelihood_of_Injury

يعرض عمود الهدف الخاص بنا، Likelihood_of_Injury، توزيعًا متوازنًا تمامًا للبيانات. وهذا يعني أنه لا توجد حاجة لتعديل كمية البيانات أثناء عملية التدريب لضمان توزيع مناسب بين الفئتين 1 و0.

```
# Count 'Likelihood_of_Injury' | Contagem 'Likelihood_of_Injury'
li_count = df['Likelihood_of_Injury'].value_counts()

# Plot pie chart | Plotar o gráfico de pizza
plt.figure(figsize=(5, 5))
plt.pie(li_count, labels=li_count.index, autopct='%1.1f%%', startangle=140,
colors=['#8CFCA4', '#FF6961'])
plt.title('Distribution of Likelihood_of_Injury', weight='bold', size=13)
plt.axis('equal')
plt.show()
```



معالجة البيانات مسبقًا

من خلال OneHotEncoder، سنقوم بتحويل المتغيرات الفئوية إلى تمثيلات رقمية ثنائية.

```
# Categorical columns
one_hot_cols = [
    "BMI_Classification",
    "Age_Group",
]

# Selecting only categorical columns from the DataFrame | Selecionando
apenas as colunas categóricas do DataFrame
df_categorical = df[one_hot_cols]

# Applying OneHotEncoder | Aplicando o OneHotEncoder
encoder = OneHotEncoder()
encoded_data = encoder.fit_transform(df_categorical)

# Obtaining names of the features generated by OneHotEncoder | Obtendo os
nomes das features geradas pelo OneHotEncoder
one_hot_feature_names = encoder.get_feature_names_out(one_hot_cols)

# Creating a DataFrame with transformed features | Criando um DataFrame com
as features transformadas
df_encoded = pd.DataFrame(encoded_data.toarray(),
columns=one_hot_feature_names)

# Joining DataFrames | Juntar os DataFrames
df_final = pd.concat([df, df_encoded], axis=1)

# Dropping categorical columns | Excluindo colunas categóricas
df_final.drop(columns=['BMI_Classification', 'Age_Group'], inplace=True)

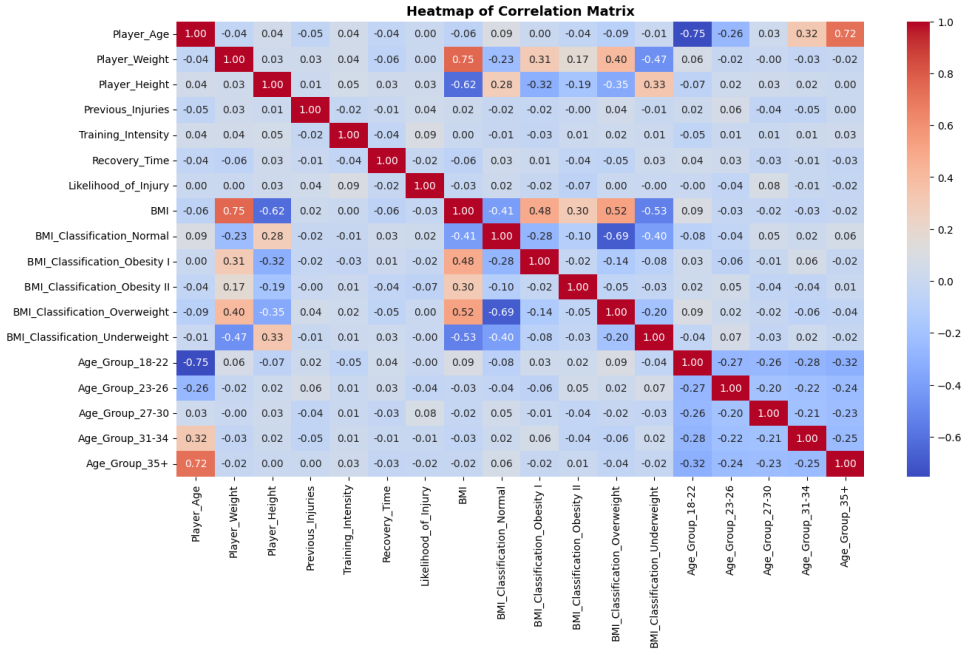
# Visualizing the first few rows of the final DataFrame | Visualizar as
primeiras linhas do DataFrame final
df_final.head()
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury	BMI
0	24	66.25	175.73	1	0.46	5	0	21.453298
1	37	71.00	174.58	0	0.23	6	1	23.295357
2	32	80.09	186.33	0	0.61	2	1	23.068148
3	28	87.47	175.50	1	0.25	4	1	28.399120
4	25	84.66	190.18	0	0.58	1	1	23.407152

الارتباط بين الأعمدة

```
# Calculating correlation matrix | Calculando a matriz de correlação
correlation_matrix = df_final.corr()

# Plotting heatmap | Plotando o heatmap
plt.figure(figsize=(15, 8))
sns.heatmap(correlation_matrix, annot=True, cmap='coolwarm', fmt=".2f")
plt.title('Heatmap of Correlation Matrix', weight='bold', size=13)
plt.show()
```



الارتباط بـ "Likelihood_of_Injury"

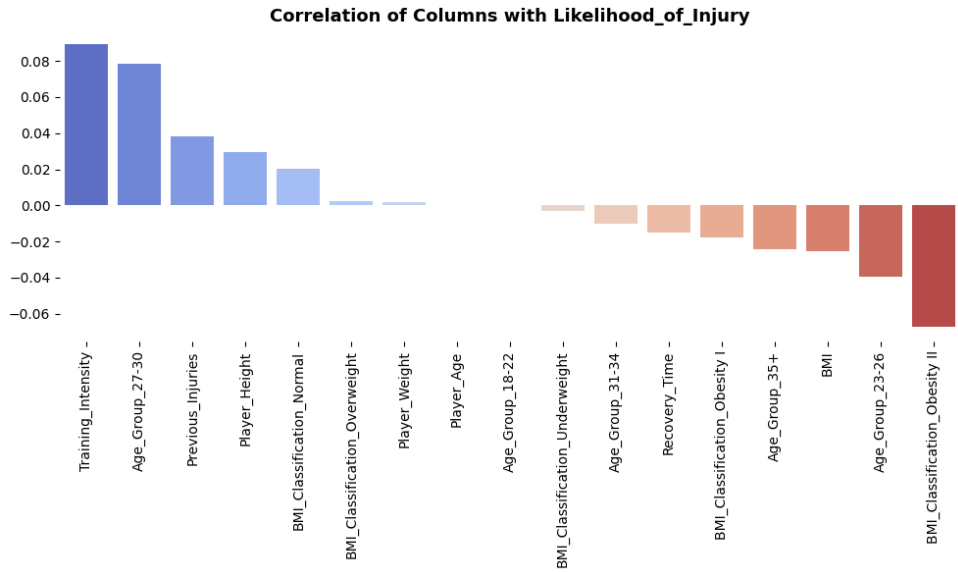
```
# Calculating correlation matrix | Calcular a matriz de correlação
correlation matrix = df final.corr()

# Selecting only 'Likelihood_of_Injury' column from the correlation matrix
| Selecionar apenas a coluna 'Likelihood_of_Injury' da matriz de correlação
correlation_with_likelihood = correlation_matrix['Likelihood_of_Injury']

# Removing the correlation with the 'Likelihood_of_Injury' column | Remover
a correlação com a coluna 'Likelihood_of_Injury'
correlation_with_likelihood =
correlation_with_likelihood.drop('Likelihood_of_Injury')

# Sorting correlations in descending order | Ordenar as correlações em
ordem decrescente
correlation_with_likelihood =
correlation_with_likelihood.sort_values(ascending=False)

# Plotting correlation bar plot | Plotar o gráfico de barras de correlação
plt.figure(figsize=(10, 6))
sns.barplot(x=correlation_with_likelihood.index,
y=correlation_with_likelihood.values, palette='coolwarm')
plt.xticks(rotation=90, ha='center')
plt.xlabel('')
plt.ylabel('')
plt.box(False)
plt.title('Correlation of Columns with Likelihood_of_Injury',
weight='bold', size=13)
plt.tight_layout()
plt.show()
```



بطبيعة الحال، هناك ارتباط بين عمودي "BMI" و"Player_Age" ومجموعاتهما الخاصة. لذلك، اعتمدت النهج التالي:

- استبعدت عمود "BMI"، واحتفظت فقط بالأعمدة المشتقة من مجموعاته ("BMIClassification").
- خلال اختباراتي، لاحظت أن النموذج حقق أداءً أفضل مع عمود "Player_Age" مقارنة بأعمدة مجموعات الأعمار ("Age_Group").

```
# Drop columns starting with "Age_Group" | Excluir colunas que começa com
"Age_Group"
df_final = df_final.loc[:, ~df_final.columns.str.startswith('Age Group')]

# Drop BMI column | Excluir a coluna BMI
df_final = df_final.drop(columns=['BMI'])

df_final.head(1)
```

	Player_Age	Player_Weight	Player_Height	Previous_Injuries	Training_Intensity	Recovery_Time	Likelihood_of_Injury	BMI_Class
0	24	66.25	175.73	1	0.46	5	0	1.0

تدريب النماذج

```
# Features
X = df_final.drop('Likelihood_of_Injury', axis=1)

# Target variable | Variável alvo
y = df_final['Likelihood_of_Injury']

# Split data into training and testing sets | Dividir os dados em conjuntos
de treinamento e teste
```

```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.1,
random_state=42)
```

```
from sklearn.metrics import classification_report, accuracy_score,
precision_score, recall_score

# Dictionary models | Dicionário de modelos
models = {
    "LGBMClassifier": LGBMClassifier(),
    "AdaBoostClassifier": AdaBoostClassifier(),
    "ExtraTreesClassifier": ExtraTreesClassifier(),
    "NuSVC": NuSVC(probability=True),
    "ExtraTreeClassifier": ExtraTreeClassifier(),
    }

for model_name, model in models.items():
    model.fit(X_train, y_train)
    predictions = model.predict(X_test)
    recall = recall_score(y_test, predictions)
    accuracy = accuracy_score(y_test, predictions)
    precision = precision_score(y_test, predictions)

    print(f"Model: {model name}")
    print(f"Recall: {recall}")
    print(f"Accuracy: {accuracy}")
    print(f"Precision: {precision}")
    print("-" * 50)
```

```
[LightGBM] [Warning] Found whitespace in feature_names, replace with underl
ines
[LightGBM] [Info] Number of positive: 441, number of negative: 459
[LightGBM] [Info] Auto-choosing row-wise multi-threading, the overhead of t
esting was 0.002991 seconds.
You can set `force_row_wise=true` to remove the overhead.
And if memory is not enough, you can set `force_col_wise=true`.
[LightGBM] [Info] Total Bins 651
[LightGBM] [Info] Number of data points in the train set: 900, number of us
ed features: 10
[LightGBM] [Info] [binary:BoostFromScore]: pavg=0.490000 -> initscore=-0.04
0005
[LightGBM] [Info] Start training from score -0.040005
Model: LGBMClassifier
Recall: 0.5423728813559322
Accuracy: 0.61
Precision: 0.7272727272727273
--------------------------------------------------
Model: AdaBoostClassifier
Recall: 0.576271186440678
Accuracy: 0.6
Precision: 0.6938775510204082
--------------------------------------------------
Model: ExtraTreesClassifier
Recall: 0.4915254237288136
Accuracy: 0.56
```

```
Precision: 0.6744186046511628
--------------------------------------------------
Model: NuSVC
Recall: 0.8135593220338984
Accuracy: 0.59
Precision: 0.6153846153846154
--------------------------------------------------
Model: ExtraTreeClassifier
Recall: 0.5254237288135594
Accuracy: 0.52
Precision: 0.6078431372549019
--------------------------------------------------
```

اختيار النموذج

في سيناريو واقعي حيث يمكن أن تساعد نتيجة هذا النموذج طاقم التدريبفي اتخاذ قرار بشأن إشراك لاعب أم لا بناءً على احتمال تعرضه لإصابة أثناء المباراة، سأختار النموذج الذي يحتوي على أقل عدد من النتائج السلبية الكاذبة. بهذه الطريقة، سيتم الحفاظ على صحة اللاعب وحالته البدنية من خلال تقليل احتمالية مشاركتهفي مباراة تم تحديدها بشكل خاطئ بواسطة النموذج على أنها لا تشكل خطرًا للإصابة.

بالنظر إلى هذا الموقف، سأختار استخدام درجة الاستدعاء recall score لاختيار أفضل نموذج، حيث يتم معاقبة هذا المقياس كلما تم التنبؤ بنتيجة سلبية كاذبة.

أخيرًا، سأرسم مصفوفات الارتباك confusion matrices ومنحنى ROC لتعزيز اختيار النموذج الذي يقلل بشكل أفضل من خطر النتائج السلبية الكاذبة.

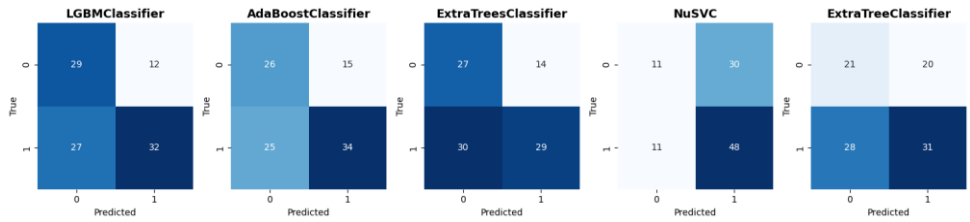
مصفوفة الارتباك

يمكننا أن نلاحظ أن النموذج الذي يحتوي على أقل عدد من النتائج السلبية الكاذبة هو NuSVC، مع 11 حالة فقط. ومن الواضح أنه الخيار الأفضل بالنظر إلى هذا الموقف المحدد.

```
# Create figure and axes
fig, axes = plt.subplots(1, len(models), figsize=(15, 3.5))

# Plot confusion matrix for each model
for ax, (model_name, model) in zip(axes, models.items()):
    predictions = model.predict(X_test)
    cm = confusion_matrix(y_test, predictions)
    sns.heatmap(cm, annot=True, fmt="d", cmap="Blues", cbar=False, ax=ax)
    ax.set_title(f"{model_name}", weight='bold', size=13)
    ax.set_xlabel("Predicted")
    ax.set_ylabel("True")

# Adjust layout and show figure
plt.tight_layout()
plt.show()
```



منحنى ROC المعكوس

يعد منحنى ROC المعكوس (inverted ROC) مفيداً عندما يركز التحليل على الفئة السلبية بدلاً من الفئة الإيجابية. فهو يوفر نظرة ثاقبة لقدرة النموذج على تجنب التصنيف الخاطئ للحالات السلبية.

```
# Plot Inverted ROC Curve
plt.figure(figsize=(8, 6))
for model_name, model in models.items():
    y_proba = model.predict_proba(X_test)[:, 0]  # Probabilities of
belonging to the negative class
    fpr, tpr, _ = roc_curve(y_test, y_proba)

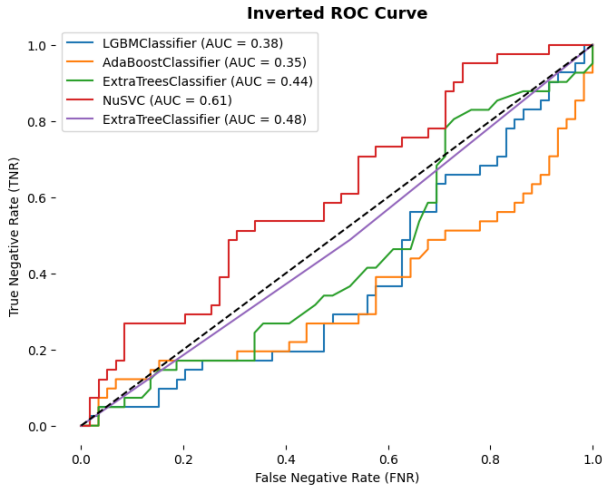
    tnr = 1 - fpr  # True Negative Rate (TNR)
    fnr = 1 - tpr  # False Negative Rate (FNR)

    auc = roc_auc_score(y_test, y_proba)
    plt.plot(fnr, tnr, label=f'{model_name} (AUC = {auc:.2f})')

# Plot the diagonal line
plt.plot([0, 1], [0, 1], linestyle='--', color='black')

plt.xlabel('False Negative Rate (FNR)')
plt.ylabel('True Negative Rate (TNR)')
plt.title('Inverted ROC Curve', weight='bold', size=13)
plt.legend()

plt.box(False)  # Remove plot borders
plt.show()
```



بناءً على قيمة الاستدعاء Recall التي تم الحصول عليها، بالإضافة إلى تصور مصفوفة الارتباك ومنحنى ROC، فإن النموذج المختار هو NuSVC.

المصدر:

https://www.kaggle.com/code/tkunzler/injury-prediction-eda-eng-pt-br

6) تحليل بيانات كرة القدم باستخدام بايثون Football Data Analysis using python

كرة القدم Football، المعروفة أيضًا باسم soccerفي أجزاء أخرى من العالم، هي واحدة من أكثر الرياضات شعبية ومتابعة على نطاق واسع على مستوى العالم.في السنوات الأخيرة، أدى توفر كميات هائلة من بيانات كرة القدم إلى فتح إمكانيات مثيرة للرؤى والتحليلات القائمة على البيانات. تعد لغة بايثون، باعتبارها لغة برمجة متعددة الاستخدامات وقوية، خيارًا ممتازًا للمبتدئين والمحترفين على حد سواء لإجراء تحليل بيانات كرة القدم.في هذه المقالة، سوف نستكشف كيفية استخدام سكريبتات بايثون Python scripts الأساسية لتحليل بيانات كرة القدم، ومناقشة التحديات والقيود، وتقديم نصوص أكواد نموذجية لفهم أفضل.

قبل أن نتعمقفي تعقيدات تحليل بيانات كرة القدم، تأكد من تثبيت بايثون على نظامك، إلى جانب المكتبات ذات الصلة مثل Pandas وNumPy وMatplotlib. تشكل هذه المكتبات العمود الفقري لمعالجة البيانات والعمليات العددية والتصور، وهي ضرورية لهذه المهمة.

جمع البيانات ومعالجتها مسبقًا

الخطوة الأولىفي أي مشروع لتحليل البيانات هي جمع البيانات. هناك مصادر مختلفة للحصول على بيانات كرة القدم، مثل واجهات برمجة التطبيقات APIs، أو قواعد البيانات عبر الإنترنت online databases، أو حتى مجموعات البيانات التي تم تنظيمها يدويًا. بمجرد الحصول على البيانات، تكون المعالجة المسبقة أمرًا بالغ الأهمية لتنظيفها وتنظيمها للتحليل.

مثال على الكود:

```
import pandas as pd

# Load data from CSV file
df = pd.read_csv('football_data.csv')

# Data cleaning and preprocessing
# ... (e.g., handling missing values, data type conversion, etc.)
```

استكشاف البيانات وتصورها

بعد معالجة البيانات مسبقًا، تكون الخطوة التالية هي استكشافها لاكتساب رؤى وتحديد الأنماط. يلعب التصور دورًا مهمًافي فهم البيانات وجعلها أكثر قابلية للتفسير.

مثال على الكود:

```
import matplotlib.pyplot as plt

# Basic data exploration
```

```
print(df.head())

# Plotting goals scored by each team
team_goals = df.groupby('Team')['Goals'].sum()
team_goals.plot(kind='bar', figsize=(10, 6))
plt.title('Total Goals Scored by Each Team')
plt.xlabel('Team')
plt.ylabel('Goals')
plt.show()
```

مقاييس الأداء

لتقييم فرق كرة القدم واللاعبين، ستحتاج إلى مقاييس الأداء performance metrics. تعد المقاييس مثل معدل تحويل الأهداف goal conversion rate ومعدل إتمام التمريرات pass completion rate ودقة التسديد shot accuracy أمرًا بالغ الأهمية لتحليل الأداء القائم على البيانات.

مثال على الكود:

```
# Calculating goal conversion rate
df['Goal Conversion Rate'] = (df['Goals'] / df['Shots']) * 100

# Visualizing goal conversion rate
plt.scatter(df['Goal Conversion Rate'], df['Player'], alpha=0.5)
plt.title('Goal Conversion Rate vs. Player')
plt.xlabel('Goal Conversion Rate (%)')
plt.ylabel('Player')
plt.show()
```

تحليل الاتجاهات

يمكن أن يوفر تحليل الاتجاهات Analyzing trends بمرور الوقت رؤى قيمة حول أداء الفريق عبر المواسم أو المسابقات المختلفة.

مثال على الكود:

```
# Converting 'Date' column to datetime format
df['Date'] = pd.to_datetime(df['Date'])

# Analyzing goals scored over time
goals_over_time = df.groupby('Date')['Goals'].sum()
goals_over_time.plot(figsize=(12, 6))
plt.title('Goals Scored Over Time')
plt.xlabel('Date')
plt.ylabel('Goals')
plt.show()
```

التحديات والقيود

على الرغم من أن سكريبتات بايثون تقدم أدوات قوية لتحليل بيانات كرة القدم، إلا أن هناك بعض التحديات والقيود التي يجب مراعاتها:

1. **جودة البيانات Data Quality:** يمكن أن تؤثر دقة البيانات واكتمالها بشكل كبير على التحليل. قد تؤدي البيانات غير المكتملة أو غير الموثوقة إلى استنتاجات مضللة.

2. **تكامل البيانات Data Integration:** غالبًا ما تأتي بيانات كرة القدم من مصادر وتنسيقات مختلفة. قد يكون دمج البيانات من أصول مختلفة أمرًا مستهلكًا للوقت وصعبًا.
3. **التحليل المتقدم Advanced Analysis:** نصوص بايثون الأساسية محدودةفي قدرتها على إجراء تحليل إحصائي معقد وتحليل التعلم الآلي. قد تتطلب التقنيات الأكثر تطورًا مكتبات وخبرات متقدمة.
4. **البيانات في الوقت الفعلي Real-time Data:** يتطلب تحليل البياناتفي الوقت الفعلي أثناء المباراة أنظمة جمع بيانات قوية وخوارزميات فعالة، والتي قد تكون خارج نطاق النصوص الأساسية.
5. **الضبط الزائد Overfitting:** في النمذجة التنبؤية predictive modeling، يمكن أن يؤدي الضبط الزائد للبيانات التاريخية إلى تنبؤات غير دقيقة للمباريات أو المواسم المستقبلية.

الاستنتاج

تعد بايثون خيارًا ممتازًا للمبتدئين والمحترفين الذين يتطلعون إلى إجراء تحليل بيانات كرة القدم. باستخدام مكتبات بايثون، يمكنك جمع بيانات كرة القدم ومعالجتها مسبقًا واستكشافها وتصورها بشكل فعال. من خلال إنشاء سكريبتات بايثون الأساسية، يمكنك اكتساب رؤى قيمة حول أداء اللاعب والفريق وتحديد الاتجاهات واستخلاص استنتاجات مدفوعة بالبيانات. ومع ذلك، من الضروري الاعتراف بالتحديات والقيود والوعي بالتقنيات الأكثر تقدمًا مع تقدمكفي رحلة تحليل بيانات كرة القدم. استمتع بالبرمجة واستكشاف عالم تحليلات كرة القدم المثير باستخدام بايثون!

المصدر:

https://medium.com/@TacticsFC/analyzing-football-data-with-python-7b4e89c7abd8

7) التنبؤ بنتائج كأس العالم باستخدام بايثون Predicting the outcome of the World Cup using Python

يبدو أن التعلم الآلي Machine learning والتعلم العميق Deep Learning موجودانفي كل مكان هذه الأيام. يمكن للذكاء الاصطناعي AI أن يفعل أي شيء! من كتابة المقالات إلى الاختبارات إلى إنشاء الصور إلى وجود آلة لتوليد الصور لا نهائية تعمل على جهازك المحلي، الاحتمالات لا حصر لها. لذا فلا ينبغي أن يكون مفاجئًا أن تتمكن من استخدام التعلم العميق للتنبؤ بنتائج البطولات الرياضية الكبرى.

أعني، لا ينبغي أن يكون الأمر مفاجئًا وأنا متأكد من أنه يمكنك ذلك، ولكن اليوم سنتحدث عن كيفية التنبؤ بكأس العالمفي قطر Qatar World Cup دون أي أدوات تعلُّم آلي،في 150 سطرًا فقط من بايثون. سأستخدم Neptyne، جدول البيانات القابل للبرمجة programmable spreadsheet– جزئيًا لأنه الشركة الناشئة التي شاركتفي تأسيسها، ولكن أيضًا لأنه منصة جيدة لهذا النوع من الأشياء. ومع ذلك، يجب أن تعمل المبادئ على أي منصة بايثون.

خطتنا الأساسية هنا هي الحصول على المباريات التي لعبت مؤخرًا، وحساب نقاط القوة النسبية لكل دولة ثم محاكاة البطولة ألف مرة حتى يكون لدينا فكرة عن النتائج المحتملة.

بيانات التدريب

إن التنبؤات صعبة خاصة فيما يتعلق بالمستقبل، ولكن عليك أن تبدأ بالماضي. يحتوي موقع Kaggle على مجموعة بيانات جيدة، لذا فلنقم بتنزيلها. تحتوي هذه المجموعة على جميع المباريات الدولية منذ عام 1872، لذا فهي كثيرة بعض الشيء. نريد فقط أن نذكر المباريات الأخيرة والمباريات التي خاضتها الدول التي لعبت أكثر من عشر مباريات:

```
def upload_results():
    """Call this function and upload the data from kaggle"""
    tbl = nt.upload("Kaggle International football results",
"*.csv").dropna()
    cutoff_date = sorted(tbl["date"])[-2500]
    tbl = tbl[tbl["date"] >= cutoff_date]
    counts = Counter(
        itertools.chain.from_iterable(
            ((rec['home_team'], rec['away_team']) for _, rec in
tbl.iterrows())
        )
    )
    thress_hold = {country for country, count in counts.items() if count <=
10}
    res = [
        (DATEVALUE(rec['date']), rec['home_team'],
                   rec['away_team'], rec['home_score'], rec['away_score'])
        for _, rec in tbl.iterrows()
        if not rec['home_team'] in thress_hold and not rec['away_team'] in
```

```
thress_hold
    ]
    res.insert(0, ["Date", "Home team", "Away team", "Goals for", "Goals
against"])
    if "History" in nt.sheets:
        nt.sheets.delete_sheet("History")
    nt.sheets.new_sheet("History")
    History!B1 = res
```

يأخذ هذا الكود ملف csv الذي حصلنا عليه، ويحتفظ بأخر 2500 مباراة، ويستبعد الفرق التي لم تلعب بشكل كافٍ ويضع كل ذلكفي ورقة جديدةفي جدول البيانات الخاص بنا تسمى "التاريخ History"، ثم يحتفظ فقط بالأعمدة التي نهتم بها.

مآثر القوة

بعد ذلك نريد استخدام هذه البيانات التاريخية لحساب القوة النسبية لكل فريق. وهنا يأتي دور التعلم العميق، ولكنه لا يفعل ذلك. فبدلاً من ذلك، نقوم بالأمر المباشر. نحاول تعيين درجة لكل فريق بحيث إذا لعب فريقان ضد بعضهما البعض، فإن الفارقفي النتيجة سيتوافق مع فارق الأهداف المتوقع بين الفريقين. نبدأ بتعيين درجة صفر لكل فريق ثم نعدل هذه الدرجات لتتناسب مع جميع المباريات. نكرر العملية حوالي ألف مرة ونحصل على تقييمات معقولة. إليك الكود:

```
def train_model():
    scores = defaultdict(float)
    min_goals = []
    home_advantage = 0
    for i in range(500):
        delta = 5 / (i *10 + 50)
        total_diff = []
        ddd = 0
        for country1, country2, goals1, goals2 in History!C2:F:
            if i == 0:
                min_goals.append(min(goals1, goals2))
            diff = goals1 - goals2 - (scores[country1] - scores[country2])
- home_advantage
            total_diff.append(diff)
            ddd += abs(diff)
            scores[country1] = scores[country1] + diff * delta
            scores[country2] = scores[country2] - diff * delta
            home_advantage += delta * diff / 20
        Model!F2:G[i // 10] = [[i + 1, ddd / len(History!C2:F)]]
        Model!C2 = sorted(scores.items(), key=lambda rec: rec[1],
reverse=True)
    Model!A2 = "Score Std:", "Av Min Goals:", "Min Goals Std:", "Home
advantage"
    Model!B2 = np.std(total_diff), sum(min_goals) / len(min_goals),
np.std(min_goals), home_advantage
```

نقرأ التاريخ من الجدول (table) الذي أنشأناه للتو ونكتب النتائجفي جدول آخر يسمى النموذج Model. كما نتابع الانحراف المعياري standard deviation للنتائج والحد الأدنى لعدد الأهداف، بالإضافة إلى الحد الأدنى الفعلي لعدد الأهداف. إن متابعة ميزة اللعب على أرض الفريق المنافس يجعل النموذج أكثر استقرارًا. وفيما يلي النتائج:

تظل البرازيل دائمًا المرشحة المفضلة — وكذلك ألمانيا التي تحتل المرتبة الأخيرةفي نظرتنا الشاملة للفرق التي حققت نتيجة جيدة.

		Country	Score
Score Std:	1.30087	Brazil	3.1916458875750457
Av Min Goals:	0.4981	Argentina	3.041523978342597
Min Goals Std:	0.66887	Portugal	2.765525467376127
Home advantage	0.279199325067735	Spain	2.6947680612516947
		England	2.693005636652647
		Belgium	2.6744007404444314
Train	1.53979	Denmark	2.5853353196917612
		Italy	2.545984537358484
		France	2.426836830746995
		Netherlands	2.371138514007821
		Germany	2.3684043887804362

التنبؤ بالمباريات

يمكننا الآن محاكاة المباريات الفردية باستخدام بعض التعليمات البرمجية البسيطة:

```
def predict(team1, team2):
    score1 = VLOOKUP(team1, Model!C2:D200, 2, False)
    score2 = VLOOKUP(team2, Model!C2:D200, 2, False)
    gf = np.random.normal(score1 - score2, Model!B2)
    ga = 0
    if gf < 0:
        ga = -gf
        gf = 0
    mg = max(0, np.random.normal(Model!B3, Model!B4))
    gf = max(int(round(mg + gf)), 0)
    ga = max(int(round(mg + ga)), 0)
    return gf, ga
```

يعتمد هذا الأمر على جدول البيانات أكثر من الكود الآخر، ولكن يجب أن يكون من السهل متابعته. نبحث عن نتائج كل فريق ثم نختار قيمة عشوائية لفارق الأهداف، موزعة بشكل طبيعي حول فارق الأهداف المتوقع. نضيف الحد الأدنى لعدد الأهداف مرة أخرى موزعة بشكل طبيعي ونتأكد من عدم إرجاع قيم أقل من الصفر.

محاكاة البطولة

إن محاكاة البطولة بأكملها الآن عبارة عن تشغيل هذه العملية على جميع المباريات.في جدول بيانات Neptyne، يمكننا بسهولة نمذجة هيكل البطولة باستخدام بعض الدوال المخصصة لنمذجة نتائج المجموعات على سبيل المثال:

```
def standings(games):
    all_teams = {}
    for team1, _flag, team2, result in games:
        all_teams[team1] = 8 * [0]
        all_teams[team2] = 8 * [0]
    for team1, _flag, team2, result in games:
        if result:
```

```
            goals1, goals2 = split_score(result)
            for team, gf, ga in zip((team1, team2), (goals1, goals2),
(goals2, goals1)):
                entry = all_teams[team]
                entry[0] += 1
                entry[1] += 1 if gf > ga else 0
                entry[2] += 1 if gf == ga else 0
                entry[3] += 1 if gf < ga else 0
                entry[4] += gf
                entry[5] += ga
                entry[6] += gf - ga
                entry[7] += (3 if gf > ga else (1 if gf == ga else 0))
    return sorted(([k, *v] for k, v in all_teams.items()), key=lambda
r:r[8], reverse=True)
```

يؤدي هذا إلى إرجاع نتائج المجموعة بناءً على قائمة الألعاب التي تم لعبها. بالنسبة للمجموعة (A)، يتوقع هذا نتيجة مثل:

	Group A								
Senegal		Netherlands	1-3						
Qatar		Ecuador	0-1						
Qatar		Senegal	2-1						
Netherlands		Ecuador	3-1						
Ecuador		Senegal	0-1						
Netherlands		Qatar	3-1						

	pd	W	D	L	Gf	Ga	Gd	Pt
Netherlands	3	3	0	0	9	3	6	9
Senegal	3	1	0	2	3	5	-2	3
Qatar	3	1	0	2	3	5	-2	3
Ecuador	3	1	0	2	2	4	-2	3

يبدو الأمر منطقيًا بالنسبة لي. هولندا تتقدم!

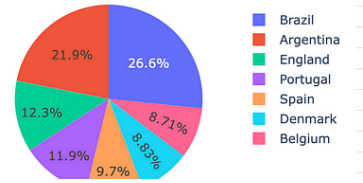
يمكننا أن نستمرفي هذا الأمر ونحاكي جميع المباريات حتى المباراة النهائية. إليكم إحدى المباريات النهائية المتوقعة:

81					
82		Semi Finals			
83	Denmark		Belgium		0-0
84	England		Morocco		2-0
85					
86		Final			
87	Belgium		England		2-1
88					
89		Winner			
90	Belgium				

محاكاة بطولات متعددة

لذا، كل ما علينا فعله الآن هو تشغيل البطولة ألف مرة وسنحصل على التنبؤ التالي:

	Team	Wins	Percentage
10	Brazil	214	21.4%
11	Argentina	176	17.6%
12	England	99	9.9%
13	Portugal	96	9.6%
14	Spain	78	7.8%
15	Denmark	71	7.1%
16	Belgium	70	7%
17	Netherlands	41	4.1%
18	France	39	3.9%
19	Germany	35	3.5%



لذا، فإن البرازيل هي المرشحة المفضلة! من كان ليتصور ذلك. ولكن هولندا هي المرشحة المفضلة بنسبة 4.1%.

هذا كل شيء لهذا اليوم. إذا كنت تريد إلقاء نظرة على الكود، فألق نظرة على جدول بيانات Neptyne.

المصدر:

https://dosinga.medium.com/predicting-the-outcome-of-the-world-cup-in-150-lines-of-python-c72772667a6e

8) تحليل الفيفا باستخدام علم البيانات FIFA Analysis with Data Science

تأسس الاتحاد الدولي لكرة القدم Federation Internationale de Football Association (FIFA)في عام 1904 بهدف توفير الوحدة بين اتحادات كرة القدم الوطنية، ويضم الاتحاد 209 عضوًا، وهو ما ينافس الأمم المتحدة، ويمكن القول إنه المنظمة الرياضية الأكثر شهرةفي العالم.

في مشروع علوم البيانات Data Science Project هذا، سنقوم بإجراء بعض التحليلات على مباريات وسجلات FIFA باستخدام بايثون.

استيراد المكتبات

لنبدأ باستيراد المكتبات:

```
import numpy as np
import pandas as pd

 #for visualizations
import matplotlib.pyplot as plt
import seaborn as sns
sns.set()
```

يمكنك تنزيل مجموعة البيانات التي نحتاجها لهذه المهمة من هنا.

```
data = pd.read_csv('data.csv')
print(data.shape)
```

```
#Output- (18207, 89)
```

للتحقق من أول 5 صفوف وأعمدة:

```
data.head()
```

	Unnamed: 0	ID	Name	Age	Photo	Nationality	Flag	Overall
0	0	158023	L. Messi	31	https://cdn.sofifa.org/players/4/19/158023.png	Argentina	https://cdn.sofifa.org/flags/52.png	94
1	1	20801	Cristiano Ronaldo	33	https://cdn.sofifa.org/players/4/19/20801.png	Portugal	https://cdn.sofifa.org/flags/38.png	94
2	2	190871	Neymar Jr	26	https://cdn.sofifa.org/players/4/19/190871.png	Brazil	https://cdn.sofifa.org/flags/54.png	92
3	3	193080	De Gea	27	https://cdn.sofifa.org/players/4/19/193080.png	Spain	https://cdn.sofifa.org/flags/45.png	91
4	4	192985	K. De Bruyne	27	https://cdn.sofifa.org/players/4/19/192985.png	Belgium	https://cdn.sofifa.org/flags/7.png	91

5 rows × 89 columns

دعونا نلقي نظرة على لاعبي كرة القدم الهنود:

```
def country(x):
    return data[data['Nationality'] ==
x][['Name','Overall','Potential','Position']]


# let's check the Indian Players
country('India')
```

	Name	Overall	Potential	Position
8605	S. Chhetri	67	67	LS
10011	S. Jhingan	65	71	RCB
12598	J. Lalpekhlua	63	64	RS
12811	G. Singh Sandhu	63	68	GK
13508	A. Edathodika	62	62	LCB
14054	P. Halder	61	67	RCM
14199	P. Kotal	61	66	RB
14218	L. Ralte	61	62	LW
14705	N. Das	60	65	LB
14786	U. Singh	60	67	RM
14915	H. Narzary	60	66	LM
15356	R. Singh	59	59	ST
15643	S. Singh	59	65	CB
15652	A. Thapa	59	71	LCM

تحليل بيانات النادي(مانشستر يونايتد):

```
def club(x):
    return data[data['Club'] == x][['Name','Jersey
Number','Position','Overall','Nationality','Age','Wage',
                                    'Value','Contract Valid Until']]

club('Manchester United')
```

45	P. Pogba	6.0	RDM	87	France	25	€210K	€64M	2021
47	R. Lukaku	9.0	ST	87	Belgium	25	€230K	€62.5M	2022
93	A. Sánchez	7.0	RW	85	Chile	29	€215K	€37.5M	2022
116	A. Martial	11.0	LW	84	France	22	€165K	€42.5M	2019
132	N. Matić	31.0	CDM	84	Serbia	29	€165K	€24M	2020
211	Juan Mata	8.0	RM	83	Spain	30	€160K	€24.5M	2019
250	Fred	17.0	CM	82	Brazil	25	€140K	€26.5M	2023
254	J. Lingard	7.0	CAM	82	England	25	€140K	€26.5M	2021
319	M. Rashford	11.0	LW	81	England	20	€110K	€27M	2020
327	E. Bailly	2.0	CB	81	Ivory Coast	24	€105K	€21M	2020
374	Ander Herrera	21.0	CM	81	Spain	28	€140K	€17.5M	2019
377	C. Smalling	12.0	RCB	81	England	28	€130K	€16M	2019
399	A. Valencia	16.0	RM	81	Ecuador	32	€120K	€10M	2019

```
x = club('Manchester United')
x.shape
```

```
#Output- (33, 9)
```

وصف البيانات

```
data.describe()
```

	Unnamed: 0	ID	Age	Overall	Potential	Special	International Reputation	Weak Foot	Skill Moves
count	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18207.000000	18159.000000	18159.000000	18159.000000
mean	9103.000000	214298.338606	25.122206	66.238699	71.307299	1597.809908	1.113222	2.947299	2.361308
std	5256.052511	29965.244204	4.669943	6.908930	6.136496	272.586016	0.394031	0.660456	0.756164
min	0.000000	16.000000	16.000000	46.000000	48.000000	731.000000	1.000000	1.000000	1.000000
25%	4551.500000	200315.500000	21.000000	62.000000	67.000000	1457.000000	1.000000	3.000000	2.000000
50%	9103.000000	221759.000000	25.000000	66.000000	71.000000	1635.000000	1.000000	3.000000	2.000000
75%	13654.500000	236529.500000	28.000000	71.000000	75.000000	1787.000000	1.000000	3.000000	3.000000
max	18206.000000	246620.000000	45.000000	94.000000	95.000000	2346.000000	5.000000	5.000000	5.000000

ملء القيمة المفقودة missing value للمتغيرات المستمرة لتصور البيانات بشكل صحيح:

```
data['ShortPassing'].fillna(data['ShortPassing'].mean(), inplace = True)
data['Volleys'].fillna(data['Volleys'].mean(), inplace = True)
data['Dribbling'].fillna(data['Dribbling'].mean(), inplace = True)
data['Curve'].fillna(data['Curve'].mean(), inplace = True)
data['FKAccuracy'].fillna(data['FKAccuracy'], inplace = True)
data['LongPassing'].fillna(data['LongPassing'].mean(), inplace = True)
data['BallControl'].fillna(data['BallControl'].mean(), inplace = True)
data['HeadingAccuracy'].fillna(data['HeadingAccuracy'].mean(), inplace =
True)
data['Finishing'].fillna(data['Finishing'].mean(), inplace = True)
data['Crossing'].fillna(data['Crossing'].mean(), inplace = True)
data['Weight'].fillna('200lbs', inplace = True)
data['Contract Valid Until'].fillna(2019, inplace = True)
data['Height'].fillna("5'11", inplace = True)
data['Loaned From'].fillna('None', inplace = True)
data['Joined'].fillna('Jul 1, 2018', inplace = True)
data['Jersey Number'].fillna(8, inplace = True)
data['Body Type'].fillna('Normal', inplace = True)
data['Position'].fillna('ST', inplace = True)
data['Club'].fillna('No Club', inplace = True)
data['Work Rate'].fillna('Medium/ Medium', inplace = True)
data['Skill Moves'].fillna(data['Skill Moves'].median(), inplace = True)
data['Weak Foot'].fillna(3, inplace = True)
data['Preferred Foot'].fillna('Right', inplace = True)
data['International Reputation'].fillna(1, inplace = True)
data['Wage'].fillna('€200K', inplace = True)
data.fillna(0, inplace = True)
```

```
def defending(data):
    return int(round((data[['Marking', 'StandingTackle',
                            'SlidingTackle']].mean()).mean()))

def general(data):
    return int(round((data[['HeadingAccuracy', 'Dribbling', 'Curve',
                            'BallControl']].mean()).mean()))

def mental(data):
    return int(round((data[['Aggression', 'Interceptions', 'Positioning',
                            'Vision','Composure']].mean()).mean()))
```

```
def passing(data):
    return int(round((data[['Crossing', 'ShortPassing',
                            'LongPassing']].mean()).mean()))

def mobility(data):
    return int(round((data[['Acceleration', 'SprintSpeed',
                            'Agility','Reactions']].mean()).mean()))
def power(data):
    return int(round((data[['Balance', 'Jumping', 'Stamina',
                            'Strength']].mean()).mean()))

def rating(data):
    return int(round((data[['Potential', 'Overall']].mean()).mean()))

def shooting(data):
    return int(round((data[['Finishing', 'Volleys', 'FKAccuracy',
                            'ShotPower','LongShots',
'Penalties']].mean()).mean()))
```

إعادة تسمية الأعمدة

```
data.rename(columns={'Club Logo':'Club_Logo'}, inplace=True)

 #adding these categories to the data

data['Defending'] = data.apply(defending, axis = 1)
data['General'] = data.apply(general, axis = 1)
data['Mental'] = data.apply(mental, axis = 1)
data['Passing'] = data.apply(passing, axis = 1)
data['Mobility'] = data.apply(mobility, axis = 1)
data['Power'] = data.apply(power, axis = 1)
data['Rating'] = data.apply(rating, axis = 1)
data['Shooting'] = data.apply(shooting, axis = 1)
```

```
players = data[['Name','Defending','General','Mental','Passing',
                'Mobility','Power','Rating','Shooting','Flag','Age',
                'Nationality', 'Photo', 'Club Logo', 'Club']]

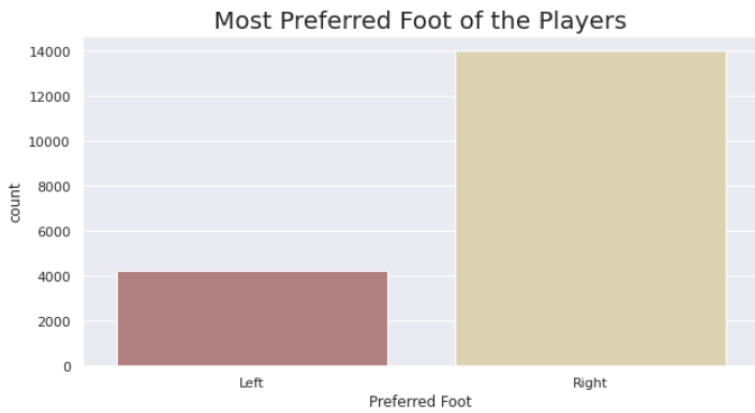
players.head()
```

	Name	Defending	General	Mental	Passing	Mobility	Power	Rating	Shooting	Flag	Age	Nationality
0	L. Messi	29	89	71	87	91	74	94	88	https://cdn.sofifa.org/flags/52.png	31	Argentina
1	Cristiano Ronaldo	27	88	73	81	91	83	94	88	https://cdn.sofifa.org/flags/38.png	33	Portugal
2	Neymar Jr	28	85	72	80	94	69	92	84	https://cdn.sofifa.org/flags/54.png	26	Brazil
3	De Gea	16	26	43	39	66	54	92	21	https://cdn.sofifa.org/flags/45.png	27	Spain
4	K. De Bruyne	59	79	81	92	81	76	92	85	https://cdn.sofifa.org/flags/7.png	27	Belgium

تصور البيانات

مقارنة القدم المفضلة لدى اللاعبين المختلفين:

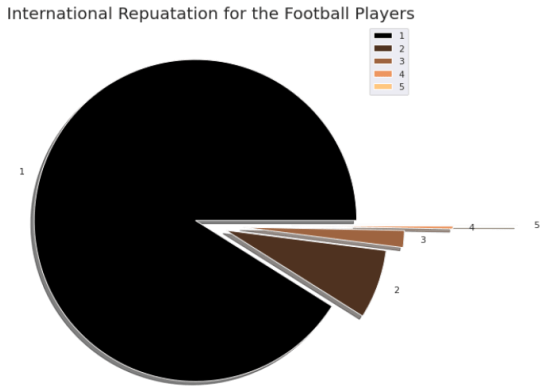
```
                                    plt.rcParams['figure.figsize'] = (10, 5)
                          sns.countplot(data['Preferred Foot'], palette = 'pink')
                 plt.title('Most Preferred Foot of the Players', fontsize = 20)
                                                                  plt.show()
```



رسم مخطط دائري pie chart لتمثيل حصة السمعة الدولية international reputation:

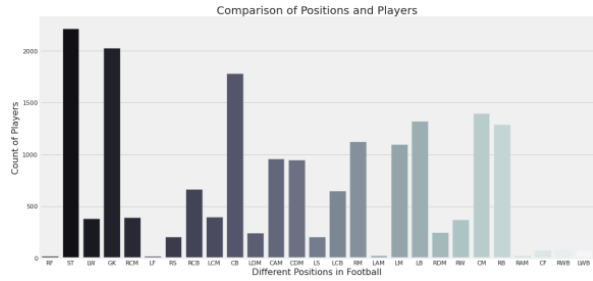
```
labels = ['1', '2', '3', '4', '5']
sizes = data['International Reputation'].value_counts()
colors = plt.cm.copper(np.linspace(0, 1, 5))
explode = [0.1, 0.1, 0.2, 0.5, 0.9]

plt.rcParams['figure.figsize'] = (9, 9)
plt.pie(sizes, labels = labels, colors = colors, explode = explode, shadow
= True)
plt.title('International Repuatation for the Football Players', fontsize =
20)
plt.legend()
plt.show()
```



المراكز المختلفة التي حصل عليها اللاعبون:

```
plt.figure(figsize = (18, 8))
plt.style.use('fivethirtyeight')
ax = sns.countplot('Position', data = data, palette = 'bone')
ax.set_xlabel(xlabel = 'Different Positions in Football', fontsize = 16)
ax.set_ylabel(ylabel = 'Count of Players', fontsize = 16)
ax.set_title(label = 'Comparison of Positions and Players', fontsize = 20)
plt.show()
```



تعريف دالة لتنظيف بيانات الوزن Weight data:

```
def extract_value_from(value):
  out = value.replace('lbs', '')
  return float(out)

 #applying the function to weight column
#data['value'] = data['value'].apply(lambda x: extract_value_from(x))
data['Weight'] = data['Weight'].apply(lambda x : extract_value_from(x))

data['Weight'].head()
```

```
#Output
0    159.0
1    183.0
2    150.0
3    168.0
4    154.0
Name: Weight, dtype: float64
```

تعريف دالة لتنظيف عمود الأجور wage column:

```
def extract_value_from(Value):
    out = Value.replace('€', '')
    if 'M' in out:
        out = float(out.replace('M', ''))*1000000
    elif 'K' in Value:
        out = float(out.replace('K', ''))*1000
    return float(out)
```

تطبيق الدالة على عمود الأجور:

```
data['Value'] = data['Value'].apply(lambda x: extract_value_from(x))
data['Wage'] = data['Wage'].apply(lambda x: extract_value_from(x))

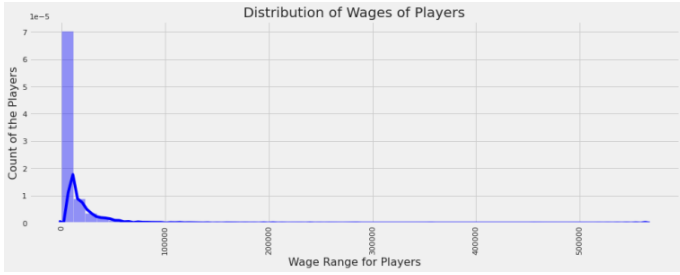
data['Wage'].head()
```

```
#Output
0    565000.0
1    405000.0
2    290000.0
3    260000.0
4    355000.0
Name: Wage, dtype: float64
```

مقارنة أجور اللاعبين players' Wages:

```
import warnings
warnings.filterwarnings('ignore')

plt.rcParams['figure.figsize'] = (15, 5)
sns.distplot(data['Wage'], color = 'blue')
plt.xlabel('Wage Range for Players', fontsize = 16)
plt.ylabel('Count of the Players', fontsize = 16)
plt.title('Distribution of Wages of Players', fontsize = 20)
plt.xticks(rotation = 90)
plt.show()
```

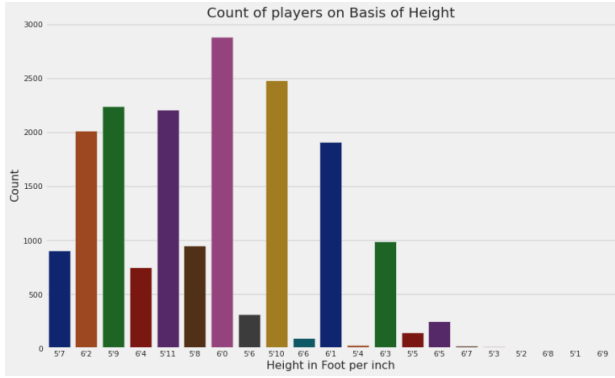


حركات المهارة للاعبين Skill Moves of Players:

```
plt.figure(figsize = (10, 8))
ax = sns.countplot(x = 'Skill Moves', data = data, palette = 'pastel')
ax.set_title(label = 'Count of players on Basis of their skill moves',
fontsize = 20)
ax.set_xlabel(xlabel = 'Number of Skill Moves', fontsize = 16)
ax.set_ylabel(ylabel = 'Count', fontsize = 16)
plt.show()
```

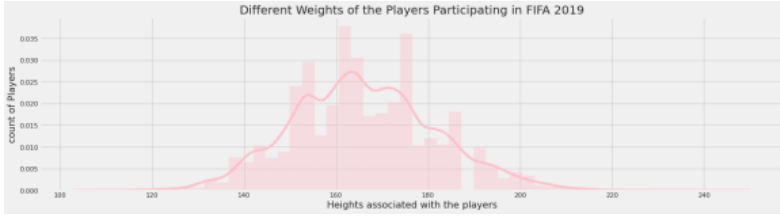
ارتفاع اللاعبين Height of Players:

```
plt.figure(figsize = (13, 8))
ax = sns.countplot(x = 'Height', data = data, palette = 'dark')
ax.set_title(label = 'Count of players on Basis of Height', fontsize = 20)
ax.set_xlabel(xlabel = 'Height in Foot per inch', fontsize = 16)
ax.set_ylabel(ylabel = 'Count', fontsize = 16)
plt.show()
```



لإظهار أوزان أجسام اللاعبين المشاركينفي بطولة فيفا 2019 المختلفة:

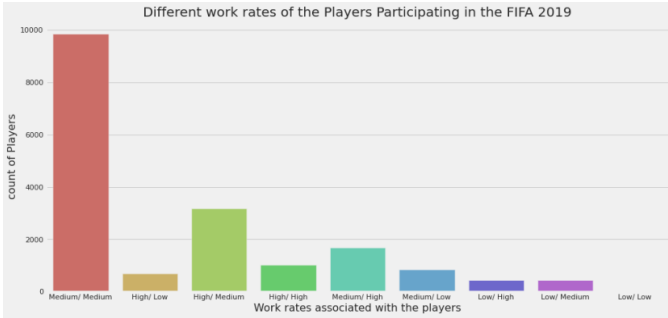
```
plt.figure(figsize = (20, 5))
sns.distplot(data['Weight'], color = 'pink')
plt.title('Different Weights of the Players Participating in FIFA 2019',
fontsize = 20)
plt.xlabel('Heights associated with the players', fontsize = 16)
plt.ylabel('count of Players', fontsize = 16)
plt.show()
```



لإظهار معدل العمل Different Work rate المختلف للاعبين المشاركينفي FIFA 2019:

```
plt.figure(figsize = (15, 7))

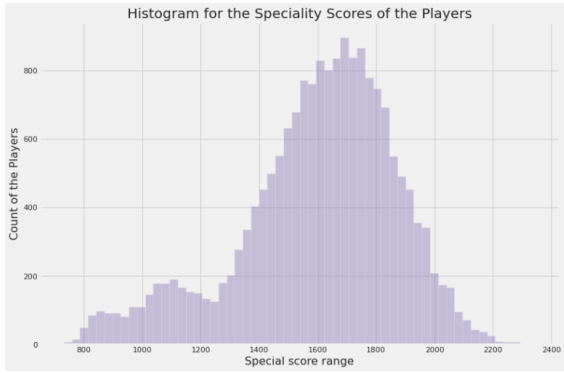
sns.countplot(x = 'Work Rate', data = data, palette = 'hls')
plt.title('Different work rates of the Players Participating in the FIFA
2019', fontsize = 20)
plt.xlabel('Work rates associated with the players', fontsize = 16)
plt.ylabel('count of Players', fontsize = 16)
plt.show()
```



لإظهار درجات التخصص المختلفة Different Speciality Score للاعبين المشاركينفي FIFA 2019:

```
x = data.Special
plt.figure(figsize = (12, 8))
plt.style.use('tableau-colorblind10')

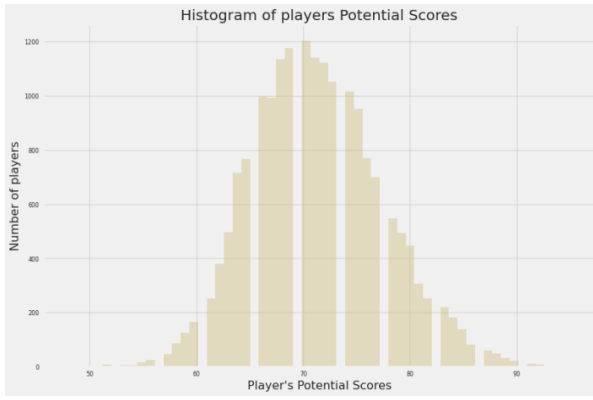
ax = sns.distplot(x, bins = 58, kde = False, color = 'm')
ax.set_xlabel(xlabel = 'Special score range', fontsize = 16)
ax.set_ylabel(ylabel = 'Count of the Players',fontsize = 16)
ax.set_title(label = 'Histogram for the Speciality Scores of the Players',
fontsize = 20)
plt.show()
```



لإظهار النتائج المحتملة المختلفة Different potential scores للاعبين المشاركينفي FIFA 2019:

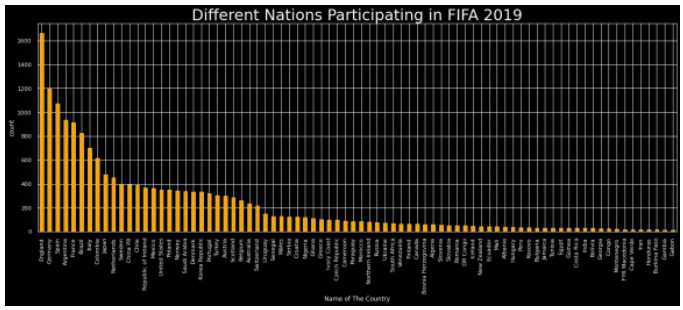
```
x = data.Potential
plt.figure(figsize=(12,8))
plt.style.use('seaborn-paper')

ax = sns.distplot(x, bins = 58, kde = False, color = 'y')
ax.set_xlabel(xlabel = "Player\'s Potential Scores", fontsize = 16)
ax.set_ylabel(ylabel = 'Number of players', fontsize = 16)
ax.set_title(label = 'Histogram of players Potential Scores', fontsize =
20)
plt.show()
```



لإظهار الدول المختلفة Different nations المشاركةفي كأس العالم FIFA 2019:

```
plt.style.use('dark_background')
data['Nationality'].value_counts().head(80).plot.bar(color = 'orange',
figsize = (20, 7))
plt.title('Different Nations Participating in FIFA 2019', fontsize = 30,
fontweight = 20)
plt.xlabel('Name of The Country')
plt.ylabel('count')
plt.show()
```



الدول التي تضم أكبر عدد من اللاعبين

اختيار الدول التي تضم أكبر عدد من اللاعبين لمقارنة نتائجها الإجمالية:

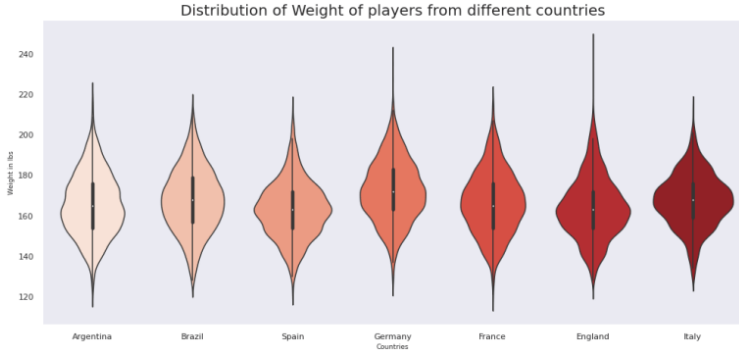
```
data['Nationality'].value_counts().head(8)
```

```
#Output-
England      1662
Germany      1198
Spain        1072
Argentina     937
France        914
Brazil        827
Italy         702
Colombia      618
Name: Nationality, dtype: int64
```

لاعبو كل الأمم وأوزانهم:

```
some_countries = ('England', 'Germany', 'Spain', 'Argentina', 'France',
'Brazil', 'Italy', 'Columbia')
data_countries = data.loc[data['Nationality'].isin(some_countries) &
data['Weight']]

plt.rcParams['figure.figsize'] = (15, 7)
ax = sns.violinplot(x = data_countries['Nationality'], y =
data_countries['Weight'], palette = 'Reds')
ax.set_xlabel(xlabel = 'Countries', fontsize = 9)
ax.set_ylabel(ylabel = 'Weight in lbs', fontsize = 9)
ax.set_title(label = 'Distribution of Weight of players from different
countries', fontsize = 20)
plt.show()
```



العثور على الأندية المشهورة popular clubsفي جميع أنحاء العالم:

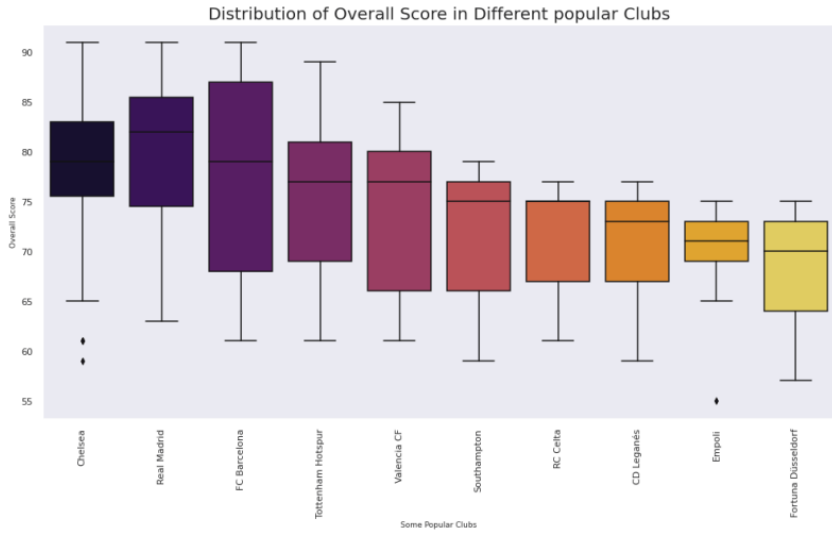
```
data['Club'].value_counts().head(10)
```

```
#Output
No Club                241
Liverpool               33
TSG 1899 Hoffenheim     33
Burnley                 33
CD Leganés              33
Arsenal                 33
Southampton             33
Frosinone               33
Empoli                  33
Fortuna Düsseldorf      33
Name: Club, dtype: int64
```

```
some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna
Düsseldorf', 'Manchestar City',
              'Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea',
'Real Madrid')
```

```
data_clubs = data.loc[data['Club'].isin(some_clubs) & data['Overall']]

plt.rcParams['figure.figsize'] = (15, 8)
ax = sns.boxplot(x = data_clubs['Club'], y = data_clubs['Overall'], palette
= 'inferno')
ax.set_xlabel(xlabel = 'Some Popular Clubs', fontsize = 9)
ax.set_ylabel(ylabel = 'Overall Score', fontsize = 9)
ax.set_title(label = 'Distribution of Overall Score in Different popular
Clubs', fontsize = 20)
plt.xticks(rotation = 90)
plt.show()
```

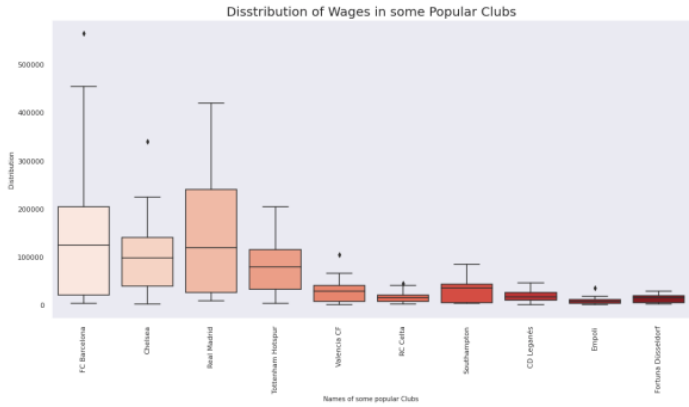


توزيع الرواتبفي بعض الأندية المشهورة:

```
some_clubs = ('CD Leganés', 'Southampton', 'RC Celta', 'Empoli', 'Fortuna
Düsseldorf', 'Manchestar City,'
'              Tottenham Hotspur', 'FC Barcelona', 'Valencia CF', 'Chelsea',
'Real Madrid('

data_club = data.loc[data['Club'].isin(some_clubs) & data['Wage']]

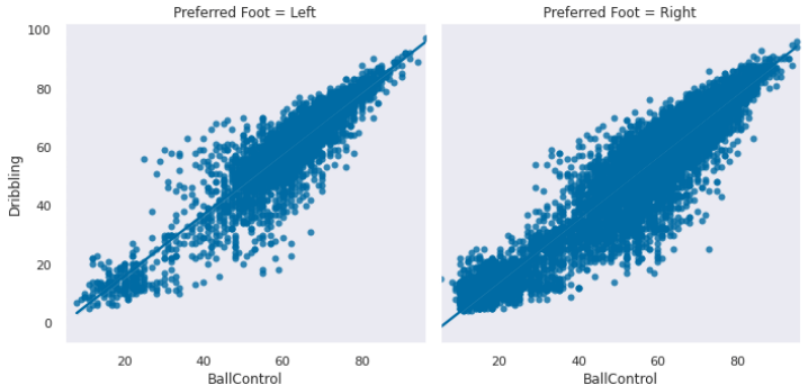
plt.rcParams['figure.figsize'](8 ,16) =
ax = sns.boxplot(x = 'Club', y = 'Wage', data = data_club, palette =
'Reds')
ax.set_xlabel(xlabel = 'Names of some popular Clubs', fontsize = 10)
ax.set_ylabel(ylabel = 'Distribution', fontsize = 10)
ax.set_title(label = 'Disstribution of Wages in some Popular Clubs',
fontsize = 20)
plt.xticks(rotation = 90)
plt.show()
```



مقارنة أداء لاعبي كرة القدم الذين يستخدمون القدم اليسرى والقدم اليمنى –left–footed and right–footed footballers:

```
# ballcontrol vs dribbing

sns.lmplot(x = 'BallControl', y = 'Dribbling', data = data, col =
'Preferred Foot')
plt.show()
```



المصدر:

/https://thecleverprogrammer.com/2020/05/22/fifa-analysis-with-data-science

9) تحليل بيانات كرة القدم الأوروبية European Soccer Data Analysis

تحليل قاعدة بيانات كرة القدم

يهدف هذا المشروع إلى استكشاف وتحليل قاعدة بيانات كرة القدم soccer database باستخدام تقنيات تحليل البيانات المختلفة والتصورات. تحتوي قاعدة البيانات على معلومات حول اللاعبين والفرق والمباريات والدوريات والمزيد. سنقوم بتحميل البيانات إلى إطارات بيانات pandas وإجراء سلسلة من التحليلات لاكتساب رؤى حول سمات اللاعبين وأداء الفريق ونتائج المباريات وغيرها من جوانب كرة القدم.

تحميل البيانات وفحصها

أولاً، نقوم بتحميل كل جدول من قاعدة بيانات SQLite إلى إطارات بيانات pandas. هذه الخطوة بالغة الأهمية للوصول إلى البيانات ومعالجتها بسهولة.

من خلال فحص الصفوف القليلة الأولى من كل إطار بيانات، نحصل على نظرة عامة على البيانات المتوفرةفي كل جدول، مما يساعدنافي التخطيط لتحليلاتنا اللاحقة.

الإحصاءات الوصفية ومخططات التوزيع

بعد ذلك، نحسب إحصاءات موجزة للسمات الرئيسيةفي جداول سمات اللاعبين والفرق. تتضمن الإحصاءات الموجزة مقاييس مثل المتوسط mean والوسيط median والانحراف المعياري standard deviation، والتي توفر نظرة عامة سريعة على توزيع البيانات.

نقوم أيضًا بإنشاء مخططات توزيع للسمات الرئيسية لتصور توزيع وتكرار القيم المحددةفي مجموعة البيانات. على سبيل المثال، نقوم بإنشاء مخططات بيانية histograms لإظهار توزيع تقييمات اللاعبين الإجمالية وسرعة بناء اللعب للفريق.

تحليل نتائج المباريات

أخيرًا، نقوم بتحليل نتائج المباريات match outcomes من خلال فحص إحصاءات موجزة لعدد الأهداف التي سجلتها الفرق المضيفة والفرق الزائرة home and away teams. نحسب متوسط عدد الأهداف المسجلة لكل مباراة من قبل كل من الفرق المضيفة والفرق الزائرة ونتصور توزيع إجمالي الأهداف لكل مباراة من خلال إنشاء مخطط بياني. يساعدنا هذافي فهم تكرار نتائج المباريات المختلفة، مثل المباريات ذات الأهداف العالية مقابل المباريات ذات الأهداف المنخفضة.

من خلال إجراء هذه التحليلات، نكتسب رؤى قيمة حول بيانات كرة القدم، والتي يمكن أن توجه المزيد من التحليلات واتخاذ القرارات.

```
# This Python 3 environment comes with many helpful analytics libraries
installed
# It is defined by the kaggle/python Docker image:
https://github.com/kaggle/docker-python
# For example, here's several helpful packages to load

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)

# Input data files are available in the read-only "../input/" directory
# For example, running this (by clicking run or pressing Shift+Enter) will
list all files under the input directory

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))

# You can write up to 20GB to the current directory (/kaggle/working/) that
gets preserved as output when you create a version using "Save & Run All"
# You can also write temporary files to /kaggle/temp/, but they won't be
saved outside of the current session
```

```
/kaggle/input/soccer/database.sqlite
```

```
#Improts

import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import sqlite3
import matplotlib.pyplot as plt

# Input data files are available in the "../input/" directory.
# For example, running this (by clicking run or pressing Shift+Enter) will
list the files in the input directory


conn = sqlite3.connect("/kaggle/input/soccer/database.sqlite")

tables = pd.read_sql("""SELECT *
                        FROM sqlite_master
                        WHERE type='table';""", conn)
tables
```

	type	name	tbl_name	rootpage	sql
0	table	sqlite_sequence	sqlite_sequence	4	CREATE TABLE sqlite_sequence(name,seq)
1	table	Player_Attributes	Player_Attributes	11	CREATE TABLE "Player_Attributes" (\n\t`id`\tIN...
2	table	Player	Player	14	CREATE TABLE `Player` (\n\t`id`\tINTEGER PRIMA...
3	table	Match	Match	18	CREATE TABLE `Match` (\n\t`id`\tINTEGER PRIMAR...
4	table	League	League	24	CREATE TABLE `League` (\n\t`id`\tINTEGER PRIMA...
5	table	Country	Country	26	CREATE TABLE `Country` (\n\t`id`\tINTEGER PRIM...
6	table	Team	Team	29	CREATE TABLE "Team" (\n\t`id`\tINTEGER PRIMARY...
7	table	Team_Attributes	Team_Attributes	2	CREATE TABLE `Team_Attributes` (\n\t`id`\tINTE...

استكشاف قاعدة بيانات كرة القدم

تحميل كل جدول إلى إطار بيانات Pandas

أولاً، نقوم بتحميل كل جدول من قاعدة بيانات SQLite إلى إطار بيانات Pandas.

```
# Load each table into a pandas DataFrame
player_attributes = pd.read_sql("SELECT * FROM Player_Attributes", conn)
player = pd.read_sql("SELECT * FROM Player", conn)
match = pd.read_sql("SELECT * FROM Match", conn)
league = pd.read_sql("SELECT * FROM League", conn)
country = pd.read_sql("SELECT * FROM Country", conn)
team = pd.read_sql("SELECT * FROM Team", conn)
team_attributes = pd.read_sql("SELECT * FROM Team_Attributes", conn)

# Inspect the first few rows of each DataFrame
print(player_attributes.head())
print(player.head())
print(match.head())
print(league.head())
print(country.head())
print(team.head())
print(team_attributes.head())
```

```
  id  player_fifa_api_id  player_api_id        date  overall_rating  \
0   1              218353         505942  2016-02-18 00:00:00
67.0
1   2              218353         505942  2015-11-19 00:00:00
67.0
2   3              218353         505942  2015-09-21 00:00:00
62.0
3   4              218353         505942  2015-03-20 00:00:00
61.0
4   5              218353         505942  2007-02-22 00:00:00
61.0

   potential preferred_foot attacking_work_rate defensive_work_rate
crossing  \
0       71.0          right              medium              medium
49.0
1       71.0          right              medium              medium
49.0
2       66.0          right              medium              medium
49.0
3       65.0          right              medium              medium
48.0
4       65.0          right              medium              medium
48.0

   ...  vision  penalties  marking  standing_tackle  sliding_tackle  \
0  ...    54.0       48.0     65.0             69.0            69.0
1  ...    54.0       48.0     65.0             69.0            69.0
2  ...    54.0       48.0     65.0             66.0            69.0
3  ...    53.0       47.0     62.0             63.0            66.0
4  ...    53.0       47.0     62.0             63.0            66.0

   gk_diving  gk_handling  gk_kicking  gk_positioning  gk_reflexes
0        6.0         11.0        10.0             8.0          8.0
1        6.0         11.0        10.0             8.0          8.0
```

```
2         6.0          11.0         10.0              8.0          8.0
3         5.0          10.0          9.0              7.0          7.0
4         5.0          10.0          9.0              7.0          7.0

[5 rows x 42 columns]
   id  player_api_id         player_name  player_fifa_api_id  \
0   1         505942  Aaron Appindangoye              218353
1   2         155782     Aaron Cresswell              189615
2   3         162549         Aaron Doran              186170
3   4          30572       Aaron Galindo              140161
4   5          23780        Aaron Hughes               17725

              birthday  height  weight
0  1992-02-29 00:00:00  182.88     187
1  1989-12-15 00:00:00  170.18     146
2  1991-05-13 00:00:00  170.18     163
3  1982-05-08 00:00:00  182.88     198
4  1979-11-08 00:00:00  182.88     154
   id  country_id  league_id     season  stage                 date  \
0   1           1          1  2008/2009      1  2008-08-17 00:00:00
1   2           1          1  2008/2009      1  2008-08-16 00:00:00
2   3           1          1  2008/2009      1  2008-08-16 00:00:00
3   4           1          1  2008/2009      1  2008-08-17 00:00:00
4   5           1          1  2008/2009      1  2008-08-16 00:00:00

   match_api_id  home_team_api_id  away_team_api_id  home_team_goal  ...  \
0        492473              9987              9993               1  ...
1        492474             10000              9994               0  ...
2        492475              9984              8635               0  ...
3        492476              9991              9998               5  ...
4        492477              7947              9985               1  ...

    SJA   VCH   VCD   VCA   GBH   GBD   GBA   BSH   BSD   BSA
0  4.00  1.65  3.40  4.50  1.78  3.25  4.00  1.73  3.40  4.20
1  3.80  2.00  3.25  3.25  1.85  3.25  3.75  1.91  3.25  3.60
2  2.50  2.35  3.25  2.65  2.50  3.20  2.50  2.30  3.20  2.75
3  7.50  1.45  3.75  6.50  1.50  3.75  5.50  1.44  3.75  6.50
4  1.73  4.50  3.40  1.65  4.50  3.50  1.65  4.75  3.30  1.67

[5 rows x 115 columns]
      id  country_id                     name
0      1           1  Belgium Jupiler League
1   1729        1729  England Premier League
2   4769        4769          France Ligue 1
3   7809        7809   Germany 1. Bundesliga
4  10257       10257           Italy Serie A
      id     name
0      1  Belgium
1   1729  England
2   4769   France
3   7809  Germany
4  10257    Italy
   id  team_api_id  team_fifa_api_id     team_long_name team_short_name
0   1         9987             673.0          KRC Genk             GEN
1   2         9993             675.0      Beerschot AC             BAC
2   3        10000           15005.0   SV Zulte-Waregem             ZUL
3   4         9994            2007.0   Sporting Lokeren             LOK
4   5         9984            1750.0  KSV Cercle Brugge             CEB
   id  team_fifa_api_id  team_api_id                 date  buildUpPlaySpeed
\
0   1               434         9930  2010-02-22 00:00:00                60
1   2               434         9930  2014-09-19 00:00:00                52
```

```
2   3               434         9930  2015-09-10 00:00:00                47
3   4                77         8485  2010-02-22 00:00:00                70
4   5                77         8485  2011-02-22 00:00:00                47

  buildUpPlaySpeedClass  buildUpPlayDribbling buildUpPlayDribblingClass  \
0              Balanced                   NaN                    Little
1              Balanced                  48.0                    Normal
2              Balanced                  41.0                    Normal
3                  Fast                   NaN                    Little
4              Balanced                   NaN                    Little

   buildUpPlayPassing buildUpPlayPassingClass  ... chanceCreationShooting
\
0                  50                   Mixed  ...                      55
1                  56                   Mixed  ...                      64
2                  54                   Mixed  ...                      64
3                  70                    Long  ...                      70
4                  52                   Mixed  ...                      52

   chanceCreationShootingClass chanceCreationPositioningClass  \
0                       Normal                      Organised
1                       Normal                      Organised
2                       Normal                      Organised
3                         Lots                      Organised
4                       Normal                      Organised

   defencePressure defencePressureClass  defenceAggression  \
0               50               Medium                 55
1               47               Medium                 44
2               47               Medium                 44
3               60               Medium                 70
4               47               Medium                 47

  defenceAggressionClass defenceTeamWidth  defenceTeamWidthClass  \
0                  Press               45                 Normal
1                  Press               54                 Normal
2                  Press               54                 Normal
3                 Double               70                   Wide
4                  Press               52                 Normal

  defenceDefenderLineClass
0                    Cover
1                    Cover
2                    Cover
3                    Cover
4                    Cover

[5 rows x 25 columns]
```

الإحصاءات الوصفية ومخططات التوزيع

إحصاءات الملخص

نبدأ بحساب إحصاءات الملخص للسمات الرئيسيةفي إطارات بيانات player_attributes و team_attributes تتضمن إحصاءات الملخص مقاييس مثل المتوسط والوسيط والانحراف المعياري وما إلى ذلك، والتي توفر نظرة عامة سريعة على توزيع البيانات.

```
# Summary statistics
print(player_attributes.describe())
print(team_attributes.describe())
```

```
# Distribution plots
player_attributes['overall_rating'].hist(bins=20)
plt.title('Distribution of Overall Player Ratings')
plt.xlabel('Overall Rating')
plt.ylabel('Frequency')
plt.show()

team_attributes['buildUpPlaySpeed'].hist(bins=20)
plt.title('Distribution of Team Build Up Play Speed')
plt.xlabel('Build Up Play Speed')
plt.ylabel('Frequency')
plt.show()
```

```
id  player_fifa_api_id  player_api_id  overall_rating  \
count  183978.00000       183978.000000  183978.000000   183142.000000
mean    91989.50000       165671.524291  135900.617324       68.600015
std     53110.01825        53851.094769  136927.840510        7.041139
min         1.00000            2.000000    2625.000000       33.000000
25%     45995.25000       155798.000000   34763.000000       64.000000
50%     91989.50000       183488.000000   77741.000000       69.000000
75%    137983.75000       199848.000000  191080.000000       73.000000
max    183978.00000       234141.000000  750584.000000       94.000000

           potential       crossing      finishing  heading_accuracy  \
count  183142.000000  183142.000000  183142.000000     183142.000000
mean       73.460353      55.086883      49.921078         57.266023
std         6.592271      17.242135      19.038705         16.488905
min        39.000000       1.000000       1.000000          1.000000
25%        69.000000      45.000000      34.000000         49.000000
50%        74.000000      59.000000      53.000000         60.000000
75%        78.000000      68.000000      65.000000         68.000000
max        97.000000      95.000000      97.000000         98.000000

       short passing        volleys  ...         vision      penalties  \
count  183142.000000  181265.000000  ...  181265.000000  183142.000000
mean       62.429672      49.468436  ...      57.873550      55.003986
std        14.194068      18.256618  ...      15.144086      15.546519
min         3.000000       1.000000  ...       1.000000       2.000000
25%        57.000000      35.000000  ...      49.000000      45.000000
50%        65.000000      52.000000  ...      60.000000      57.000000
75%        72.000000      64.000000  ...      69.000000      67.000000
max        97.000000      93.000000  ...      97.000000      96.000000

             marking  standing_tackle  sliding_tackle      gk_diving  \
count  183142.000000    183142.000000   181265.000000  183142.000000
mean       46.772242        50.351257       48.001462      14.704393
std        21.227667        21.483706       21.598778      16.865467
min         1.000000         1.000000        2.000000       1.000000
25%        25.000000        29.000000       25.000000       7.000000
50%        50.000000        56.000000       53.000000      10.000000
75%        66.000000        69.000000       67.000000      13.000000
max        96.000000        95.000000       95.000000      94.000000

         gk_handling     gk_kicking  gk_positioning    gk_reflexes
count  183142.000000  183142.000000   183142.000000  183142.000000
mean       16.063612      20.998362       16.132154      16.441439
std        15.867382      21.452980       16.099175      17.198155
min         1.000000       1.000000        1.000000       1.000000
25%         8.000000       8.000000        8.000000       8.000000
```

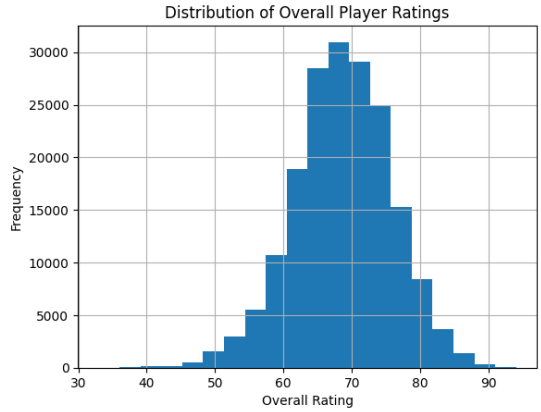
```
50%        11.000000      12.000000       11.000000      11.000000
75%        15.000000      15.000000       15.000000      15.000000
max        93.000000      97.000000       96.000000      96.000000

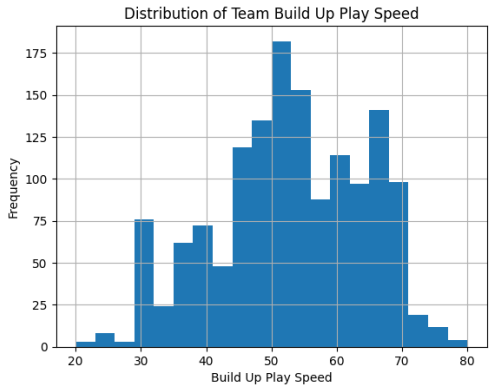
[8 rows x 38 columns]
                id  team_fifa_api_id    team_api_id  buildUpPlaySpeed  \
count  1458.000000       1458.000000    1458.000000       1458.000000
mean    729.500000      17706.982167    9995.727023         52.462277
std     421.032659      39179.857739   13264.869900         11.545869
min       1.000000          1.000000    1601.000000         20.000000
25%     365.250000        110.000000    8457.750000         45.000000
50%     729.500000        485.000000    8674.000000         52.000000
75%    1093.750000       1900.000000    9904.000000         62.000000
max    1458.000000     112513.000000  274581.000000         80.000000

       buildUpPlayDribbling  buildUpPlayPassing  chanceCreationPassing  \
count            489.000000         1458.000000            1458.000000
mean              48.607362           48.490398              52.165295
std                9.678290           10.896101              10.360793
min               24.000000           20.000000              21.000000
25%               42.000000           40.000000              46.000000
50%               49.000000           50.000000              52.000000
75%               55.000000           55.000000              59.000000
max               77.000000           80.000000              80.000000

       chanceCreationCrossing  chanceCreationShooting  defencePressure  \
count             1458.000000             1458.000000      1458.000000
mean                53.731824               53.969136        46.017147
std                 11.086796               10.327566        10.227225
min                 20.000000               22.000000        23.000000
25%                 47.000000               48.000000        39.000000
50%                 53.000000               53.000000        45.000000
75%                 62.000000               61.000000        51.000000
max                 80.000000               80.000000        72.000000

       defenceAggression  defenceTeamWidth
count        1458.000000       1458.000000
mean           49.251029         52.185871
std             9.738028          9.574712
min            24.000000         29.000000
25%            44.000000         47.000000
50%            48.000000         52.000000
75%            55.000000         58.000000
max            72.000000         73.000000
```





تحليل نتائج المباريات

تحليل نتائج المباريات الأساسية

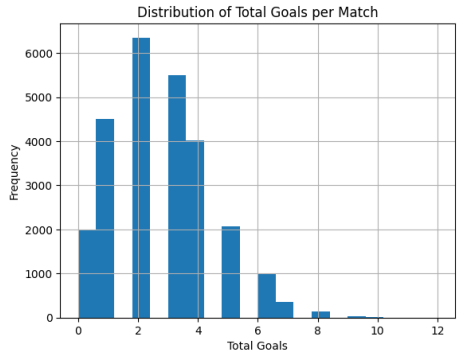
أولاً، نقوم بإجراء تحليل أساسي لنتائج المباريات من خلال فحص الإحصائيات الموجزة لعدد الأهداف التي سجلتها الفرق المضيفة والفرق الزائرة.

```
# Basic match outcome analysis
print(match[['home_team_goal', 'away_team_goal']].describe())

# Average goals per match
average_goals = match[['home_team_goal', 'away_team_goal']].mean().sum()
print(f'Average goals per match: {average_goals}')

# Distribution of match outcomes
match['total_goals'] = match['home_team_goal'] + match['away_team_goal']
match['total_goals'].hist(bins=20)
plt.title('Distribution of Total Goals per Match')
plt.xlabel('Total Goals')
plt.ylabel('Frequency')
plt.show()
```

```
home_team_goal  away_team_goal
count    25979.000000    25979.000000
mean         1.544594        1.160938
std          1.297158        1.142110
min          0.000000        0.000000
25%          1.000000        0.000000
50%          1.000000        1.000000
75%          2.000000        2.000000
max         10.000000        9.000000
Average goals per match: 2.7055313907386735
```



المصدر:

https://www.kaggle.com/code/ahmadelmorshedy/european-soccer-data-analysis

10) العثور على لاعبين مشابهين في فيفا 20 Finding Similar Players In FIFA20

في عالم تحليلات الرياضة التنافسية competitive sports analytics، أصبح التجميع والتحليل الاستراتيجي لبيانات اللاعبين أمرًا أساسيًا لتحديد المواهب، ووضع استراتيجيات اللعبة، واستكشاف الخصوم. تعمل مجموعة بيانات FIFA20، وهي مجموعة شاملة من إحصائيات اللاعبين من سلسلة ألعاب الفيديو الشهيرة FIFA، كمصدر بيانات مثالي لمثل هذه المساعي التحليلية. وهي تشمل مجموعة واسعة من المقاييس، بمافي ذلك السمات البدنية physical attributes، وتقييمات المهارات skill ratings، والبيانات الموضعية positional data لآلاف اللاعبين، مما يعكس نظرائهمفي الحياة الواقعية.

تكمن المشكلة المطروحةفي تطوير إطار منهجي يسمح بتحديد اللاعبين ذوي السمات والمجموعات المهارية المماثلة. وهذا له تطبيقات عمليةفي العديد من المجالات، مثل استكشاف اللاعبين ذوي الصفات المماثلة للاعب نجم معروف، أو العثور على بدائل محتملة لأعضاء الفريق المنتهية ولايته، أو تحليل الفرق المنافسة للاعبين ذوي أنماط لعب مماثلة. والهدف هو التنقل عبر مجموعة البيانات الغنية، وتطبيق تقنيات التعلم غير الخاضعة للإشراف unsupervised learning techniques، وإنشاء نظام موثوق وقابل للتكرار لمقارنة اللاعبين.

يهدف هذا التحليل إلى الاستفادة من البيانات الموجودةفي مجموعة بيانات FIFA20 لبناء نموذج يمكنه التنبؤ بتشابه اللاعبين. ومن خلال القيام بذلك، يسعى إلى تزويد محللي الرياضة ومخططي استراتيجيات الفريق برؤى قابلة للتنفيذ، وتمكينهم من اتخاذ قرارات مستنيرة بناءً على أدلة مدفوعة بالبيانات. سيتم تقييم النموذج على قدرته على تجميع اللاعبينفي مجموعات ذات مغزى باستخدام التجميع KMeans (KMeans clustering)، وتحديد أقرب الجيران للاعب معين بناءً على مقاييس التشابه المختلفة. سيستكشف المشروع بشكل أكبر فعالية هذه الأساليب، ومناقشة حدودها، واقتراح سبل محتملة للبحثفي المستقبل لتحسين وتعزيز القوة التنبؤية للنموذج.

مصدر البيانات

تم الحصول على مجموعة البيانات لهذا المشروع من Kaggle: https://www.kaggle.com/datasets/stefanoleone992/fifa-20-complete-player-dataset/data

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

استكشاف البيانات وتنظيفها (EDA)

دعنا نحمل مجموعة البيانات ونرى ما يمكن استخدامه للعثور على لاعبين مشابهين.

```
fifa20_players = pd.read_csv('/kaggle/input/fifa-20-complete-player-dataset/players_20.csv')
fifa20_players.head()
```

	sofifa_id	player_url	short_name	long_name	age	dob	height_cm	weight_kg	nationality	clu
0	158023	https://sofifa.com/player/158023/lionel-messi/...	L. Messi	Lionel Andrés Messi Cuccittini	32	1987-06-24	170	72	Argentina	FC Ba
1	20801	https://sofifa.com/player/20801/c-ronaldo-dos-...	Cristiano Ronaldo	Cristiano Ronaldo dos Santos Aveiro	34	1985-02-05	187	83	Portugal	Ju
2	190871	https://sofifa.com/player/190871/neymar-da-sil...	Neymar Jr	Neymar da Silva Santos Junior	27	1992-02-05	175	68	Brazil	Pa Sa Ge
3	200389	https://sofifa.com/player/200389/jan-oblak/20/...	J. Oblak	Jan Oblak	26	1993-01-07	188	87	Slovenia	Atl Ma
4	183277	https://sofifa.com/player/183277/eden-hazard/2...	E. Hazard	Eden Hazard	28	1991-01-07	175	74	Belgium	Re Ma

5 rows × 104 columns

```
fifa20_players.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18278 entries, 0 to 18277
Data columns (total 104 columns):
 #    Column                      Non-Null Count  Dtype
---   ------                      --------------  -----
 0    sofifa_id                   18278 non-null  int64
 1    player_url                  18278 non-null  object
 2    short_name                  18278 non-null  object
 3    long_name                   18278 non-null  object
 4    age                         18278 non-null  int64
 5    dob                         18278 non-null  object
 6    height_cm                   18278 non-null  int64
 7    weight_kg                   18278 non-null  int64
 8    nationality                 18278 non-null  object
 9    club                        18278 non-null  object
 10   overall                     18278 non-null  int64
 11   potential                   18278 non-null  int64
 12   value_eur                   18278 non-null  int64
 13   wage_eur                    18278 non-null  int64
 14   player_positions            18278 non-null  object
 15   preferred_foot              18278 non-null  object
 16   international_reputation    18278 non-null  int64
 17   weak_foot                   18278 non-null  int64
 18   skill_moves                 18278 non-null  int64
 19   work_rate                   18278 non-null  object
 20   body_type                   18278 non-null  object
 21   real_face                   18278 non-null  object
 22   release_clause_eur          16980 non-null  float64
 23   player_tags                 1499 non-null   object
 24   team_position               18038 non-null  object
 25   team_jersey_number          18038 non-null  float64
 26   loaned_from                 1048 non-null   object
```

```
27   joined                      16990 non-null  object
28   contract_valid_until        18038 non-null  float64
29   nation_position             1126 non-null   object
30   nation_jersey_number        1126 non-null   float64
31   pace                        16242 non-null  float64
32   shooting                    16242 non-null  float64
33   passing                     16242 non-null  float64
34   dribbling                   16242 non-null  float64
35   defending                   16242 non-null  float64
36   physic                      16242 non-null  float64
37   gk_diving                   2036 non-null   float64
38   gk_handling                 2036 non-null   float64
39   gk_kicking                  2036 non-null   float64
40   gk_reflexes                 2036 non-null   float64
41   gk_speed                    2036 non-null   float64
42   gk_positioning              2036 non-null   float64
43   player_traits               7566 non-null   object
44   attacking_crossing          18278 non-null  int64
45   attacking_finishing         18278 non-null  int64
46   attacking_heading_accuracy  18278 non-null  int64
47   attacking_short_passing     18278 non-null  int64
48   attacking_volleys           18278 non-null  int64
49   skill_dribbling             18278 non-null  int64
50   skill_curve                 18278 non-null  int64
51   skill_fk_accuracy           18278 non-null  int64
52   skill_long_passing          18278 non-null  int64
53   skill_ball_control          18278 non-null  int64
54   movement_acceleration       18278 non-null  int64
55   movement_sprint_speed       18278 non-null  int64
56   movement_agility            18278 non-null  int64
57   movement_reactions          18278 non-null  int64
58   movement_balance            18278 non-null  int64
59   power_shot_power            18278 non-null  int64
60   power_jumping               18278 non-null  int64
61   power_stamina               18278 non-null  int64
62   power_strength              18278 non-null  int64
63   power_long_shots            18278 non-null  int64
64   mentality_aggression        18278 non-null  int64
65   mentality_interceptions     18278 non-null  int64
66   mentality_positioning       18278 non-null  int64
67   mentality_vision            18278 non-null  int64
68   mentality_penalties         18278 non-null  int64
69   mentality_composure         18278 non-null  int64
70   defending_marking           18278 non-null  int64
71   defending_standing_tackle   18278 non-null  int64
72   defending_sliding_tackle    18278 non-null  int64
73   goalkeeping_diving          18278 non-null  int64
74   goalkeeping_handling        18278 non-null  int64
75   goalkeeping_kicking         18278 non-null  int64
76   goalkeeping_positioning     18278 non-null  int64
77   goalkeeping_reflexes        18278 non-null  int64
78   ls                          16242 non-null  object
79   st                          16242 non-null  object
80   rs                          16242 non-null  object
81   lw                          16242 non-null  object
82   lf                          16242 non-null  object
83   cf                          16242 non-null  object
84   rf                          16242 non-null  object
85   rw                          16242 non-null  object
86   lam                         16242 non-null  object
87   cam                         16242 non-null  object
88   ram                         16242 non-null  object
```

```
 89   lm                        16242 non-null  object
 90   lcm                       16242 non-null  object
 91   cm                        16242 non-null  object
 92   rcm                       16242 non-null  object
 93   rm                        16242 non-null  object
 94   lwb                       16242 non-null  object
 95   ldm                       16242 non-null  object
 96   cdm                       16242 non-null  object
 97   rdm                       16242 non-null  object
 98   rwb                       16242 non-null  object
 99   lb                        16242 non-null  object
 100  lcb                       16242 non-null  object
 101  cb                        16242 non-null  object
 102  rcb                       16242 non-null  object
 103  rb                        16242 non-null  object
dtypes: float64(16), int64(45), object(43)
memory usage: 14.5+ MB
```

تقدم مجموعة بيانات FIFA20، التي تتكون من 18278 إدخالاً و104 أعمدة، جدولاً غنيًا بنقاط البيانات التي تلخص عددًا لا يحصى من سمات اللاعبين. وفي حين أن مجموعة البيانات شاملة، فهناك حاجة واضحة للتنظيف والمعالجة المسبقة لضمان دقة وكفاءة التحليلات اللاحقة. تحتوي العديد من الأعمدة على عدد كبير من القيم الفارغة null values، مثل player_tags وloarded_from وnation_position والسمات الخاصة بحارس المرمى goalkeeper-specific attributes، والتي تنطبق فقط على مجموعة فرعية من اللاعبين. قد توفر أعمدة أخرى، مثل player_url وlong_name وdob، معلومات سياقية ولكنها ليست ذات صلة مباشرة بالنمذجة الإحصائية statistical modeling.

```
fifa20_players['cm'].unique()
```

```
array(['87+2', '81+3', '82+3', nan, '83+3', '87+3', '74+3', '78+3',
       '65+3', '79+3', '62+3', '77+3', '79+5', '80+3', '84+3', '85+3',
       '76+3', '75+3', '67+3', '74+2', '82+2', '66+2', '72+3', '83+2',
       '70+3', '71+3', '69+3', '62+2', '81+2', '78+2', '58+2', '72+2',
       '84+2', '55+3', '74+4', '75+2', '80+2', '73+2', '68+3', '64+3',
       '60+3', '73+3', '79+2', '77+2', '65+2', '69+2', '64+2', '70+2',
       '71+2', '63+2', '76+2', '66+3', '60+2', '67+2', '61+3', '61+2',
       '57+2', '59+2', '68+2', '53+3', '63+3', '57+3', '54+2', '56+2',
       '53+2', '52+2', '55+2', '51+2', '58+3', '50+2', '47+2', '49+2',
       '48+2', '45+2', '43+2', '44+2', '46+2', '41+2', '42+2', '40+2',
       '39+2', '37+2', '38+2', '35+2', '36+2', '34+2', '33+2', '31+2',
       '32+2'], dtype=object)
```

بالإضافة إلى ذلك، فإن السمات الموضعية positional attributes، بمافي ذلك cm وls وst وrs وlw وlf وغيرها، هي أنواع كائنات نظرًا لوجود سلاسل تمثل تقييمات اللاعبينفي مواضع مختلفة. ويجب تحويلها إلى تنسيق رقمي لتكون قابلة للاستخدامفي خوارزميات التعلم الآلي.

كما أن التحليل لا يحتاج حقًا إلى موضع لعب اللاعب المحدد بل إلى سماته فقط. لذا، سيتم دمج مواضع اللاعبينفي منطقة تشغيل محددة (على سبيل المثال، تعمل lw وrwفي الجناح). يساعد هذافي تقليل أبعاد البيانات.

```
position_cols = ['lw', 'rw', 'ls', 'st', 'rs', 'lf', 'cf', 'rf', 'lam',
'cam', 'ram', 'lcm', 'cm', 'rcm', 'lm', 'rm', 'ldm', 'cdm', 'rdm', 'lwb',
'rwb', 'lb', 'rb', 'lcb', 'cb', 'rcb']

def convert_position_rating(rating):
    if pd.isna(rating):
        return None
    else:
        parts = rating.split('+')
        return int(parts[0]) + int(parts[1])

for col in position_cols:
    fifa20_players[col] =
fifa20_players[col].apply(convert_position_rating)

# Combine positions into broader categories
fifa20_players['wing'] = fifa20_players[['lw', 'rw']].mean(axis=1)
fifa20_players['forward'] = fifa20_players[['ls', 'st', 'rs', 'lf', 'cf',
'rf']].mean(axis=1)
fifa20_players['attacking_midfield'] = fifa20_players[['lam', 'cam',
'ram']].mean(axis=1)
fifa20_players['midfield'] = fifa20_players[['lcm', 'cm',
'rcm']].mean(axis=1)
fifa20_players['wide_midfield'] = fifa20_players[['lm', 'rm']].mean(axis=1)
fifa20_players['defensive_midfield'] = fifa20_players[['ldm', 'cdm',
'rdm']].mean(axis=1)
fifa20_players['full_back'] = fifa20_players[['lwb', 'rwb', 'lb',
'rb']].mean(axis=1)
fifa20_players['defense'] = fifa20_players[['lcb', 'cb',
'rcb']].mean(axis=1)

fifa20_players.drop(columns=position_cols, inplace=True)
```

قبل الخوض في التحليل، من الأهمية بمكان تبسيط مجموعة البيانات من خلال تحديد الميزات التي تمثل بدقة سمات اللاعبين. تقدم مجموعة بيانات FIFA20 مجموعة واسعة من المقاييس، وينصب التركيز على تلك التي توفر رؤية شاملة لملف اللاعب.

- **معلومات أساسية عن اللاعب Basic Player Information:** تعتبر الميزات مثل العمر والطول والوزن أساسية، حيث يمكنها التأثير بشكل كبير على الأداء. يوفر تضمين short_name وclub معلومات يمكن التعرف عليها للتقييمات النوعية اللاحقة.
- **تصنيفات المهارات Skill Ratings:** تصنيفات المهارات الشاملة المتاحةفي مجموعة البيانات، والتي تغطي كل شيء من السرعة إلى المراوغة، لا غنى عنها لقياس قدرات اللاعب على أرض الملعب.
- **السمات العقلية والدفاعية Mental and Defensive Attributes:** السمات التي تعكس قوة اللاعب العقلية وبراعته الدفاعية، مثل العدوانية والاعتراض والهدوء، تقدم رؤى حول جوانب اللعب التي تتجاوز مجرد المقاييس البدنية أو المتعلقة بالمهارة.
- **تقييمات المواضع المجمعة Combined Position Ratings:** المواضع المجمعة بناءً على منطقة التشغيل من الخلية السابقة.

```
relevant_features = [
    # Basic Player Information
    'age', 'height_cm', 'weight_kg', 'short_name', 'club',

    # Skill Ratings
    'pace', 'shooting', 'passing', 'dribbling', 'defending', 'physic',
    'attacking_crossing', 'attacking_finishing',
'attacking_heading_accuracy',
    'attacking_short_passing', 'attacking_volleys',
    'skill_dribbling', 'skill_curve', 'skill_fk_accuracy',
'skill_long_passing', 'skill_ball_control',
    'movement_acceleration', 'movement_sprint_speed', 'movement_agility',
    'movement_reactions', 'movement_balance',
    'power_shot_power', 'power_jumping', 'power_stamina', 'power_strength',
'power_long_shots',
    'mentality_aggression', 'mentality_interceptions',
'mentality_positioning', 'mentality_vision',
    'mentality_penalties', 'mentality_composure',
    'defending_marking', 'defending_standing_tackle',
'defending_sliding_tackle',
    'work_rate',

    # Combined Position Ratings
    'wing', 'forward', 'attacking_midfield', 'midfield', 'wide_midfield',
'defensive_midfield', 'full_back', 'defense'
]

df = fifa20_players[relevant_features]
df
```

	age	height_cm	weight_kg	short_name	club	pace	shooting	passing	dribbling	defending	...	defending_slidi
0	32	170	72	L. Messi	FC Barcelona	87.0	92.0	92.0	96.0	39.0	...	26
1	34	187	83	Cristiano Ronaldo	Juventus	90.0	93.0	82.0	89.0	35.0	...	24
2	27	175	68	Neymar Jr	Paris Saint-Germain	91.0	85.0	87.0	95.0	32.0	...	29
3	26	188	87	J. Oblak	Atlético Madrid	NaN	NaN	NaN	NaN	NaN	...	18
4	28	175	74	E. Hazard	Real Madrid	91.0	83.0	86.0	94.0	35.0	...	22
...	...	...	...	...	...	...	...	...	...	...	...	...
18273	22	186	79	Shao Shuai	Beijing Renhe FC	57.0	23.0	28.0	33.0	47.0	...	52
18274	22	177	66	Xiao Mingjie	Shanghai SIPG FC	58.0	24.0	33.0	35.0	48.0	...	57
18275	19	186	75	Zhang Wei	Hebei China Fortune FC	54.0	35.0	44.0	45.0	48.0	...	46
18276	18	185	74	Wang Haijian	Shanghai Greenland Shenhua FC	59.0	35.0	47.0	47.0	45.0	...	54
18277	26	182	78	Pan Ximing	Hebei China Fortune FC	60.0	32.0	51.0	45.0	47.0	...	48

18278 rows × 49 columns

```
df.info(verbose=True, show_counts=True)
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 18278 entries, 0 to 18277
Data columns (total 49 columns):
 #   Column                       Non-Null Count  Dtype
---  ------                       --------------  -----
 0   age                          18278 non-null  int64
 1   height_cm                    18278 non-null  int64
 2   weight_kg                    18278 non-null  int64
 3   short_name                   18278 non-null  object
 4   club                         18278 non-null  object
 5   pace                         16242 non-null  float64
 6   shooting                     16242 non-null  float64
 7   passing                      16242 non-null  float64
 8   dribbling                    16242 non-null  float64
 9   defending                    16242 non-null  float64
 10  physic                       16242 non-null  float64
 11  attacking crossing           18278 non-null  int64
 12  attacking_finishing          18278 non-null  int64
 13  attacking_heading_accuracy   18278 non-null  int64
 14  attacking_short_passing      18278 non-null  int64
 15  attacking_volleys            18278 non-null  int64
 16  skill dribbling              18278 non-null  int64
 17  skill_curve                  18278 non-null  int64
 18  skill_fk_accuracy            18278 non-null  int64
 19  skill_long_passing           18278 non-null  int64
 20  skill_ball_control           18278 non-null  int64
 21  movement_acceleration        18278 non-null  int64
 22  movement_sprint_speed        18278 non-null  int64
 23  movement_agility             18278 non-null  int64
 24  movement_reactions           18278 non-null  int64
 25  movement_balance             18278 non-null  int64
 26  power_shot_power             18278 non-null  int64
 27  power_jumping                18278 non-null  int64
 28  power stamina                18278 non-null  int64
 29  power_strength               18278 non-null  int64
 30  power_long_shots             18278 non-null  int64
 31  mentality_aggression         18278 non-null  int64
 32  mentality_interceptions      18278 non-null  int64
 33  mentality_positioning        18278 non-null  int64
 34  mentality_vision             18278 non-null  int64
 35  mentality_penalties          18278 non-null  int64
 36  mentality_composure          18278 non-null  int64
 37  defending_marking            18278 non-null  int64
 38  defending_standing_tackle    18278 non-null  int64
 39  defending_sliding_tackle     18278 non-null  int64
 40  work_rate                    18278 non-null  object
 41  wing                         16242 non-null  float64
 42  forward                      16242 non-null  float64
 43  attacking_midfield           16242 non-null  float64
 44  midfield                     16242 non-null  float64
 45  wide_midfield                16242 non-null  float64
 46  defensive_midfield           16242 non-null  float64
 47  full_back                    16242 non-null  float64
 48  defense                      16242 non-null  float64
dtypes: float64(14), int64(32), object(3)
memory usage: 6.8+ MB
```

من الواضح أن هناك بعض القيم المفقودة للسمات والقيم المفقودة هي بالضبط عدد حراس المرمى. وهذا يعني أن حراس المرمى فقط لديهم قيم مفقودة لسمات الملعب الخارجي. وبسبب هذا، وبما أن

هدف هذا المشروع هو العثور على لاعبين مشابهينفي الملعب الخارجي، فيمكن استبعاد جميع اللاعبين الذين لديهم قيم فارغةفي سمات الملعب الخارجي الحاسمة (وهو ما سيؤديفي الأساس إلى استبعاد جميع حراس المرمى).

```
outfield_players = df.dropna(subset=['pace', 'shooting', 'passing',
'dribbling', 'defending', 'physic'])
outfield_players.info(verbose=True)
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 16242 entries, 0 to 18277
Data columns (total 49 columns):
 #   Column                        Non-Null Count  Dtype
---  ------                        --------------  -----
 0   age                           16242 non-null  int64
 1   height_cm                     16242 non-null  int64
 2   weight_kg                     16242 non-null  int64
 3   short_name                    16242 non-null  object
 4   club                          16242 non-null  object
 5   pace                          16242 non-null  float64
 6   shooting                      16242 non-null  float64
 7   passing                       16242 non-null  float64
 8   dribbling                     16242 non-null  float64
 9   defending                     16242 non-null  float64
 10  physic                        16242 non-null  float64
 11  attacking_crossing            16242 non-null  int64
 12  attacking_finishing           16242 non-null  int64
 13  attacking_heading_accuracy    16242 non-null  int64
 14  attacking_short_passing       16242 non-null  int64
 15  attacking_volleys             16242 non-null  int64
 16  skill_dribbling               16242 non-null  int64
 17  skill_curve                   16242 non-null  int64
 18  skill_fk_accuracy             16242 non-null  int64
 19  skill_long_passing            16242 non-null  int64
 20  skill_ball_control            16242 non-null  int64
 21  movement_acceleration         16242 non-null  int64
 22  movement_sprint_speed         16242 non-null  int64
 23  movement_agility              16242 non-null  int64
 24  movement_reactions            16242 non-null  int64
 25  movement_balance              16242 non-null  int64
 26  power_shot_power              16242 non-null  int64
 27  power_jumping                 16242 non-null  int64
 28  power_stamina                 16242 non-null  int64
 29  power_strength                16242 non-null  int64
 30  power_long_shots              16242 non-null  int64
 31  mentality_aggression          16242 non-null  int64
 32  mentality_interceptions       16242 non-null  int64
 33  mentality_positioning         16242 non-null  int64
 34  mentality_vision              16242 non-null  int64
 35  mentality_penalties           16242 non-null  int64
 36  mentality_composure           16242 non-null  int64
 37  defending_marking             16242 non-null  int64
 38  defending_standing_tackle     16242 non-null  int64
 39  defending_sliding_tackle      16242 non-null  int64
 40  work_rate                     16242 non-null  object
 41  wing                          16242 non-null  float64
 42  forward                       16242 non-null  float64
 43  attacking_midfield            16242 non-null  float64
 44  midfield                      16242 non-null  float64
 45  wide_midfield                 16242 non-null  float64
```

```
 46  defensive_midfield          16242 non-null  float64
 47  full_back                   16242 non-null  float64
 48  defense                     16242 non-null  float64
dtypes: float64(14), int64(32), object(3)
memory usage: 6.2+ MB
```

```
outfield_players.describe().T
```

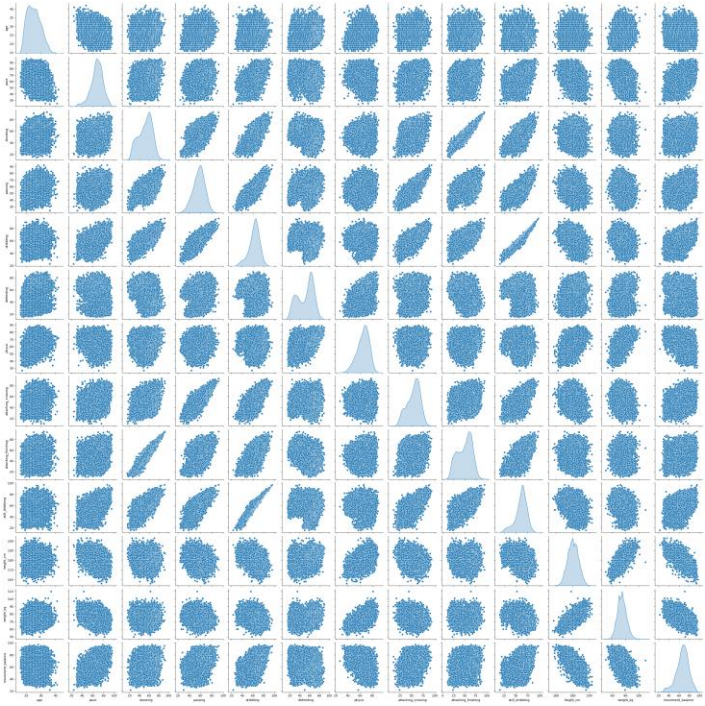
	count	mean	std	min	25%	50%	75%	max
age	16242.0	25.160017	4.544366	16.0	21.0	25.0	28.0	42.0
height_cm	16242.0	180.475249	6.454725	156.0	176.0	180.0	185.0	203.0
weight_kg	16242.0	74.438493	6.692058	50.0	70.0	74.0	79.0	110.0
pace	16242.0	67.700899	11.297656	24.0	61.0	69.0	75.0	96.0
shooting	16242.0	52.298301	14.029418	15.0	42.0	54.0	63.0	93.0
passing	16242.0	57.233777	10.407844	24.0	50.0	58.0	64.0	92.0
dribbling	16242.0	62.531585	10.284950	23.0	57.0	64.0	69.0	96.0
defending	16242.0	51.553503	16.419528	15.0	36.0	56.0	65.0	90.0
physic	16242.0	64.876678	9.760162	27.0	59.0	66.0	72.0	90.0
attacking_crossing	16242.0	54.180766	14.044269	11.0	44.0	56.0	65.0	93.0
attacking_finishing	16242.0	49.851188	16.340531	10.0	36.0	52.0	63.0	95.0
attacking_heading_accuracy	16242.0	56.995382	11.638265	12.0	49.0	58.0	65.0	93.0
attacking_short_passing	16242.0	62.728543	9.619125	23.0	57.0	64.0	69.0	92.0
attacking_volleys	16242.0	46.674116	14.700085	10.0	35.0	46.0	58.0	90.0
skill_dribbling	16242.0	60.856360	12.346669	16.0	55.0	63.0	69.0	97.0
skill_curve	16242.0	51.434491	15.094479	11.0	40.0	52.0	63.0	94.0
skill_fk_accuracy	16242.0	46.282478	14.994287	10.0	34.0	44.0	58.0	94.0
skill_long_passing	16242.0	56.136929	12.301053	19.0	48.0	58.0	65.0	92.0
skill_ball_control	16242.0	63.322189	9.952176	24.0	58.0	64.0	70.0	96.0
movement_acceleration	16242.0	67.635821	11.800788	20.0	61.0	68.0	76.0	97.0
movement_sprint_speed	16242.0	67.742950	11.548859	25.0	62.0	69.0	75.0	96.0
movement_agility	16242.0	66.478759	12.281811	23.0	59.0	68.0	75.0	96.0
movement_reactions	16242.0	62.152198	8.807388	31.0	56.0	62.0	68.0	96.0
movement_balance	16242.0	66.520256	12.174540	22.0	60.0	68.0	75.0	97.0
power_shot_power	16242.0	59.644994	13.290262	14.0	51.0	61.0	69.0	95.0
power_jumping	16242.0	65.876986	11.627546	25.0	59.0	67.0	74.0	95.0
power_stamina	16242.0	66.993720	11.404695	29.0	60.0	68.0	75.0	97.0
power_strength	16242.0	65.830378	12.568418	20.0	58.0	67.0	75.0	97.0
power_long_shots	16242.0	51.147149	15.782163	11.0	39.0	54.0	63.0	94.0
mentality_aggression	16242.0	59.475988	14.326789	10.0	50.0	61.0	70.0	95.0
mentality_interceptions	16242.0	50.121229	18.856113	10.0	33.0	55.0	65.0	92.0
mentality_positioning	16242.0	54.989780	14.582565	11.0	47.0	57.0	65.0	95.0
mentality_vision	16242.0	55.558490	12.819629	12.0	47.0	57.0	65.0	94.0
mentality_penalties	16242.0	51.998338	12.437148	11.0	42.0	52.0	61.0	92.0
mentality_composure	16242.0	60.460781	10.212126	27.0	53.0	61.0	68.0	96.0

defending_marking	16242.0	50.879571	17.443311	10.0	36.0	55.0	65.0	94.0
defending_standing_tackle	16242.0	51.863995	19.047749	10.0	35.0	58.0	67.0	92.0
defending_sliding_tackle	16242.0	49.590445	19.046176	10.0	32.0	56.0	65.0	90.0
wing	16242.0	61.105036	9.882466	27.0	55.0	62.0	68.0	95.0
forward	16242.0	60.327669	9.456421	30.5	54.0	61.0	67.0	93.5
attacking_midfield	16242.0	61.069388	9.765853	29.0	55.0	62.0	68.0	95.0
midfield	16242.0	60.261852	8.833362	33.0	54.0	61.0	66.0	90.0
wide_midfield	16242.0	61.784694	9.253605	29.0	56.0	63.0	68.0	94.0
defensive_midfield	16242.0	58.752555	10.180389	30.0	51.0	60.0	66.0	90.0
full_back	16242.0	59.053442	9.292749	32.0	52.5	59.5	66.0	87.5
defense	16242.0	57.494767	11.793411	28.0	48.0	59.0	67.0	90.0

لقد تم بنجاح تقطير distilled البيانات الأصلية لتحديد السمات، مما أدى إلى تقليص مجموعة السمات من 104 إلى مجموعة فرعية أكثر قابلية للإدارة تتكون من 49 سمة ذات صلة. ومع ذلك، لا يزال هذا العدد يمثل تحديًا من حيث القدرة على التفسير والتصور.

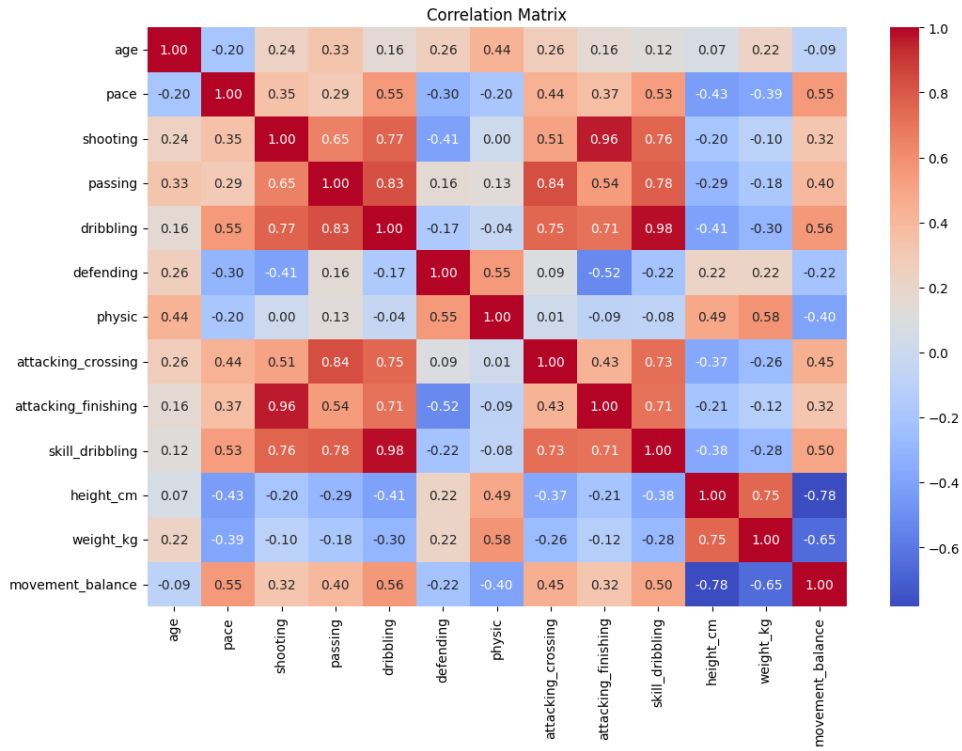
```
features_for_pairplot = ['age', 'pace', 'shooting', 'passing', 'dribbling',
                         'defending', 'physic', 'attacking_crossing',
                         'attacking_finishing', 'skill_dribbling',
                         'height_cm', 'weight_kg', 'movement_balance']

plt.figure(figsize=(15, 15))
sns.pairplot(outfield_players[features_for_pairplot], diag_kind='kde')
plt.show()
```



كما هو موضحفي الرسم البياني الزوجي pair plot، يمكن أن تكون العلاقات المتبادلة بين السمات معقدة بصريًا، مع الحاجة إلى العديد من الرسوم البيانية المتناثرة scatter plots لمقارنة كل سمة على حدة. توفر مصفوفة الرسوم البيانية هذه نظرة عامة قيمة حول كيفية ارتباط سمات اللاعب المختلفة ببعضها البعض، وكشف الأنماط والتجمعات المحتملة داخل البيانات. ومع ذلك، فإن كثافة المعلومات يمكن أن تجعل من الصعب استخلاص استنتاجات فورية وواضحة.

```
plt.figure(figsize=(12, 8))
corr_matrix = outfield_players[features_for_pairplot].corr()
sns.heatmap(corr_matrix, annot=True, fmt='.2f', cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```



توجد ارتباطات إيجابية بشكل ملحوظ بين مهارات التصويب shooting والتمرير passing والمراوغة dribbling، مما يشير إلى أن اللاعبين غالبًا ما يتفوقونفي هذه المجالاتفي نفس الوقت. وعلى النقيض من ذلك، يعكس الارتباط السلبي بين السمات الدفاعية والمهارات الهجومية التخصص بين الأدوار الهجومية والدفاعية للاعبين. يُظهر العمر علاقة عكسية مع السرعة، مما يشير إلى أن السرعة تتضاءل مع تقدم العمر،في حين يشير الارتباط الإيجابي مع السمة البدنية إلى أن القوة البدنية قد تزيد. يرتبط

الطول والوزن بشكل إيجابي، مما يتماشى مع التوقعات البدنية، ومع ذلك يُظهر كلاهما علاقة سلبية مع التوازن، مما يشير إلى حل وسط بين الحجم وخفة الحركة.

يُبرر تحليل المكونات الأساسية Principal Component Analysis (PCA) لتقليل الأبعاد ومعالجة التعدد الخطي multicollinearity وتحسين قابلية تفسير وتصور مجموعة البيانات. سيحول تحليل المكونات الأساسية مجموعة البيانات إلى شكل مبسط، ويلتقط أهم ميزات أداء اللاعب ويسمح بنمذجة أكثر فعالية.

```
from sklearn.preprocessing import StandardScaler

numerical_features = outfield_players.select_dtypes(include=['int64',
'float64']).columns

scaler = StandardScaler()
outfield_players_standardized =
scaler.fit_transform(outfield_players[numerical_features])
```

```
from sklearn.decomposition import PCA

pca = PCA(n_components=0.95)  # Keeping 95% of the variance
principal_components = pca.fit_transform(outfield_players_standardized)

pca_df = pd.DataFrame(data=principal_components, columns=['PC' + str(i) for
i in range(1, pca.n_components_ + 1)])

original_shape = outfield_players_standardized.shape
transformed shape = pca df.shape

(original_shape, transformed_shape)
```

```
((16242, 46), (16242, 17))
```

يُلاحظ أن تحليل المكونات الأساسية قد اختصر مجموعة الميزات الأصلية إلى 17 مكونًا رئيسيًا مع الاحتفاظ بالمكونات التي تفسر 95% من التباينفي مجموعة البيانات. ومن المفترض أن تساعد هذه المجموعة المختصرة من الميزاتفي الكشف عن الأنماطفي البيانات بشكل أكثر فعالية وبضوضاء أقل مقارنة بالمجموعة الكاملة من الميزات الأصلية.

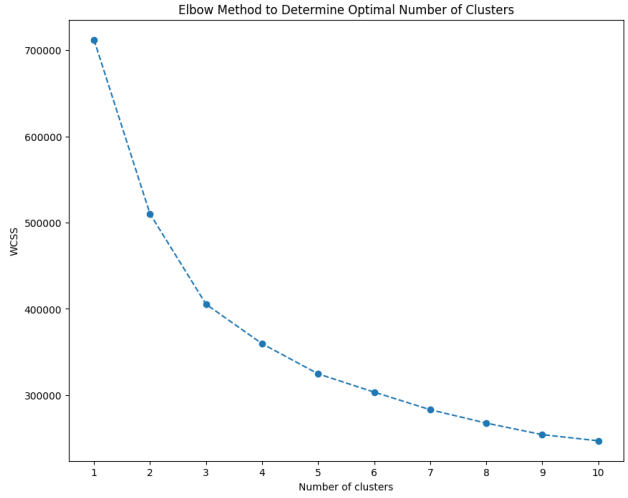
تجميع K-Means

تم بعد ذلك استخدام تجميع K-Means على مجموعة البيانات المحولة هذه لتقسيم اللاعبين إلى مجموعات مميزة. يمثل كل مجموعة نظريًا لاعبين لديهم سمات متشابهة، حيث يهدف K-Means إلى تقليل التباين داخل المجموعة. تم تحديد عدد المجموعات number of clusters، وهو أحد المعلمات الفائقة hyperparameter المهمةفي K-Means، من خلال طريقة Elbow، والتي وفرت نهجًا قائمًا على البيانات لتحديد عدد مناسب لمجموعة البيانات الموجودة.

```
from sklearn.cluster import KMeans
```

```
wcss = []
for i in range(1, 11):  # Let's test for 1 to 10 clusters
    kmeans = KMeans(n_clusters=i, random_state=42, n_init='auto')
    kmeans.fit(pca_df)
    wcss.append(kmeans.inertia_)

plt.figure(figsize=(10, 8))
plt.plot(range(1, 11), wcss, marker='o', linestyle='--')
plt.title('Elbow Method to Determine Optimal Number of Clusters')
plt.xlabel('Number of clusters')
plt.ylabel('WCSS')  # Within-cluster sum of squares
plt.xticks(range(1, 11))
plt.show()
```



استنادًا إلى الرسم البياني، تنخفض قيم WCSS (مجموع المربعات داخل المجموعة) بسرعة حتى نقطة معينة، وبعدها يتباطأ معدل الانخفاض. هذه النقطة، التي يبدو أنها تقع عند k=4، هي المكان الذي يُعتقد عادةً أنه يقع فيه الكوع elbow. عند k=4، لا يؤدي إضافة المزيد من المجموعات إلى انخفاض كبيرفي WCSS كما فعلت المجموعات السابقة، مما يشير إلى تناقص العائدات على جودة المجموعة.

وبالتالي، فإن k=4 ستكون نقطة بداية جيدة لعدد المجموعات.

```
kmeans_optimal = KMeans(n_clusters=4, random_state=42, n_init='auto')
kmeans_clusters_optimal = kmeans_optimal.fit_predict(pca_df)

# Adding the optimal cluster labels to the PCA DataFrame
pca_df['Optimal_KMeans_Cluster'] = kmeans_clusters_optimal
```

```
player_name = 'L. Messi'
player_index = outfield_players[outfield_players['short_name'] ==
player_name].index[0]
target_player_cluster = pca_df.loc[player_index, 'Optimal_KMeans_Cluster']
```

```
similar_player_indices = pca_df[pca_df['Optimal_KMeans_Cluster'] ==
target_player_cluster].index
similar_players = outfield_players.iloc[similar_player_indices]
similar_players = similar_players[similar_players['short_name'] !=
player_name]

print(f"Players similar to {player_name} based on KMeans clustering:")
similar_players[['short_name', 'age', 'club']]
```

اللاعبون المشابهون لـ L. Messi استنادًا إلى مجموعة KMeans:

	short_name	age	club
1	Cristiano Ronaldo	34	Juventus
2	Neymar Jr	27	Paris Saint-Germain
4	E. Hazard	28	Real Madrid
5	K. De Bruyne	28	Manchester City
8	L. Modrić	33	Real Madrid
...	...	...	...
13029	L. Nizzetto	33	Virtus Entella
13063	Y. Iwakami	29	Matsumoto Yamaga
13072	Felipe Soldivia	31	Ceará Sporting Club
13880	F. Torsteinbø	28	Viking FK
14428	D. Serna	25	Colorado Rapids

3220 rows × 3 columns

الهدف من هذا النموذج هو تقليل التباين داخل كل مجموعة، مما يؤدي إلى مجموعة من المجموعات التي تجمع اللاعبين حسب التشابه العامفي السمات.في حالة تحديد اللاعبين المشابهين لليونيل ميسي، وضعت مجموعة KMeans ميسيفي مجموعة واسعة تضم 4012 لاعبًا. يعد هذا النهج الواسع النطاق مفيدًا للتجزئة عالية المستوى وهو فعال حسابيًا لمجموعات البيانات الكبيرة. ومع ذلك، فإنه يفتقر إلى الدقة المطلوبة لمقارنات التشابه المحددة لأنه لا يأخذفي الاعتبار قرب اللاعبين الفرديين من اللاعب المستهدف بل بالأحرى قربهم من مركز المجموعة. وهذا مقيد بشكل خاص عندما تكون المقارنة الدقيقة والمباشرة مطلوبة، حيث لا يزال اللاعبون داخل نفس المجموعة يُظهِرون تباينًا كبيرًافي سماتهم.

أقرب الجيران

لننتقل إلى تطبيق خوارزمية أقرب الجيران Nearest Neighbors algorithm على مجموعة البيانات المخفضة باستخدام تحليل المكونات الأساسية PCA لتحديد اللاعبين الذين لديهم أقرب قربفي مساحة الميزة للاعب معين. وعلى عكس خوارزمية K-Means، التي تقسم مجموعة البيانات بالكامل إلى مجموعات أوسع، تركز تقنية أقرب الجيران على العلاقات الفردية، وتحديد اللاعبين الأكثر تشابهًا استنادًا إلى متجهات الميزة الخاصة بهم.

```
from sklearn.neighbors import NearestNeighbors

nn_model = NearestNeighbors(n_neighbors=10, algorithm='ball_tree')
nn_model.fit(pca_df.values)
```

```
NearestNeighbors
NearestNeighbors(algorithm='ball_tree', n_neighbors=10)
```

```
# Finding players like Messi
player_name = 'L. Messi'
player_index = outfield_players.loc[outfield_players['short_name'] ==
player_name].index[0]
player_pca = pca_df.iloc[player_index].values.reshape(1, -1)

distances, indices = nn_model.kneighbors(player_pca)
similar_player_indices = indices.flatten()
similar_players = outfield_players.iloc[similar_player_indices]

print(f"Players similar to {player_name} based on NearestNeighbors:")
similar_players[['short_name', 'age', 'club']]
```

اللاعبون المشابهون لـ L. Messi استنادًا إلى NearestNeighbors:

	short_name	age	club
0	L. Messi	32	FC Barcelona
4	E. Hazard	28	Real Madrid
2	Neymar Jr	27	Paris Saint-Germain
37	M. Reus	30	Borussia Dortmund
23	P. Dybala	25	Juventus
9	M. Salah	27	Liverpool
43	H. Son	26	Tottenham Hotspur
17	S. Agüero	31	Manchester City
22	A. Griezmann	28	FC Barcelona
66	A. Di María	31	Paris Saint-Germain

على عكس KMeans، لا يهتم برنامج Nearest Neighbors بتحسين البُنية العامة ولكنه يركز على التشابه المحلي، وقياس المسافة الإقليدية بين اللاعب المستهدف وجميع اللاعبين الآخرين. وقد أسفر هذا النهج عن مجموعة موجزة وذات صلة عالية من اللاعبين الذين يعتبرون الأكثر تشابهًا مع ميسي، مثل إيدن هازارد ونيمار جونيور. تكمن قوة برنامج Nearest Neighborsفي قدرته على العثور على أقرب مجموعة فرعية من نقاط البيانات، مما يجعله مناسبًا بطبيعته للمقارنات الفردية.

المناقشة والمقارنة

إن قدرة KMeans على تقسيم البيانات إلى عدد محدد مسبقًا من المجموعات ترتكز على هدفها المتمثلفي تقليل التباين داخل المجموعة. هذه الطريقة فعالة للغاية لمجموعات البيانات الكبيرة وهي فعالةفي تقسيم البيانات إلى مجموعات مميزة. ومع ذلك، فإن تطبيقهافي سياق العثور على لاعب مشابه لميسي أدى إلى مجموعة واسعة تضم 4012 لاعبًا. وهذا يشير إلى أنهفي حين أن KMeans قيمة لتجزئة السوق أو تجميع اللاعبينفي طبقات عامة، إلا أنها تفشل عندما تكون الدقة مطلوبة. غالبًا ما يتجاهل

اعتمادها على بُنية المتوسط الفروق الدقيقة لنقاط البيانات الفردية، مما يؤدي إلى مجموعات قد تحتوي على مجموعة متنوعة من سمات اللاعب.

على العكس من ذلك، تتميز خوارزمية أقرب الجيران بالدقة. من خلال حساب المسافة الإقليدية بين نقاط البيانات، فإنها تحدد أقرب اللاعبين إلى اللاعب المستهدففي مساحة الميزة. هذه الخصوصية مفيدة عند البحث عن مجموعة فرعية صغيرة من اللاعبين المتشابهين للغاية. أظهرت القائمة الناتجة، والتي تضمنت لاعبين مثل إيدن هازارد ونيمار جونيور، اختيارًا أكثر تمييزًا وأهمية. ومع ذلك، تتناسب كثافة الحسابفي النموذج مع عدد الميزات ونقاط البيانات، مما قد يصبح محظورًا مع البيانات عالية الأبعاد. تم التخفيف من هذا القيد من خلال المعالجة المسبقة للبيانات باستخدام تحليل المكونات الأساسية PCA، وبالتالي تعزيز التطبيق العملي لخوارزمية أقرب الجيران.

يسلط التباين بين KMeans وأقرب الجيران الضوء على المقايضة بين الاتساع breadth والدقة precision. KMeans أكثر ملاءمة للتحليل الاستكشافي حيث يكون الهدف هو فهم البُنية الأوسع للبيانات. على النقيض من ذلك، تم تصميم أقرب الجيران للاستعلامات المستهدفة حيث تكون المقارنات المحددة ذات أهمية قصوى. هذا التمييز أمر بالغ الأهمية عند النظرفي تطبيق كل نموذج. يمكن أن يكون KMeans هو النموذج المفضل لاستكشاف البيانات الأولية أو عند تجميع السماتفي فئات أوسع، بينما يفضل أقرب الجيران للتوصيات الشخصية أو تحليل اللاعبين التفصيلي.

الاستنتاج

يجب أن يتحدد قرار استخدام KMeans أو أقرب الجيران وفقًا للأهداف التحليلية المطروحة. إذا كان الهدف هو تصنيف اللاعبينفي مجموعات عامة للتحليل الأولي، فإن KMeans يخدم بشكل جيد. ولكن عندما تتطلب المهمة تحديد اللاعبين الأقرب شبهاً بملف تعريف معين، فإن أقرب الجيران هو الخيار الأمثل، خاصة عندما يتم استباق ذلك بواسطة تحليل المكونات الأساسية لتقليل الأبعاد والمتطلبات الحسابية. تؤكد هذه المقارنة على أهمية مواءمة اختيار النموذج مع الاحتياجات التحليلية المحددة، وموازنة نطاق التحليل مع الحبيبات المطلوبة للرؤى القابلة للتنفيذ.

القيود

يواجه تحليل مجموعة بيانات FIFA20 لتشابه اللاعبين العديد من القيود.

1. قد لا يعكس تجميع KMeans التوزيعات المعقدة لسمات اللاعبين بسبب افتراضه للمجموعات الكروية.
2. تعتمد فعالية خوارزمية Nearest Neighborsعلى مقياس المسافة المختار ومقياس الميزة، والذي قد لا يتماشى مع أهمية مهارات كرة القدم معينة.

3. على الرغم من أن تحليل المكونات الرئيسية يقلل من الأبعاد، إلا أنه يتجاهل العلاقات غير الخطية المهمة والجوانب الديناميكية لأداء اللاعب. علاوة على ذلك، تفشل الطبيعة الثابتة للبياناتفي التقاط التقدم الزمني لقدرات اللاعبين.
4. قد لا تتضمن مجموعة البيانات نفسها، المأخوذة من لعبة فيديو، جميع عناصر أداء اللاعب في الحياة الواقعية، مثل العوامل النفسية ومستويات اللياقة البدنية، والتي تعد حاسمة لتأثيره على أرض الملعب.

ستستفيد التحسينات المستقبلية من معالجة هذه القيود من خلال دمج بيانات أكثر ديناميكية وطرق تحليلية متقدمة.

التوصيات

بالنسبة للتحسينات المستقبلية لنموذج تشابه اللاعب ضمن مجموعة بيانات FIFA20، يمكن للباحثين استكشاف مجموعة متنوعة من الأساليب. يمكن أن توفر هندسة الميزات feature engineering المتقدمة لتشمل مقاييس الأداء التفصيلية مقارنة أكثر دقة بين اللاعبين. يمكن أن يؤدي تحسين خوارزميات التجميع من خلال ضبط المعلمات الفائقة إلى تحسين التجميعات لزيادة الدقة. إن دمج بيانات الأداء التاريخي للاعبين من شأنه أن يسمح بإجراء تحليل ديناميكي يأخذفي الاعتبار تطور مسيرة اللاعب. ويمكن أن يضمن التحقق من صحة نتائج النموذج مع خبراء صناعة كرة القدم مدى ملاءمتها ودقتها العملية. بالإضافة إلى ذلك، قد يؤدي التجريب باستخدام خوارزميات التجميع المتطورة وتقنيات التعلم العميق إلى الكشف عن أنماط وعلاقات معقدة لا يمكن رؤيتها على الفور من خلال الأساليب التقليدية، مما يؤدي إلى فهم أعمق لأوجه التشابه بين اللاعبين وقدراتهم.

المصدر:

https://www.kaggle.com/code/grapeseed/finding-similar-players

11) تحليل الرياضيين الأعلى أجراً باستخدام بايثون Highest-Paid Athletes Analysis with Python

أحد أكبر الأسباب التي تجعل الرياضيين يكسبون الكثير من المال هو أننا نحب مشاهدة مبارياتهم.في هذه المقالة، سأطلعك على مشروع علم البيانات حول تحليل الرياضيين الأعلى أجراً Highest-Paid Athletes Analysis باستخدام بايثون.

استيراد المكتبات

سأبدأ مهمة تحليل الرياضيين الأعلى أجراً باستيراد مكتبات بايثون الضرورية ومجموعة البيانات:

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
import seaborn as sns
import matplotlib.pyplot as plt
sns.set(style="darkgrid")
plt.style.use("seaborn-pastel")

df = pd.read_excel("Forbes Athlete List 2012-2019.xlsx")
df.head()
```

	Rank	Name	Pay	Salary/Winnings	Endorsements	Sport	Year
0	#1	Lionel Messi	$127 M	$92 M	$35 M	Soccer	2019
1	#2	Cristiano Ronaldo	$109 M	$65 M	$44 M	Soccer	2019
2	#3	Neymar	$105 M	$75 M	$30 M	Soccer	2019
3	#4	Canelo Alvarez	$94 M	$92 M	$2 M	Boxing	2019
4	#5	Roger Federer	$93.4 M	$7.4 M	$86 M	Tennis	2019

لذا، تحتوي مجموعة البيانات على 7 أعمدة و795 صفًا، دعني أصف البيانات باختصار:

- Rank: الترتيب السنوي بناءً على الراتب.
- Name: اسم الرياضي.
- Salary: الراتب والتأييد قابلان للدفع.
- Salary / Winnings: راتب الرياضي.
- Endorsements: الإيرادات من الإعلانات ووسائل التواصل الاجتماعي والرعاة وما إلى ذلك.
- Sport: نوع الرياضة التي يمارسها الرياضي.

- Year: سنة الرواتب.

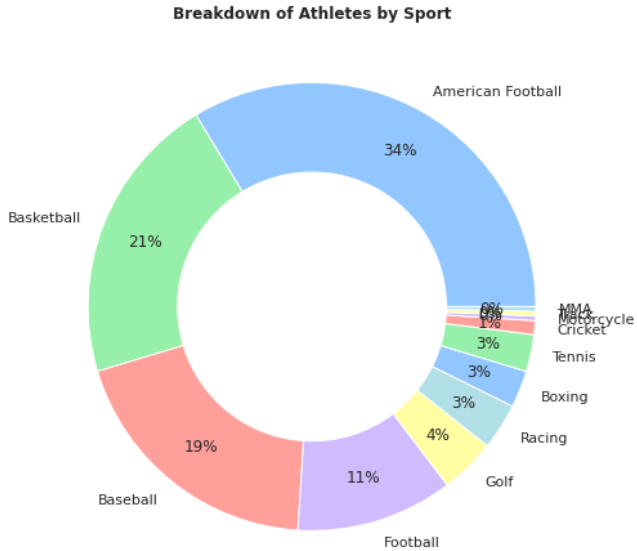
مجموعة البيانات التي نستخدمها مأخوذة من مجلة فوربس Forbes. بعض الأعمدة ليست متسقة عبر مجموعة البيانات لأن مجلة فوربس غيرت رأيها بشأن ما إذا كان ينبغي وضع "#" قبل قيمة المرتبة rank بمرور الوقت. دعنا نصلح هذا الأمر ونزيل "علامات الدولار dollar signs" و"M". دعنا أيضًا نغير "Football" إلى "Football" و"Football" إلى "American Football":

```
df.Rank = df.Rank.apply(lambda x: int(x.split("#")[1]) if type(x) == np.str
else x)
df.Pay = df.Pay.apply(lambda x: float(x.split(" ")[0].split("$")[1]))
df.Endorsements = df.Endorsements.apply(lambda x: float(x.split("
")[0].split("$")[1]))
df["Salary/Winnings"].replace("-",'$nan M',inplace=True)
df["Salary/Winnings"] = df["Salary/Winnings"].apply(lambda x:
float(x.split(" ")[0].split("$")[1]))
df.Sport.replace({"Soccer":"Football",
                  "Football":"American Football",
                 "Mixed Martial Arts":"MMA",
                 "Auto racing":"Racing",
                  "Auto Racing":"Racing",
                  "Basketbal":"Basketball",
                 },inplace=True)

df.columns = ['Rank', 'Name', 'Pay', 'Salary_Winnings', 'Endorsements',
'Sport', 'Year']
```

الآن دعونا نرى توزيع الرياضيينفي مجموعة البيانات بناءً على نوع الرياضة التي يمارسونها:

```
df.groupby("Name").first()["Sport"].value_counts().plot(kind="pie",autopct=
"%.0f%%",figsize=(8,8),wedgeprops=dict(width=0.4),pctdistance=0.8)
plt.ylabel(None)
plt.title("Breakdown of Athletes by Sport",fontweight="bold")
plt.show()
```



لنرسم إجمالي رواتب الرياضيينفي رسوم متحركة لشريط السباق running bar animation. أولاً، سنحول عمود السنة إلى كائن تاريخ ووقت DateTime object:

```
df.Year = pd.to_datetime(df.Year,format="%Y")
```

بعد ذلك، قم بإعداد جدول محوري pivot table حيث تكون الأعمدة عبارة عن الرياضيين ويكون المؤشر عبارة عن السنوات:

```
racing_bar_data = df.pivot_table(values="Pay",index="Year",columns="Name")
```

الرياضيون المذكورون أدناه هم الوحيدون الذين تم تضمينهم باستمرارفي قائمة أفضل 100 لاعبفي كل عام منذ عام 2012. أما بقية الرياضيين فلديهم قيم NaN. سنقوم أولاً باستيفاء قيم NaN بشكل خطي واستخدام ملء قيم NaN المتبقية عن طريق إعادة التعبئة:

```
racing_bar_data.columns[racing_bar_data.isnull().sum() == 0]
```

الآن قم بتحويل البيانات إلى مجموع الرواتب التراكمي على مدى عدة سنوات:

```
racing_bar_filled =
racing_bar_data.interpolate(method="linear").fillna(method="bfill")
racing_bar_filled = racing_bar_filled.cumsum()
```

الآن، دعنا نأخذ عينات إضافية من مجموعة البيانات باستخدام الاستيفاء (الخطي) interpolation (linear) لتحقيق انتقال سلسفي إطارات الرسوم المتحركة:

```
racing_bar_filled =
racing_bar_filled.resample("1D").interpolate(method="linear")[::7]
```

الآن دعنا نستورد حزم بايثون اللازمة لإنشاء الرسوم المتحركة وحفظها، وتشغيل المسارات وعناصرها (الخطوط والأشرطة والنصوص وما إلى ذلك). سيعمل الكود أدناه على إنشاء رسوم متحركة لأعلى 10 رياضيين أجراً بين عامي 2012 و2019:

```
from matplotlib.animation import FuncAnimation, FFMpegWriter

selected  = racing_bar_filled.iloc[-
1,:].sort_values(ascending=False)[:20].index
data = racing_bar_filled[selected].round()

fig,ax = plt.subplots(figsize=(9.3,7))
fig.subplots_adjust(left=0.18)
no_of_frames = data.shape[0] #Number of frames

#initiate the barplot with the first rows of the dataframe
bars = sns.barplot(y=data.columns,x=data.iloc[0,:],orient="h",ax=ax)
ax.set_xlim(0,1500)
txts = [ax.text(0,i,0,va="center") for i in range(data.shape[1])]
title_txt = ax.text(650,-1,"Date: ",fontsize=12)
ax.set_xlabel("Pay (Millions USD)")
ax.set_ylabel(None)

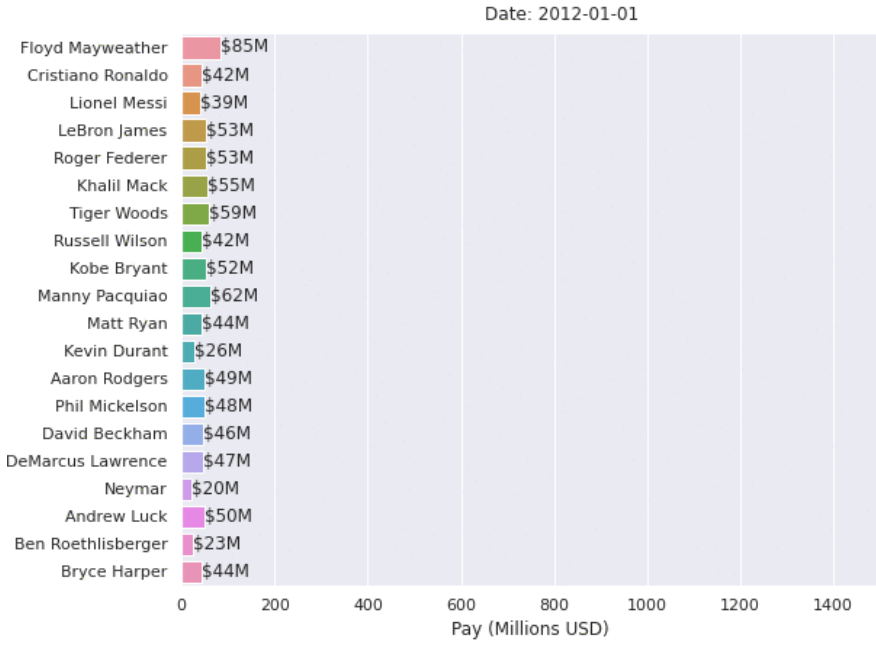
def animate(i):
#     print(f"i={i}/{no_of_frames}")
    #get i'th row of data
    y = data.iloc[i,:]
```

```
    #update title of the barplot axis
    title_txt.set_text(f"Date: {str(data.index[i].date())}")

    #update elements in both plots
    for j, b, in enumerate(bars.patches):
        #update each bar's height
        b.set_width(y[j])

        #update text for each bar (optional)
        txts[j].set_text(f"${y[j].astype(int)}M")
        txts[j].set_x(y[j])

anim=FuncAnimation(fig,animate,repeat=False,frames=no_of_frames,interval=1,
blit=False)
anim.save('athletes.gif', writer='imagemagick', fps=120)
plt.close(fig)
```



المصدر:

https://thecleverprogrammer.com/2020/12/21/highest-paid-athletes-analysis-/with-python

12) التنبؤ بنتائج مباريات الدوري الأميركي للمحترفين NBA باستخدام التعلم الآلي ولغة بايثون Predicting NBA Game Results Using Machine Learning and Python

بصفتي من مشجعي الدوري الأميركي للمحترفين NBA، وخاصة فريق لوس أنجلوس ليكرز Los Angeles Lakers، فإن كل مباراة تحمل الإثارة والترقب. بعد الخسارة المخيبة للآمال أمام فريق دنفر ناجتس Denver Nuggetsفي نهاية موسمنا، وجدت نفسي أرغبفي تحديد فرصنافي هزيمة حامل اللقب من الموسم الماضي، فريق بوسطن سيلتيكس Boston Celtics. وقد أثار هذا فضولي لاستخدام التعلم الآلي Machine Learning ولغة بايثون لمعرفة ما إذا كانت لدينا فرصة حقيقية لهزيمة فريق سيلتيكس.

في البداية، قررت استخدام Google Colab، الذي يوفر وصولاً مجانيًا قائمًا على السحابة إلى Jupyter notebooks المزودة بوحدات معالجة رسومية GPUs. سمح لي هذا الاختيار بالاستفادة من موارده الحسابية والتكامل السلس مع مكتبات بايثون مثل scikit-learn وmatplotlib. باستخدام البيانات المستمدة من واجهة برمجة تطبيقات الدوري الأميركي للمحترفين NBA API، يمكنني التنبؤ بنتائج المباريات من خلال تحليل أداء الفريق التاريخي ومتوسط النقاط المسجلة وميزة اللعب على أرضه والمزيد.

تثبيت المكتبات واستيرادها

```
!pip install pandas numpy scikit-learn requests nba-api shap
```

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report
from sklearn.preprocessing import LabelEncoder
from nba_api.stats.endpoints import leaguegamefinder
from nba_api.stats.static import teams
import shap
import matplotlib.pyplot as plt
import seaborn as sns
```

- **pandas وnumpy**: لمعالجة البيانات والعمليات العددية.
- **train_test_split وRandomForestClassifier وaccuracy_score وcategory_report**: مكونات من scikit-learn لتقسيم البيانات وتدريب النموذج وتقييم أدائه.
- **LabelEncoder**: لترميز المتغيرات الفئوية categorical variables إلى قيم عددية numerical values.

- **leaguegamefinder وteams**: مكونات من nba-api لجلب بيانات لعبة NBA ومعلومات الفريق.
- **shap**: لتفسير النموذج.
- **matplotlib وseaborn**: لتصور البيانات.

جلب البيانات

```
nba_teams = teams.get_teams()
team_abbr_to_id = {team['abbreviation']: team['id'] for team in nba_teams}
all_games = pd.DataFrame()

for team in nba_teams:
    team_id = team['id']
    gamefinder =
leaguegamefinder.LeagueGameFinder(team_id_nullable=team_id)
    games = gamefinder.get_data_frames()[0]
    all_games = pd.concat([all_games, games], ignore_index=True)

print(all_games.columns)
```

أقوم بإحضار قائمة بفرق NBA باستخدام الدالة teams.get_teams(). باستخدام قاموس يسمى team_abbr_to_id، أقوم بربط اختصار كل فريق بمعرفه الفريد unique ID.

بعد ذلك، أقوم بتهيئة إطار بيانات فارغ باسم all_games لتجميع إحصائيات اللعبة عبر جميع الفرق. من خلال LeagueGameFinder، أقوم بجمع بيانات اللعبة لكل فريق ودمجهافي all_games. تسمح لي هذه الخطوة بتحليل الأداء والأنماط التاريخية للتنبؤ بالنتائج. أقوم بطباعة أعمدة all_games للتحقق من دقة استرجاع البيانات.

معالجة البيانات

```
all_games['GAME_DATE'] = pd.to_datetime(all_games['GAME_DATE'])
all_games['WIN'] = all_games['WL'].apply(lambda x: 1 if x == 'W' else 0)
all_games['PTS'] = all_games['PTS'].astype(float)
all_games['Points_Per_Game'] =
all_games.groupby('TEAM_ID')['PTS'].transform('mean')

def get_opponent_team_id(matchup, team_abbr_to_id, team_id):
    if '@' in matchup:
        opponent_abbr = matchup.split(' @ ')[-1]
    else:
        opponent_abbr = matchup.split(' vs. ')[-1]
    return team_abbr_to_id.get(opponent_abbr, team_id)

all_games['OPPONENT_TEAM_ID'] = all_games.apply(
    lambda row: get_opponent_team_id(row['MATCHUP'], team_abbr_to_id,
row['TEAM_ID']), axis=1
)

all_games['HOME_GAME'] = all_games['MATCHUP'].apply(lambda x: 1 if 'vs.' in
x else 0)
all_games['LAST_GAME_RESULT'] =
all_games.groupby('TEAM_ID')['WIN'].shift(1).fillna(0)
```

أقوم بتحويل عمود GAME_DATE إلى تنسيق التاريخ والوقت للتحليل القائم على الوقت time-based analysis. ثم أقوم بإنشاء عمود WIN الذي يصنف كل مباراة على أنها فوز (1) أو خسارة (0) بناءً على عمود "WL".

بعد ذلك، أريد الاتساق العددي numerical consistency من خلال تحويل عمود PTS إلى نوع float، وهو أمر ضروري للحسابات الإحصائية. بالنسبة لأداء الفريق، أقوم بحساب متوسط النقاط لكل مباراة باستخدام عمود Points_Per_Game، والذي يوفر معيارًا لمخرجات الهجوم لكل فريق.

لتحسين مجموعة البيانات الخاصة بي بشكل أكبر، أقوم بتعريف دالة get_opponent_team_id لاستخراج معرف فريق الخصم من عمود MATCHUP. تساعد هذه الدالةفي إنشاء عمود OPPONENT_TEAM_ID، مما يسهل التحليل المقارن بين الفرق.

كما أقوم بتقديم عمود HOME_GAME الذي يميز بين المباريات التي تقام على أرض الفريق (1) والمباريات التي تقام خارج أرضه (0). وأخيرًا، أقوم بدمج عمود LAST_GAME_RESULT لتتبع نتيجة المباراة السابقة للفريق.

ترميز التسمية للبيانات

```
le = LabelEncoder()
all_games['TEAM_ID'] = le.fit_transform(all_games['TEAM_ID'])
all_games['OPPONENT_TEAM_ID'] =
le.fit_transform(all_games['OPPONENT_TEAM_ID'])
```

أستخدم أداة LabelEncoderفي بايثون لتحويل البيانات الفئوية إلى تنسيق رقمي. يحول هذا التحويل عمودي TEAM_ID وOPPONENT_TEAM_ID، اللذان كانا يمثلان اختصارات الفريق سابقًا، إلى قيم رقمية يمكن لخوارزميات التعلم الآلي فهمها.

تقسيم البيانات

```
X = all_games[['TEAM_ID', 'OPPONENT_TEAM_ID', 'Points_Per_Game',
'HOME_GAME', 'LAST_GAME_RESULT']]
y = all_games['WIN']
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)
```

أبدأ بتحديد مصفوفة الميزات X التي تتضمن أعمدة رئيسية من مجموعة البيانات الخاصة بي، مثل TEAM_ID وOPPONENT_TEAM_ID وPoints_Per_Game وHOME_GAME وLAST_GAME_RESULT. ستساعدني هذه الميزاتفي تحليل أداء الفريق وديناميكيات اللعبة.

بالنسبة للمتغير المستهدف target variable، y، أركز على عمود WIN، الذي يشير إلى ما إذا كانت اللعبة قد فازت (1) أو خسرت (0). يعمل هذا العمود كأساس للتنبؤ بنتائج اللعبة.

لتقييم دقة النموذج وفعاليته، قمت بتقسيم مجموعة البيانات الخاصة بي إلى مجموعات تدريب واختبار training and testing sets باستخدام تقسيم 80ـ20. يخصص هذا النهج 80٪ من البيانات لتدريب النموذج، مما يسمح له بتعلم الأنماط والعلاقات، مع الاحتفاظ بنسبة 20٪ لاختبار قدراته التنبؤية على البيانات غير المرئية.

تدريب النموذج

```
model = RandomForestClassifier(n_estimators=100, random_state=42)
model.fit(X_train, y_train)
```

أقوم بتشغيل RandomForestClassifier — وهي خوارزمية تستخدم 100 شجرة قرار decision trees.

باستخدام مصفوفة الميزات X_train والمتغير المستهدف y_train المجهزين من مجموعة بيانات التدريب، أشرعفي تدريب النموذج. تتضمن هذه المرحلة تغذية RandomForestClassifier ببيانات اللعبة التاريخية، مما يتيح له التعلم من الأنماطفي أداء الفريق وظروف اللعبة والنتائج السابقة.

تقييم النموذج

```
y_pred = model.predict(X_test)
print("Accuracy:", accuracy_score(y_test, y_pred))
print(classification_report(y_test, y_pred))
```

باستخدام RandomForestClassifier المدرب على بيانات لعبة NBA، أطبقه للتنبؤ بنتائج اللعبة على مجموعة الاختبار (X_test). تتيح لي هذه الخطوة تقييم مدى قدرة النموذج على التعميم على البيانات غير المرئية.

بعد ذلك، أقوم بطباعة درجة الدقة accuracy score، والتي تقيس النسبة المئوية للتنبؤات الصحيحة التي قام بها النموذج. بالإضافة إلى ذلك، أقوم بإنشاء تقرير تصنيف classification report يتضمن الدقة precision والاستدعاء ونتيجة F1 (F1-score). توضح هذه المقاييس قدرة النموذج على تحديد نتائج اللعبة بشكل صحيح، مع مراعاةكل من الدقة التنبؤية predictive accuracy والاتساق consistency عبر نتائج اللعبة المختلفة.

أهمية الميزة

```
feature_importances = pd.DataFrame(model.feature_importances_,
                                   index = X_train.columns,

columns=['importance']).sort_values('importance', ascending=False)
print("Feature Importances:\n", feature_importances)
```

للحصول على رؤى حول العوامل الأكثر تأثيرًا على نتائج مباريات الدوري الأميركي للمحترفين، أقوم باستخراج أهمية الميزات من RandomForestClassifier المدرب. تكشف هذه الأهمية عن

مساهمة كل ميزة — مثل TEAM_ID وOPPONENT_TEAM_ID وPoints_Per_Game وHOME_GAME وLAST_GAME_RESULT — في التنبؤ بنتائج المباريات.

باستخدام هذه الأهمية، أقوم بإنشاء إطار بيانات ينظم ويصنف كل ميزة بناءً على تأثيرها. من خلال فرز الميزات بترتيب تنازلي للأهمية، أقوم بطباعة النتائج لتحديد جوانب أداء الفريق وديناميكيات اللعبة الأكثر أهميةفي تحديد نتائج المباريات.

اجراء التنبؤات

```
team_abbr = 'LAL'
opponent_abbr = 'BOS'
average_points_per_game = 110.5

new_data = pd.DataFrame({
    'TEAM_ID': [le.transform([team_abbr_to_id[team_abbr]])[0]],
    'OPPONENT_TEAM_ID':
[le.transform([team_abbr_to_id[opponent_abbr]])[0]],
    'Points_Per_Game': [average_points_per_game],
    'HOME_GAME': [1],
    'LAST_GAME_RESULT': [1]
})

predictions = model.predict(new_data)
prediction_probabilities = model.predict_proba(new_data)

print("Predictions: ", predictions)
print("Prediction Probabilities: ", prediction_probabilities)
```

عند تطبيق نموذجي التنبئي على مباراة افتراضية بين فريق ليكرز (LAL) وفريق بوسطن سيلتيكس (BOS)، أقوم أولاً بتحديد اختصارات الفريق والخصم، إلى جانب متوسط النقاط لكل مباراة. ومع توفر هذه التفاصيل، أقوم بإنشاء إطار بيانات باسم new_data والذي يلخص الميزات الرئيسية اللازمة للتنبؤ.

بالاستفادة من RandomForestClassifier المدرب على بيانات لعبة NBA، أشرعفي التنبؤ بالنتيجة والاحتمالية لهذه المباراة الجديدة. بعد إدخال new_dataفي النموذج، أقوم بطباعة التنبؤ والاحتمالات المقابلة، مما يوفر رؤى حول احتمالية فوز فريق ليكرز على فريق سيلتيكس.

النتائج

```
Prediction Probabilities:  [[0.67180749 0.32819251]]
```

هذا يعني أن النموذج يتنبأ باحتمالية تبلغ حوالي 67.18% لخسارة (0) وحوالي 32.82% لفوز (1) لليكرز ضد سيلتيكس إذا كانت هذه مباراة خارج أرضه لليكرز.

```
Prediction Probabilities:  [[0.50777377 0.49222623]]
```

هذه هي احتمالية فوز فريق ليكرزفي مباراة خارج أرضه، وهي نسبة 50.78% من خسارتنا و49.22% من فوزنا.

للأسف، خسرنا المباراتين، مما يعني أن فريق ليكرز ليس مهيأ للفوز بالبطولة بعد.

تحسين النموذج في المستقبل

طالما أن مجموعة البيانات الخاصة بي محدثة دائمًا، يمكنني تضمين المزيد من الميزات التي يمكن أن تكون تنبؤية، مثل إحصائيات اللاعبين وتقارير الإصابات والأداء التاريخي وما إلى ذلك. يمكن أن تكون هناك أيضًا ميزات إضافية مثل أيام الراحة بين المباريات ومسافات السفر والمباريات المتتالية وما إلى ذلك لالتقاط شكل الفريق حقًا.

يمكن أن يكون التحسين الآخر هو تجربة نماذج التعلم الآلي المختلفة مثل XGBoost أو LightGBM أو الشبكات العصبية لمعرفة ما إذا كانت تعمل بشكل أفضل من RandomForestClassifier.

ثم يمكنني استخدام تقنيات مثل Grid Search أو Random Search للعثور على أفضل المعلمات الفائقة hyperparameters لنموذجي.

أخيرًا، هناك طرق لمعالجة اختلالات التوازن imbalances المحتملةفي الفئات إذا كان هناك عدد أكبر بكثير من الانتصارات مقارنة بالخسائر على سبيل المثال.

الاستنتاج

على الرغم من أن نموذج التنبؤ الخاص بيفي الدوري الأميركي للمحترفين NBA ربما تنبأ بخسارة فريق ليكرز أمام فريق سيلتيكس، إلا أن هذا المشروع ساعدني على تعلم الكثير. فقد ساعدني على التعرف على أساسيات التعلم الآلي وبرمجة بايثون، مما ساعدني على فهم مفاهيم مثل معالجة البيانات مسبقًا، وهندسة الميزات، وتدريب النموذج باستخدام RandomForestClassifier، وتقييم أداء النموذج من خلال مقاييس مثل الدقة والدقة والتذكير ونتيجة F1.

رابط الكود :

https://github.com/juliuscecilia33/NBA-Game-Predictions?source=post_page----------------------------6be209d6d165

المصدر:

https://medium.com/@juliuscecilia33/predicting-nba-game-results-using-machine-learning-and-python-6be209d6d165

13) تحليل بيانات تسديدات دوري كرة السلة الأمريكي باستخدام بايثون NBA shot data analytics with Python

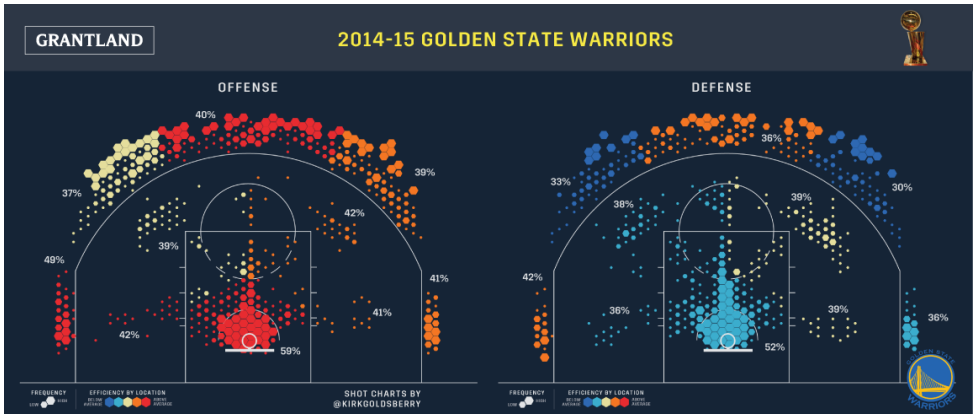
لقد دخل موسم الدوري الأميركي للمحترفين NBA الآن على قدم وساق. ومثل العديد من الرياضات الأخرى هذه الأيام، كان الدوري الأميركي للمحترفين NBA يمر بتغيير كبيرفي طريقة لعب اللعبة.

يمكن أن يُعزى الكثير من هذا التغيير إلى حركة "التحليلات المتقدمة advanced analytics" ـ كما تعلمون، أشياء مثل Moneyball (أو Moreyball، بالنسبة لنا نحن المهووسين بالدوري الأميركي للمحترفين NBA). هناك العديد من الإحصائيات الرائعة التي يمكنك استخدامها لقياس أداء فريقك ولاعبك المفضلين هذه الأيام، وأدوات التصور للتواصل معهم.

في هذه المقالة، أستخدم بايثون وPandas وMatplotlib للتلاعب بإحصائيات الدوري الأميركي للمحترفين NBA (مخططات التسديدات الأولية primary shot charts) وتحليلها وتصورها. الكثير من هذه البيانات يمكن الوصول إليها بسهولة (مجانًا أو بتكلفة منخفضة نسبيًا)، لذا يجب أن تكون قادرًا على تحليل هذه البيانات بنفسك.

تصور بيانات التسديد

كنقطة دخول، دعنا نلقي نظرة على بيانات التسديد shot data. هدف اللعبة هو تسجيل الأهداف. ولتسهيل ذلك، عليك إنشاء أكبر عدد ممكن من التسديدات عالية الجودة (نسبة مئوية عالية). لذا، فإن النظر إلى بيانات التسديد هو بوابة طبيعية لنشر تحليلات البياناتفي كرة السلة.



مخطط التسديدات بواسطة كيرك جولدسبيري (جرانتلاند)في أحد الرسوم البيانية، يتم رسم تسديدات فريق واحد طوال الموسم. ويُظهر موقع التسديدات، وتكرارها النسبي مقارنة بالمواقع الأخرى، وكفاءة

كل موقع مقارنة ببقية الدوري، كل ذلك بطريقة بديهية ومفيدة. نادرًا ما يتحسن تصور البيانات أكثر من هذا.

لذا، على مدار المنشورات القليلة القادمة، سنعمل على إعادة إنشاء مخططات مماثلة لهذه، ولكن ببيانات أكثر حداثة. ولكن أولاً، دعنا نبدأ من البداية.

ملاحظة - أفترض بعض الألفة مع بايثون وpandas وmatplotlib. ولكن ربما لا يكون من الصعب متابعتها إذا كان لديك بعض الخبرةفي بايثون على الأقل.

الحصول على البيانات

ما ستحتاج إليه لإنشاء هذه المخططات الجميلة هو مجموعة بيانات dataset تحتوي على جميع التسديدات التي تريد وضعهافي رسم بياني. يجب أن تتضمن مجموعة البيانات لكل تسديدة إحداثيات x وy، واسم اللاعب، وما إذا كانت التسديدة ناجحة أم لا.

بالنسبة لي، اشتريت للتو بيانات الموسم من هذا الموقع. حصلت على بيانات اللعب حسب اللعبة، والتي تتضمن كل لعبة من كل مباراة ـ مصنفة حيثما أمكن، بمافي ذلك أوصاف اللعبة.

إذا كنت تبحث عن اختبار البرامج النصية فقط، فإن BigDataBall يوفر مجموعة بيانات تجريبية، والتي يجب أن تكون قادرًا على تنزيلها ومتابعتها.

استيراد البيانات وتنظيفها

لقد حصلت على بياناتي بتنسيق csv، والتي يمكن تحميلها مباشرة إلى إطار البيانات dataframe ـ وهذا هو بالضبط ما فعلته.

```
import pandas as pd
logpath = [CSV DATA FILE PATH]
log_df = pd.read_csv(logpath)
```

نحن مهتمون بالتسديدات فقط —— لذا نود تصفية إطار البيانات. بالنظر إلى البيانات باستخدام log_df.columns —— نجد عمود "event_type". دعنا نرى ما إذا كان هذا هو العمود الذي نبحث عنه —— أقوم بطباعة قيمه الفريدة من خلال:

```
print(log_df.event_type.unique())
```

أرى أن هناك قيمة "shot" وقيمة "miss" —— رائع. باستخدام هذا، يمكنني إنشاء إطار بيانات مفلتر، مع فلتر حيث تكون معلمة event_type عبارة عن "shot" أو "miss".

```
shots_df = log_df[(log_df.event_type == 'shot') | (log_df.event_type ==
'miss')]
```

هكذا تمامًا، لدينا إطار بيانات مفلتر!

حسنًا، هناك شيء أخير — دعنا نبني عمودًا جديدًا حيث لدينا تصنيف ثنائي (0/1) حيث 1 هو للضربة الناجحة، و0 للضربات الفاشلة. سيكون هذا مفيدًا بدلاً من الاضطرار إلى التصفية حسب متغير سلسلة.

لقد فعلت ذلك عن طريق إنشاء عمود جديد بقيمة صفرية، واستخدام طريقة maskdataframe. لإعطاء القيم حيث تكون قيمة العمود "event_type" هي "ضربة".

```
shots_df['shot_made'] = 0
shots_df['shot_made'] = shots_df['shot_made'].mask(shots_df.event_type ==
'shot', 1)
```

حسنًا، رائع. أوه — إذا كنت تقوم بتحميل ملفات متعددة، فكل ما عليك فعله هو تحميلها جميعًا وإضافتها إلى قائمة ثم استخدام pd.concat.

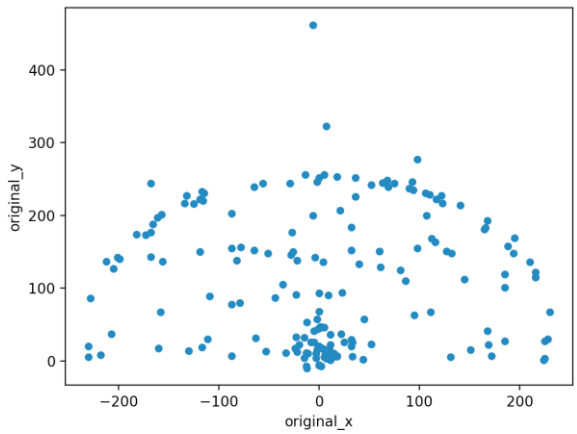
على سبيل المثال، إذا كانت temp_list هي قائمة إطارات البيانات الخاصة بك، فيمكنك إضافتها جميعًا معًا من خلال: pd.concat(temp_list, axis=0, ignore_index=True).

ارسم لي شيئًا

حسنًا، حسنًا. دعنا ننتقل إلى الجزء الممتع.في الواقع، الآن بعد أن تم إنجاز كل العمل، أصبح جزء الرسم سهلًا للغاية. دعنا نلقي نظرة على جميع محاولات التسديدفي مباراة 20 أكتوبر 2018 بين فريق سبيرز Spurs وفريق تريل بليزرز Trail Blazers.

shots_df كما تمت تصفيته يتضمن 184 صفًا – وإلقاء نظرة على نتيجة المباراة على Basketball-reference.com يخبرني أن هذا صحيح. كيف نرسمها؟

حسنًا، دعنا نبدأ بمخطط تشتت scatter plot بسيط – الإحداثيات موجودة بالفعل، لذا فإن shots_df.plot.scatter('original_x', 'original_y') البسيط سيرسمفي الواقع جميع التسديدات!... مهما كان الأمر قبيحًا.



مخطط تسديداتي الأولى (نعم، ليس جميلاً) إذا حدقت قليلاً، يمكنك أن ترى منطقة الطلاء وقوس النقاط الثلاث. ولكن بدون خطوط الملعب، يكون الأمر سيئًا للغاية، ويبدو أن النسبة خاطئة، والمحاور ليست ضرورية، ولا يمكننا حتى معرفة أي منها أخطأ وأي منها نجح! دعنا نصلح كل هذه الأشياء.

رسم الملعب

لحسن الحظ، يسمح برنامج matplotlib بالرسم على النافذة باستخدام patches.. يمكنك رسم أي شيء تريده باستخدام patches — حتى هذه الأشكال العشوائية random shapes تمامًا، والتي توجدفي الوثائق لسبب ما.

لكن النقطة المهمة هي أنه يمكنك رسم الملعب باستخدام مجموعة من الأقواس والخطوط والدوائر. يوفر هذا البرنامج التعليمي الممتاز الكود. إنه ممتاز، وقد قمت ببساطة بإعادة استخدام الكود بدلاً من إعادة اختراع العجلات. لقد احتفظت أيضًا باسم الدالة، والذي يُسمى: draw_court

ترميز التسديدات الناجحة/الفاشلة

هذه هي اللحظة التي يصبح فيها عمود "shot_made" الذي أنشأناه أعلاه مفيدًا. يمكننا ببساطة تعيين هذه القيم كألوانفي مخطط التشتت عن طريق تمرير هذا العمود إلى معلمة "c"في مخطط التشتت باستخدام matplotlib.

يشير الرسم البياني الآن بصريًا إلى محاولات التسديد الفاشلة وأيها كانت أهدافًا ميدانية.

إذا جربت هذا، فستلاحظ أن خرائط الألوان الافتراضية بشعة للغاية — لذلك بحثتفي خرائط الألوان المتاحة هذه واخترت coolwarm_r (تشير "_r" إلى خرائط ألوان معكوسة، لترميز التسديدات الناجحة باللون الأزرق). يمكنك مشاهدة البرنامج التعليمي هنا بمافي ذلك قائمة خرائط الألوان.

إصلاح نسبة الملعب

يسمح لك Matplotlib بتحديد أبعاد الإخراج لمخططك، باستخدام معلمة figsize لدالة figure.

نظرًا لأن أبعاد الملعب هي 94 × 50 قدمًا، ونحن نرسم نصف ملعب، فسوف نستخدم بعض الاختلافاتفي نسبة 47:50 (أو ما يقرب منها). أيضًا، فإن 50 هو البعد "x"في هذه الحالة – لذا فقد يكون أقرب إلى 47:50.

لذا، يمكننا تحقيق ذلك باستخدام: plt.figure(figsize(5, 4.7))

ملاحظة: إذا كنت تتساءل ما هو plt، فهو الاختصار المعتاد لـ matplotlib.pyplot – مستورد كـ plt:

```
import matplotlib.pyplot as plt
```

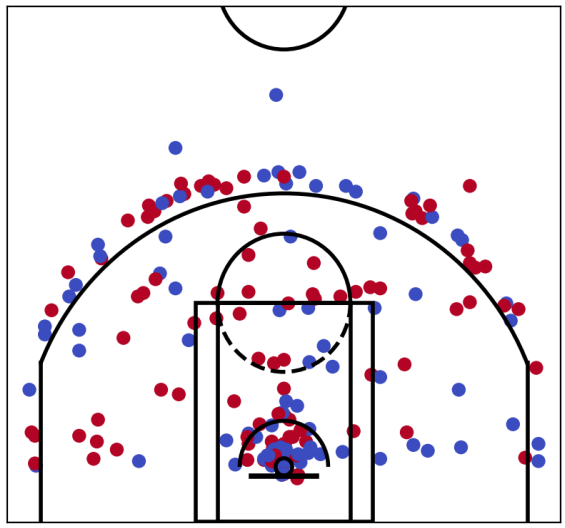
إخفاء المحاور

يمكننا استخدام طريقة gca().في matplotlib للحصول على كائن المحاور للرسم البياني الحالي — وببساطة باستخدام axes.get_xaxis().set_visible(False). (أو yaxis بدلاً من xaxis) سيتم إخفاء المحور — مع العلامات وكل شيء.

أرني الرسم البياني!

وبجمع كل ذلك معًا، يمكننا إنشاء مخطط التسديات المحسن الخاص بنا باستخدام:

```
plt.figure(figsize=(5, 4.7))
plt.scatter(shots_df.original_x, shots_df.original_y, c=shots_df.shot_made,
cmap='coolwarm_r')
draw_court(outer_lines=False)
cur_axes = plt.gca()
cur_axes.axes.get_xaxis().set_visible(False)
cur_axes.axes.get_yaxis().set_visible(False)
plt.xlim(250, -250)
plt.ylim(-47.5, 422.5)
plt.show()
```



هنا مخطط التسديدات طوال المباراة. ولكن ـ انتظر، هذا هو مخطط التسديدات لكلا الفريقين. دعنا نبدأي تحليل هذه الأرقام ومقارنة الفريقين واللاعبين.

مخططات التسديدات— POR مقابل SAS

الآن بعد أن أصبح لدينا كل ما نحتاجه لرسم مخططات التسديات، أصبح الأمر مجرد تصفية إطار البيانات الخاص بنا وتمريره عبر نفس العملية.

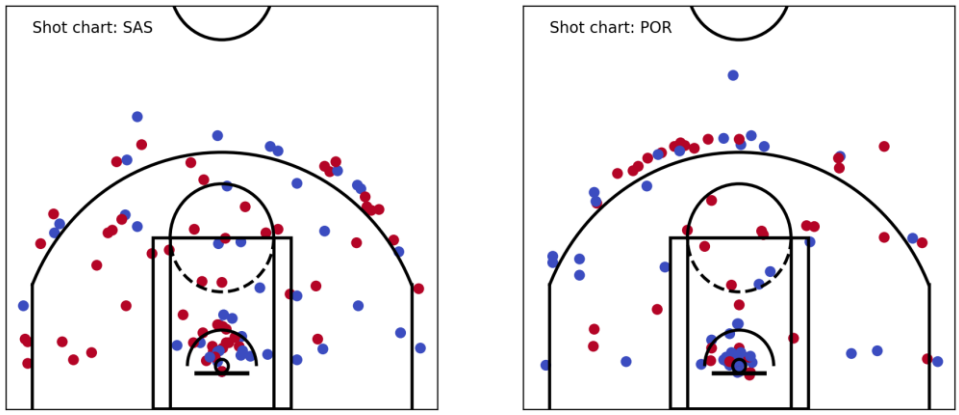
بالنظر إلى عمود "team" باستخدام print(shots_df.team.unique())، يظهر "SAS" و"POR" كقيمتين فقط، لذا يمكننا التصفية حسبهما. على سبيل المثال، يمكننا الحصول على تسديدات سان أنطونيو San Antonio من خلال:

```
sas_shots_df = shots_df[shots_df.team == 'SAS']
```

ويمكننا إنشاء مخططات التسديات باستخدام نفس العملية المذكورة أعلاه. حسنًا، دعنا نضيف القليل من النص لتوضيح أيهما أي. نظرًا لنظام الإحداثيات الخاص بنا، يمكنني رسم القليل من النص في الجزء العلوي الأيسر باستخدام:

```
plt.text(220, 370, 'Shot chart: ' + teamname)
```

وهكذا ـ فهي تبدو مثل هذا:



مخططات التسديد الخاصة بكل فريق — SAS vs POR

الحصول على مخطط تسديدات لاعب فردي

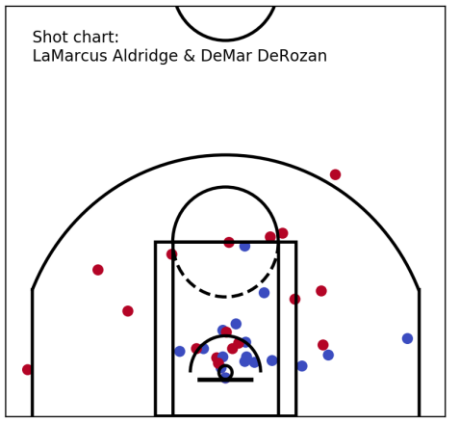
أخيرًا، دعنا نلقي نظرة على إنشاء مخططات تسديدات فردية. سنختار أفضل لاعبين من كل فريق لإنشاء مخططات تسديدات لهما (ونعرضهما معًا).

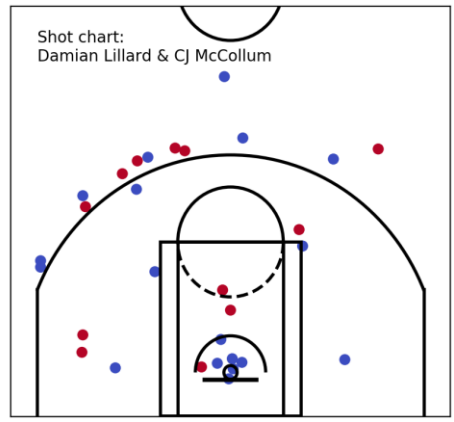
مرة أخرى، الأمر يتعلق فقط بتصفية البيانات — هذه المرة بأسماء اللاعبين. نظرًا لأننا نريد التصفية لأشخاص متعددين، فإننا نستخدم مرة أخرى عامل "أو"في تصفية إطار البيانات، وهو عبارة عن خط عمودي (|).

للتصفية لـ LaMarcus Aldridge و DeMar DeRozan من فريق Spurs، استخدم:

```
pl_shots_df = shots_df[(shots_df.player == 'LaMarcus Aldridge') |
(shots_df.player == 'DeMar DeRozan')]
```

وبإمكاننا أن نفعل الشيء نفسه مع داميان ليلارد Damian Lillard وسي جيه ماكولوم CJ McCollum من فريق تريل بليزرز Trail Blazer.

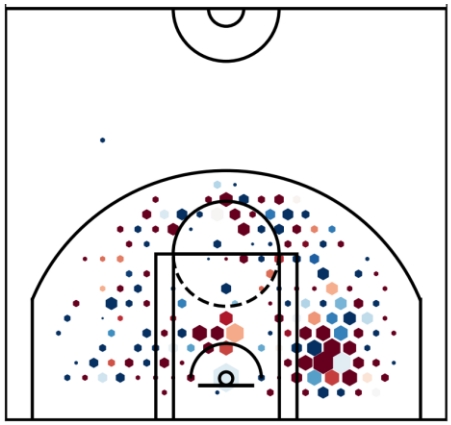


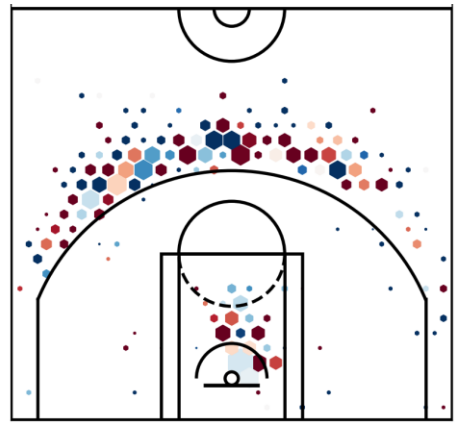


يبدو رائعًا، أليس كذلك؟ باستثناء كل التسديدات متوسطة المدى التي نفذها الثنائي العملاقفي فريق سبيرز... لكن هذه قصة ليوم آخر.

دعونا نتوقف عند هذا الحد الآن. ولكن بفكرة واحدة ــ كيف ستبدو مخططات التسديد هذه، إذا كانت تتضمن مخططات من مباريات متعددة ــ حتى موسم كامل؟ هل يمكنك الاستفادة منها كثيرًا؟ كيف يمكنك مقارنة مخطط تسديدات لاعب أو فريق بآخر والحصول على معلومات مفيدة منه؟

وهنا تأتي فائدة مخططات الهيكس بين hexbin plots هذه.





المصدر:

https://www.jphwang.com/posts/nba-shot-data-analytics-visualization-with-/python-pandas-and-matplotlib-part-1-the-basics

14)تحليل أداء اللاعبين في كرة السلة Player Performance Analysis in basketball

لقد توصلت إلى مجموعة البيانات dataset هذه لمعرفة المدة التي يمكن للاعب أن يلعبها بناءً على إحصائيات الملخص السابقة. يوجدفي مجموعة البيانات هذه 21 سمة تصف مقاييس الأداء لكل لاعب أو يمكنك القول ملخص كل لاعب. مهمتك هي التنبؤ بالمتغير المستهدف Target Variable. المتغير المستهدف: 1 ـ ما إذا كانت مسيرة اللاعب تساوي أو تزيد عن 5 سنوات. 0 ـ المسيرة أقصر من 5 سنوات الإرسال المتوقع يجب عليك حل المهمةفي المقام الأول باستخدام دفاتر ملاحظات التقييم Notebooks Evaluation. استخدم خوارزميات التصنيف المختلفة Classification Algorithms للتنبؤ بالمتغير المستهدف بدرجة دقة أعلى.

استيراد المكتبات وتحميل البيانات

استيراد المكتبات المطلوبة

```
import numpy as np # linear algebra
import pandas as pd # data processing, CSV file I/O (e.g. pd.read_csv)
pd.set_option('display.max_rows', None)
pd.set_option('display.max_columns', None)
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
from imblearn.combine import SMOTETomek
```

تحميل وفحص البيانات

```
df = pd.read_csv('../input/performance-prediction/summary.csv')
df.head()
```

	Name	GamesPlayed	MinutesPlayed	PointsPerGame	FieldGoalsMade	FieldGoalsAttempt	FieldGoalPercent	3PointMade
0	Brandon Ingram	36	27.4	7.4	2.6	7.6	34.7	0.5
1	Andrew Harrison	35	26.9	7.2	2.0	6.7	29.6	0.7
2	JaKarr Sampson	74	15.3	5.2	2.0	4.7	42.2	0.4
3	Malik Sealy	58	11.6	5.7	2.3	5.5	42.6	0.1
4	Matt Geiger	48	11.5	4.5	1.6	3.0	52.4	0.0

التحقق من بُنية مجموعة البيانات:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1340 entries, 0 to 1339
Data columns (total 21 columns):
 #   Column              Non-Null Count  Dtype
---  ------              --------------  -----
 0   Name                1340 non-null   object
```

```
 1   GamesPlayed         1340 non-null   int64
 2   MinutesPlayed       1340 non-null   float64
 3   PointsPerGame       1340 non-null   float64
 4   FieldGoalsMade      1340 non-null   float64
 5   FieldGoalsAttempt   1340 non-null   float64
 6   FieldGoalPercent    1340 non-null   float64
 7   3PointMade          1340 non-null   float64
 8   3PointAttempt       1340 non-null   float64
 9   3PointPercent       1329 non-null   float64
 10  FreeThrowMade       1340 non-null   float64
 11  FreeThrowAttempt    1340 non-null   float64
 12  FreeThrowPercent    1340 non-null   float64
 13  OffensiveRebounds   1340 non-null   float64
 14  DefensiveRebounds   1340 non-null   float64
 15  Rebounds            1340 non-null   float64
 16  Assists             1340 non-null   float64
 17  Steals              1340 non-null   float64
 18  Blocks              1340 non-null   float64
 19  Turnovers           1340 non-null   float64
 20  Target              1340 non-null   int64
dtypes: float64(18), int64(2), object(1)
memory usage: 220.0+ KB
```

التحقق من القيم المفقودة:

```
df.isnull().sum()
```

```
Name                   0
GamesPlayed            0
MinutesPlayed          0
PointsPerGame          0
FieldGoalsMade         0
FieldGoalsAttempt      0
FieldGoalPercent       0
3PointMade             0
3PointAttempt          0
3PointPercent         11
FreeThrowMade          0
FreeThrowAttempt       0
FreeThrowPercent       0
OffensiveRebounds      0
DefensiveRebounds      0
Rebounds               0
Assists                0
Steals                 0
Blocks                 0
Turnovers              0
Target                 0
dtype: int64
```

لدينا 11 قيمة مفقودةفي PointPercent3،

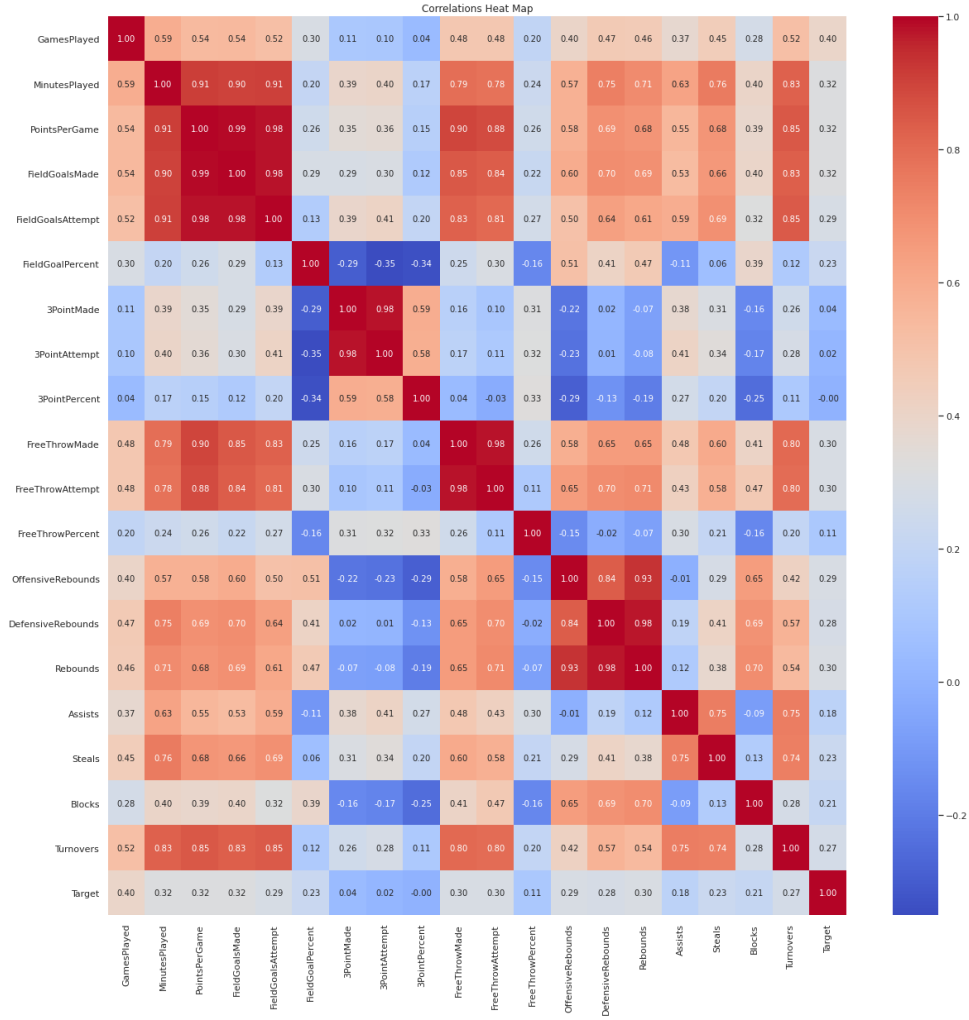
بعد التحقق من إدخالات إطار البيانات حيث توجد بيانات مفقودة، تبين أنها مفقودة لأنها قسمة صفرية، لذا نستبدلها بالصفر بدلاً من ذلك.

```
df = df.fillna(0)
```

تحليل البيانات الاستكشافي

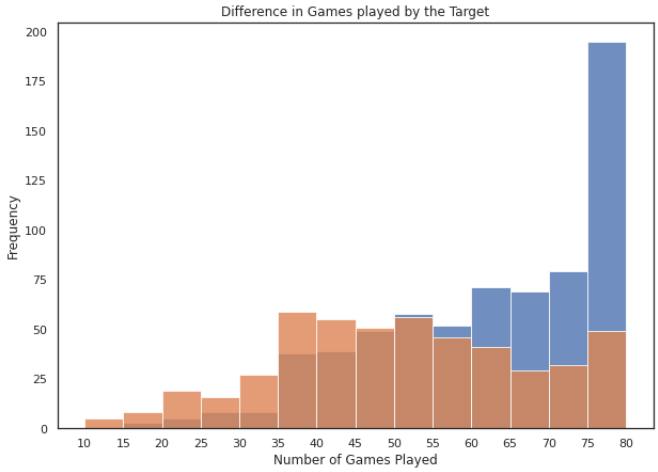
دعنا نتحقق من ارتباطات الميزات الأخرى باستخدام خريطة حرارية heat map.

```
columns = df.columns.tolist()[1:]
plt.figure(figsize=(20,20))
sns.heatmap(df[columns].corr(),annot=True, fmt = ".2f", cmap = "coolwarm")
plt.title('Correlations Heat Map')
plt.show()
```



دعنا نستكشف ميزة GamesPlayed لمعرفة ما إذا كانت مرتبطة:

```
bins = np.arange(10,df.GamesPlayed.max(),5)
plt.figure(figsize=(10,7))
plt.hist(df[df.Target == 1].GamesPlayed,alpha=0.8,bins=bins)
plt.hist(df[df.Target == 0].GamesPlayed,alpha=0.8,bins=bins)
plt.title('Difference in Games played by the Target')
plt.xlabel('Number of Games Played')
plt.ylabel('Frequency')
plt.xticks(bins)
plt.show()
```



نستنتج من هذا أن متوسط مسيرة اللاعبين الذين لعبوا أكثر من 50 مباراة من المرجح أن تكون مساوية أو أكبر من 5 سنوات.

من الواضح أن المزيد من المباريات ستؤدي إلى المزيدفي كل ميزة أخرى، لذا دعنا نقدم بعض الميزات الجديدة.

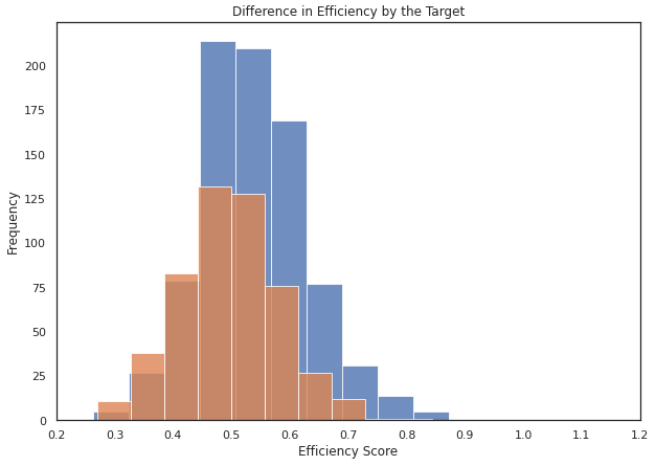
بعد بعض البحث على جوجل، يمكننا تقديم ميزتين جديدتين:

Player Efficiency Rating : (FieldGoalsMade + Rebounds + Assists + Steals + Blocks + Turnovers)/ MinutesPlayed

Participation : MinutesPlayed/GamedPlayed

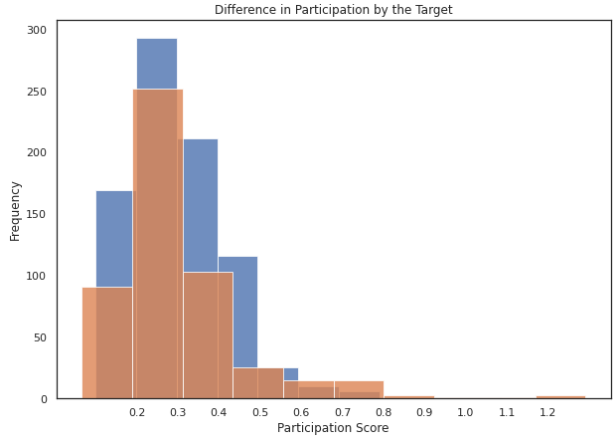
```
df['efficiency'] =
(df['FieldGoalsMade']+df['Rebounds']+df['Assists']+df['Steals']+df['Blocks'
]+df['Turnovers'])/df['MinutesPlayed']
df['Participation'] = df['MinutesPlayed']/df['GamesPlayed']
```

```
bins = np.arange(0.2,df.Participation.max(),0.1)
plt.figure(figsize=(10,7))
plt.hist(df[df.Target == 1]['efficiency'],alpha=0.8)
plt.hist(df[df.Target == 0]['efficiency'],alpha=0.8)
plt.title('Difference in Efficiency by the Target')
plt.xlabel('Efficiency Score')
plt.ylabel('Frequency')
plt.xticks(bins)
plt.show()
```



حسنًا،في كل الأحوال، ترتبط درجات الكفاءة الأعلى باللاعبين الأكثر خبرة experienced players.

```
bins = np.arange(0.2,df.Participation.max(),0.1)
plt.figure(figsize=(10,7))
plt.hist(df[df.Target == 1]['Participation'],alpha=0.8)
plt.hist(df[df.Target == 0]['Participation'],alpha=0.8)
plt.title('Difference in Participation by the Target')
plt.xlabel('Participation Score')
plt.ylabel('Frequency')
plt.xticks(bins)
plt.show()
```



لا يوجد فرق كبير بين الهدفين عندما يتعلق الأمر بنتيجة المشاركة.

الآن دعنا نجري بعض التوقعات.

سنبدأ بـ XGBoost.

```
from sklearn.model_selection import train_test_split
```

```
from sklearn import metrics
from xgboost import XGBClassifier
from sklearn.preprocessing import MinMaxScaler
mms = MinMaxScaler(feature_range=(0,1))
```

```
#Target
Y = df['Target'].values
#Inputs
X = df[['GamesPlayed', 'MinutesPlayed', 'PointsPerGame',
       'FieldGoalsMade', 'FieldGoalsAttempt', 'FieldGoalPercent',
'3PointMade',
       '3PointAttempt', '3PointPercent', 'FreeThrowMade',
'FreeThrowAttempt',
       'FreeThrowPercent', 'OffensiveRebounds', 'DefensiveRebounds',
       'Rebounds', 'Assists', 'Steals', 'Blocks', 'Turnovers',
       'efficiency', ]].values
#Normalize our variables
X = mms.fit_transform(X)
#Split to training and testing
X train, X test, y train, y test = train test split(X, Y, test size=0.2)
#Define the model
model = XGBClassifier(learning_rate = 0.1,n_estimators=200, max_depth=6)
#train the model
model.fit(X_train, y_train)
#Check training accuracy
trainingAccuracy =  metrics.accuracy_score(y_train,model.predict(X_train))
print("Training Accuracy: %.2f%%" % (trainingAccuracy * 100.0))
#Check testing accuracy
testingAccuracy =  metrics.accuracy_score(y_test, model.predict(X_test))
print("Testing Accuracy: %.2f%%" % (testingAccuracy * 100.0))
```

```
Training Accuracy: 98.60%
Testing Accuracy: 67.91%
```

حصلنا على دقة 72.4% من XGBoost.

لنجرب مصنفات أخرى:

```
from collections import Counter

from sklearn.ensemble import RandomForestClassifier, AdaBoostClassifier,
GradientBoostingClassifier, ExtraTreesClassifier, VotingClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.linear_model import LogisticRegression
from sklearn.neighbors import KNeighborsClassifier
from sklearn.tree import DecisionTreeClassifier
from sklearn.neural_network import MLPClassifier
from sklearn.svm import SVC
from sklearn.model_selection import GridSearchCV, cross_val_score,
StratifiedKFold, learning_curve

sns.set(style='white', context='notebook', palette='deep')
```

```
# Cross validate model with Kfold stratified cross val
kfold = StratifiedKFold(n_splits=10)
# Modeling step Test differents algorithms
random state = 2
classifiers = []
classifiers.append(SVC(random_state=random_state))
classifiers.append(DecisionTreeClassifier(random_state=random_state))
```

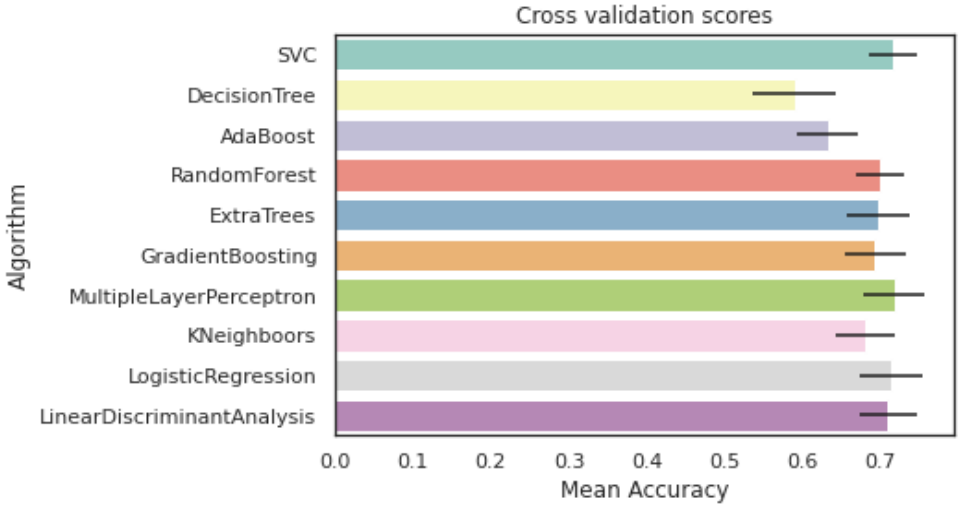
```
classifiers.append(AdaBoostClassifier(DecisionTreeClassifier(random_state=r
andom_state),random_state=random_state,learning_rate=0.1))
classifiers.append(RandomForestClassifier(random_state=random_state))
classifiers.append(ExtraTreesClassifier(random_state=random_state))
classifiers.append(GradientBoostingClassifier(random_state=random_state))
classifiers.append(MLPClassifier(random_state=random_state))
classifiers.append(KNeighborsClassifier())
classifiers.append(LogisticRegression(random_state = random_state))
classifiers.append(LinearDiscriminantAnalysis())

cv_results = []
for classifier in classifiers :
    cv_results.append(cross_val_score(classifier, X_train, y = y_train,
scoring = "accuracy", cv = kfold, n_jobs=4))

cv_means = []
cv_std = []
for cv_result in cv_results:
    cv_means.append(cv_result.mean())
    cv_std.append(cv_result.std())

cv_res = pd.DataFrame({"CrossValMeans":cv_means,"CrossValerrors":
cv_std,"Algorithm":["SVC","DecisionTree","AdaBoost",
"RandomForest","ExtraTrees","GradientBoosting","MultipleLayerPerceptron","K
Neighboors","LogisticRegression","LinearDiscriminantAnalysis"]})

g = sns.barplot("CrossValMeans","Algorithm",data = cv_res,
palette="Set3",orient = "h",**{'xerr':cv_std})
g.set_xlabel("Mean Accuracy")
g = g.set_title("Cross validation scores")
```



لذا فإن XGBoost لديه أفضل دقة بنسبة 72.4%.

المصدر:

https://www.kaggle.com/code/xmorra/player-performance

15) تحليل اختيارات تسديدات كوبي براينت Kobe Bryant Shot Selection Analysis

لقد أصيب جميع مشجعي كرة السلة، وبشكل عام جميع عشاق الرياضة، بالصدمة بسبب فقدان أحد أعظم اللاعبين على الإطلاقفي عام 2020 المأساوي. كان كوبي براينت Kobe Bryant لاعبًا غير متوقع قضى حياته المهنية بأكملهافي فريق لوس أنجلوس ليكرز Los Angeles Lakers، وفاز لهم بخمس بطولات.

كان المامبا الأسود أيضًا نجمًا 18 مرة وفاز بميداليتين ذهبيتينفي الألعاب الأولمبيةفي لندن وبكين ممثلاً لبلده: الولايات المتحدة.

مع كل هذه الإنجازات وحركاته التي لا تُنسى، يُعتبر واحدًا من الأفضلفي اللعبة. بعد الحدث المأساوي، تلقى كوبي التقدير والمودة وأحر وداع ممكن من المشجعينفي جميع أنحاء العالم.

في هذه المقالة، سأقوم بتحليل استكشافي exploratory analysis للتسديدات التي سددها هذا اللاعب طوال حياته المهنية بالكامل، بمافي ذلك التصورات المثيرة للاهتمام واستخراج بعض الأفكار عنها. علاوة على ذلك، سأقوم بعمل نموذج يتنبأ بما إذا كانت التسديدة ناجحة أم لا بالنظر إلى بعض ميزات هذه التسديدة نفسها. للقيام بذلك، سيتم تنفيذ مجموعة من النماذج المختلفة ensemble of different models. لذا إذا لم تكن على دراية بهذا النوع من الإجراءات، فاستمرفي القراءة، وسأشرح لك كل ما تحتاجه.

مجموعة البيانات

البيانات مستمدة من مسابقة Kaggle's Playground Prediction Competition، ويمكن العثور عليها هنا.

كما يشير وصف البيانات: تحتوي هذه البيانات على موقع وظروف كل هدف ميداني حاول كوبي براينت تسجيله خلال مسيرته التي استمرت 20 عامًا. وتتلخص المهمةفي التنبؤ بما إذا كانت السلة قد دخلت (shot_made_flag). وقد تمت إزالة 5000 من shot_made_flags وتم تمثيلها كقيم مفقودة missing valuesفي ملف csv. وهذه هي مجموعة التسديدات التجريبية التي يجب أن نقدم لها تنبؤًا.

أسماء الحقول واضحة بذاتها وتحتوي على السمات التالية:

```
action_type
combined_shot_type
game_event_id
game_id
```

```
lat
loc_x
loc_y
lon
minutes_remaining
period
playoffs
season
seconds_remaining
shot_distance
shot_made_flag (this is what you are predicting)
shot_type
shot_zone_area
shot_zone_basic
shot_zone_range
team_id
team_name
game_date
matchup
opponent
shot_id
```

تحميل المكتبات الضرورية

```
# For processing the data
import numpy as np
import pandas as pd

# Visualization tools
import matplotlib.pyplot as plt
import seaborn as sns
from matplotlib.lines import Line2D
%matplotlib inline
sns.set_style("white") # set style for seaborn plots

# Machine learning
from sklearn.decomposition import PCA, KernelPCA
from sklearn.preprocessing import MinMaxScaler
from sklearn.model_selection import KFold, cross_val_score, GridSearchCV
from sklearn.feature_selection import VarianceThreshold, RFE, SelectKBest,
chi2
from sklearn.metrics import make_scorer, log_loss
from sklearn.pipeline import Pipeline, FeatureUnion
from sklearn.neighbors import KNeighborsClassifier
from sklearn.discriminant_analysis import LinearDiscriminantAnalysis
from sklearn.svm import SVC
from sklearn.linear_model import LogisticRegression
from sklearn.naive_bayes import GaussianNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import (BaggingClassifier, ExtraTreesClassifier,
                              GradientBoostingClassifier, VotingClassifier,
                              RandomForestClassifier, AdaBoostClassifier)

# Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

تحميل مجموعة البيانات

كما هو مذكورفي قسم البيانات، تتكون مجموعة البيانات من ملف csv واحد فقط. هناك 5000 ملاحظة shot_made_flags كقيم مفقودة. تمثل هذه القيم مجموعة الاختبار الخاصة بنا، وهدفنا هنا هو التنبؤ بها.

```
df = pd.read_csv("../input/kobe-bryant-shot-selection/data.csv.zip")
```

لقد قمنا بتعيين الفهرس باستخدام العمود الموجود shot_id:

```
df.set_index('shot_id', inplace=True)
df.head()
```

shot_id	action_type	combined_shot_type	game_event_id	game_id	lat	loc_x	loc_y	lon	minutes_remaining	peri
1	Jump Shot	Jump Shot	10	20000012	33.9723	167	72	-118.1028	10	1
2	Jump Shot	Jump Shot	12	20000012	34.0443	-157	0	-118.4268	10	1
3	Jump Shot	Jump Shot	35	20000012	33.9093	-101	135	-118.3708	7	1
4	Jump Shot	Jump Shot	43	20000012	33.8693	138	175	-118.1318	6	1
5	Driving Dunk Shot	Dunk	155	20000012	34.0443	0	0	-118.2698	6	2

```
5 rows × 24 columns
```

تصحيح انواع المتغيرات

```
df.dtypes
```

```
action_type            object
combined_shot_type     object
game_event_id           int64
game_id                 int64
lat                   float64
loc_x                   int64
loc_y                   int64
lon                   float64
minutes_remaining       int64
period                  int64
playoffs                int64
season                 object
seconds_remaining       int64
shot_distance           int64
shot_made_flag        float64
shot_type              object
shot_zone_area         object
shot_zone_basic        object
shot_zone_range        object
team_id                 int64
team_name              object
game_date              object
matchup                object
opponent               object
dtype: object
```

أولاً، سنحول عمود الفترة period column إلى كائن. لن نجري عليه عمليات حسابية، لذا فليس من الضروري الاحتفاظ به كعدد صحيح.

ومن ناحية أخرى، هناك العديد من المتغيرات التي يمكن ترميزها كفئة category. سيسمح لنا هذا بالتفاعل مع إطار بيانات أكثر كفاءة من حيث سرعة التشغيل واستخدام الذاكرة.

```
df["period"] = df["period"].astype('object')

vars_to_category = ["combined_shot_type", "game_event_id", "game_id",
"playoffs",
                    "season", "shot_made_flag", "shot_type", "team_id"]
for col in vars_to_category:
    df[col] = df[col].astype('category')

# Let us check the final types
df.dtypes
```

```
action_type             object
combined_shot_type    category
game_event_id         category
game_id               category
lat                    float64
loc_x                    int64
loc_y                    int64
lon                    float64
minutes_remaining        int64
period                  object
playoffs              category
season                category
seconds_remaining        int64
shot_distance            int64
shot_made_flag        category
shot_type             category
shot_zone_area          object
shot_zone_basic         object
shot_zone_range         object
team_id               category
team_name               object
game_date               object
matchup                 object
opponent                object
dtype: object
```

ملخص سريع للبيانات

```
print("Dimensions of out DataFrame:", df.shape)
```

```
Dimensions of out DataFrame: (30697, 24)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30697 entries, 1 to 30697
Data columns (total 24 columns):
```

```
 #   Column               Non-Null Count  Dtype
---  ------               --------------  -----
 0   action_type          30697 non-null  object
 1   combined_shot_type   30697 non-null  category
 2   game_event_id        30697 non-null  category
 3   game_id              30697 non-null  category
 4   lat                  30697 non-null  float64
 5   loc_x                30697 non-null  int64
 6   loc_y                30697 non-null  int64
 7   lon                  30697 non-null  float64
 8   minutes_remaining    30697 non-null  int64
 9   period               30697 non-null  object
 10  playoffs             30697 non-null  category
 11  season               30697 non-null  category
 12  seconds_remaining    30697 non-null  int64
 13  shot_distance        30697 non-null  int64
 14  shot_made_flag       25697 non-null  category
 15  shot_type            30697 non-null  category
 16  shot_zone_area       30697 non-null  object
 17  shot_zone_basic      30697 non-null  object
 18  shot_zone_range      30697 non-null  object
 19  team_id              30697 non-null  category
 20  team_name            30697 non-null  object
 21  game_date            30697 non-null  object
 22  matchup              30697 non-null  object
 23  opponent             30697 non-null  object
dtypes: category(8), float64(2), int64(5), object(9)
memory usage: 4.4+ MB
```

كما نعلم، فإن shot_made_flag تحتوي على قيم فارغة تتوافق مع ملاحظات الاختبار. ولكن من المدهش أن هذا هو المتغير الوحيد الذي يحتوي على بيانات مفقودة، لذا لن تكون هناك حاجة إلى أي تضمين imputation.

```
df.describe(include=['number'])
```

	lat	loc_x	loc_y	lon	minutes_remaining	seconds_remaining	shot_distance
count	30697.000000	30697.000000	30697.000000	30697.000000	30697.000000	30697.000000	30697.000000
mean	33.953192	7.110499	91.107535	-118.262690	4.885624	28.365085	13.437437
std	0.087791	110.124578	87.791361	0.110125	3.449897	17.478949	9.374189
min	33.253300	-250.000000	-44.000000	-118.519800	0.000000	0.000000	0.000000
25%	33.884300	-68.000000	4.000000	-118.337800	2.000000	13.000000	5.000000
50%	33.970300	0.000000	74.000000	-118.269800	5.000000	28.000000	15.000000
75%	34.040300	95.000000	160.000000	-118.174800	8.000000	43.000000	21.000000
max	34.088300	248.000000	791.000000	-118.021800	11.000000	59.000000	79.000000

```
df.describe(include=['object', 'category'])
```

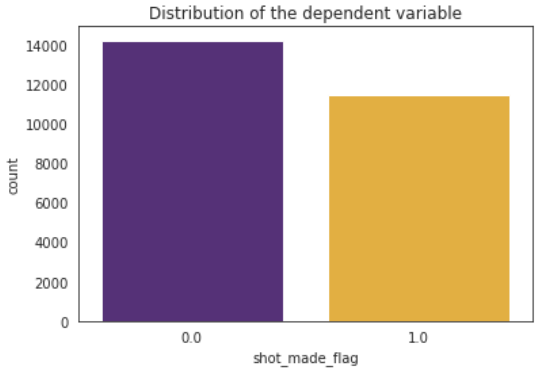
	action_type	combined_shot_type	game_event_id	game_id	period	playoffs	season	shot_made_flag	shot_type	shot
count	30697	30697	30697	30697	30697	30697	30697	25697.0	30697	3069
unique	57	6	620	1559	7	2	20	2.0	2	6
top	Jump Shot	Jump Shot	2	21501228	3	0	2005-06	0.0	2PT Field Goal	Cent
freq	18880	23485	132	50	8296	26198	2318	14232.0	24271	1345

بعض الاستكشاف

في هذا القسم، سنستكشف مجموعة البيانات الخاصة بنا بشكل أكبر.في المقام الأول، سنعرض التصورات visualizations، وهي طريقة فعالة للغاية للحصول على رؤى حول بياناتنا.

وبالمثل، سنحاول تحديد المتغيرات التي يمكن أن يكون لها تأثير كبير على إمكانية تفسير المتغير التابع dependant variable لدينا: shot_made_flag . نبدأ بتوزيع الفئة المستهدفة target class distribution لدينا:

```
ax = plt.axes()
sns.countplot("shot_made_flag", data=df, ax=ax, palette=("#552583",
"#FDB927"))
ax.set_title("Distribution of the dependent variable")
plt.show()
```

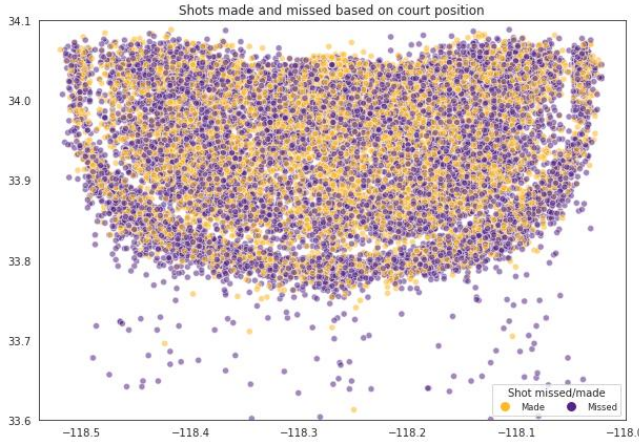


في البداية، يمكننا أن نرى أن المتغير المستهدف موزع بشكل موحد إلى حد ما. هناك فرص ضائعة أكثر من الفرص التي تم تحقيقها، ولكننا لن نحتاج إلى تنفيذ أي إجراءات للتعامل مع مجموعات البيانات غير المتوازنة.

نستمر الآنفي عرض التسديدات التي تم تحقيقها أو التي أخطأت فيما يتعلق بالموضع الذي تم التقاطها فيه. سيعرض الرسم البياني التالي هذا بالضبط:

```
fig, ax = plt.subplots(1, 1, figsize=(10, 7))
scatter = sns.scatterplot(x=df["lon"], y=df["lat"],
hue=df['shot_made_flag'],
                                    alpha=0.55, ax=ax, palette=("#552583",
"#FDB927"))
scatter.set_xlim(left=-118.54, right=-118)
scatter.set_ylim(bottom=33.6, top=34.1)
ax.set_title("Shots made and missed based on court position")
ax.set_xlabel("")
ax.set_ylabel("")
legend_elemnts = [Line2D([0], [0], marker="o", color='w', label="Made",
                         markerfacecolor="#FDB927", markersize=10),
                  Line2D([0], [0], marker="o", color='w', label="Missed",
                         markerfacecolor="#552583", markersize=10)]
```

```
plt.legend(handles=legend_elemnts, title="Shot missed/made",
           ncol=2, fontsize='small', fancybox=True);
```



لاحظ أن هناك مجموعة واضحة من النقاط المسجلة بجوار السلة. ومن ناحية أخرى، هناك اتجاه واضح في المنطقة الوسطى: يبدو أن كوبي كان أكثر دقة هناك. أعتقد أن الجانب الأيمن من الملعب يحتوي على القليل من اللون الأصفر، وإن كان أقل وضوحًا. دعونا نتحقق من هذين الافتراضين الأخيرين:

```
# We don't want to modify the original DataFrame
subset = df.copy()
subset["x_zones"] = pd.cut(df["loc_x"], bins=25)
df_grouped1 = subset.groupby("x_zones").agg({"shot_made_flag":
"count"}).reset_index()
df_shots_made = subset[subset["shot_made_flag"]==1]
df_grouped2 = df_shots_made.groupby("x_zones").agg({"shot_made_flag":
"count"}).reset_index()
proportions = round(df_grouped2["shot_made_flag"] /
df grouped1["shot made flag"], 2)

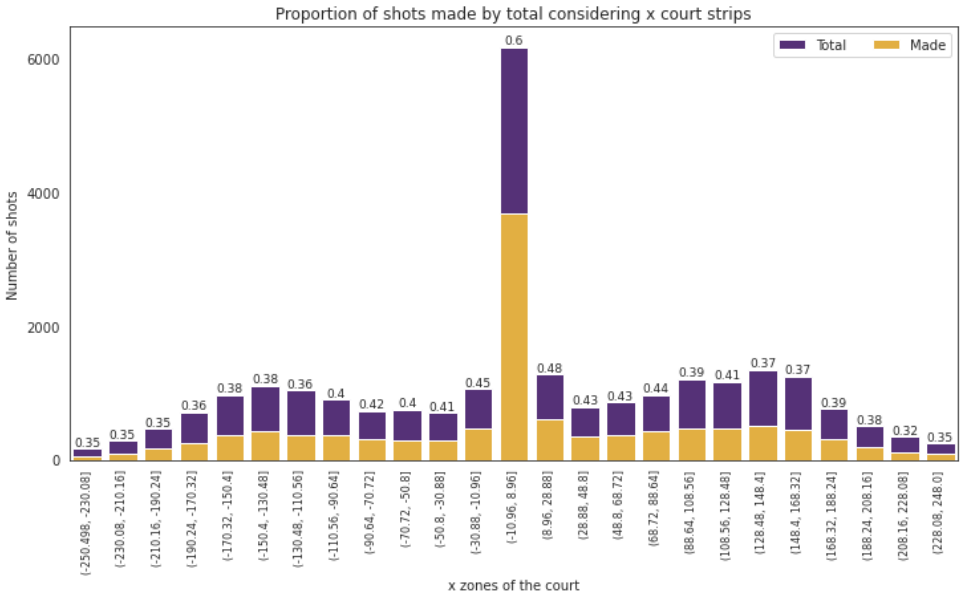
f, ax = plt.subplots(figsize=(12, 6))
# Plot total shots
g1 = sns.barplot(x="x_zones", y="shot_made_flag", data=df_grouped1,
                 label="Total", color="#552583")

# Plot shots made
g2 = sns.barplot(x="x_zones", y="shot_made_flag", data=df_grouped2,
                 label="Made", color="#FDB927")

idx = 0
for p in g1.patches:
    g1.annotate(proportions[idx],
                (p.get_x() + p.get_width() / 2., p.get_height()-80),
                 ha="center", va="center",
                 xytext=(0, 9), fontsize=9,
                 textcoords="offset points")
    if idx < 24: idx += 1
    else: break

plt.yticks(ticks=[0, 2000, 4000, 6000])
plt.xticks(fontsize=8, rotation=90)
```

```
ax.set_title("Proportion of shots made by total considering x court
strips")
ax.set_xlabel("x zones of the court")
ax.set_ylabel("Number of shots")
ax.legend(ncol=2, loc="upper right", frameon=True);
```



من الواضح الآن أن التسديدات المركزية central shots تتمتع بدقة أكبر من التسديدات الجانبية. وتحديدًا، كانت 60% من التسديدات التي تم تنفيذهافي الشريط المركزي ناجحة. أما التسديداتفي الزاوية Shots in the corner فهي التي كان كوبي أقل دقة فيها، وهي ظاهرة طبيعية بين الغالبية العظمى من اللاعبين.

والأمر المثير للاهتمام هو أن هناك أداءً أفضلفي بعض المناطق الجانبية مقارنة بمناطق أخرى أقرب إلى المركز. ويبدو أيضًا أن التسديدات من الملعب الأيمن كانت نتائجها أفضل بهامش ضيق.

```
def make_zone_scatter(var, ax):
    sns.scatterplot(x=df["lon"], y=df["lat"],
                    hue=df[var], ax=ax,
                    palette="Dark2")
    ax.legend(ncol=len(df[var].unique())//3, fontsize='small',
fancybox=True)


def make_zone_countplot(var, ax):
    sns.countplot(x=var, data=df,
              order=df[var].value_counts().index,
              ax=ax, palette="Dark2")
    ax.set_xlabel("")
    ax.set_xticklabels(df[var].unique(), fontsize=8, rotation=90)


def make_acc_lollipop(var, ax):
    subset = df[[var, "shot_made_flag"]].dropna()
```

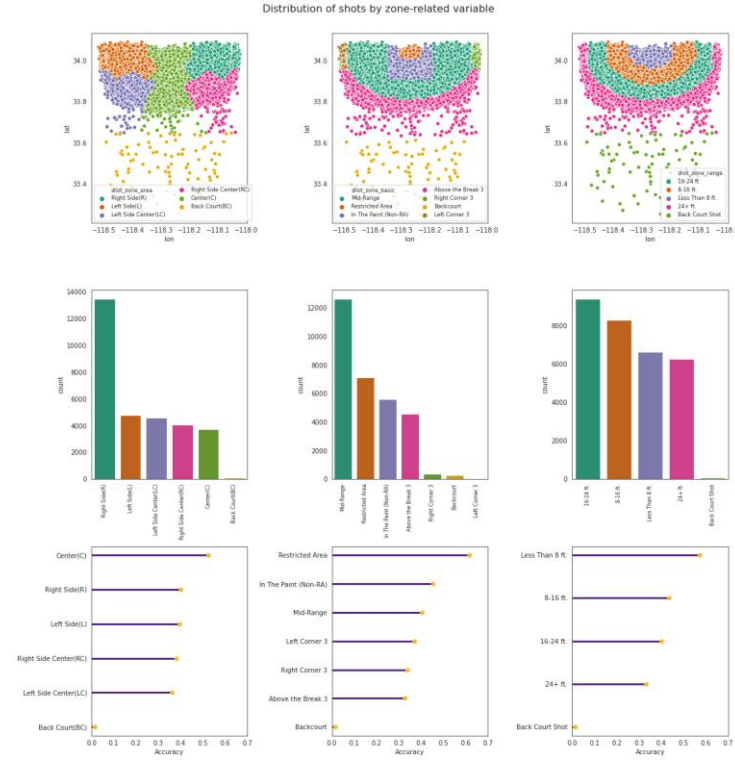
```
    subset["shot_made_flag"] = pd.to_numeric(subset["shot_made_flag"])
    df_grouped = subset.groupby(var).agg({"shot_made_flag":
"mean"}).reset_index()
    df_grouped = df_grouped.sort_values(by="shot_made_flag")
    ax.hlines(y=df_grouped[var], xmin=0,
              xmax=df_grouped["shot_made_flag"], color="#552583",
linewidth=3)
    ax.plot(df_grouped["shot_made_flag"], range(0,len(df_grouped.index)),
"o", color="#FDB927")
    ax.set_xlim([0, .7])
    ax.set_xlabel("Accuracy")


f, ((ax0, ax1, ax2), (ax3, ax4, ax5), (ax6, ax7, ax8)) = plt.subplots(3, 3,
figsize=(16, 16))
make_zone_scatter("shot_zone_area", ax0)
make_zone_scatter("shot_zone_basic", ax1)
make_zone_scatter("shot_zone_range", ax2)

make_zone_countplot("shot_zone_area", ax3)
make_zone_countplot("shot_zone_basic", ax4)
make_zone_countplot("shot_zone_range", ax5)

make_acc_lollipop("shot_zone_area", ax6)
make_acc_lollipop("shot_zone_basic", ax7)
make_acc_lollipop("shot_zone_range", ax8)

f.tight_layout()
f.suptitle("Distribution of shots by zone-related variable", fontsize=16,
y=1.03);
```



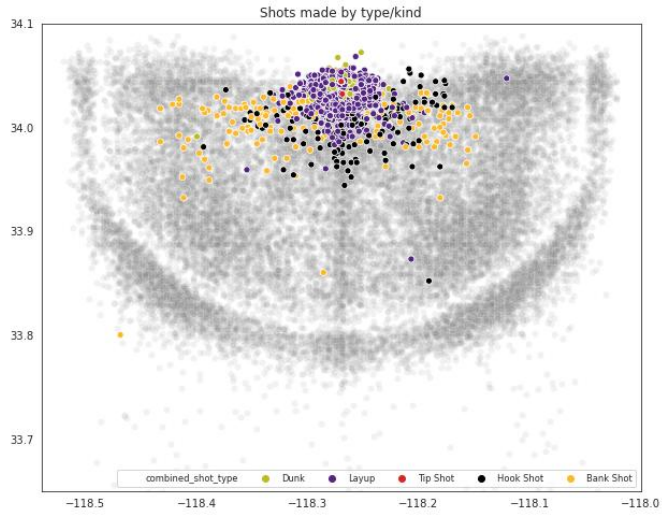
باستخدام هذا الشكل المجمع combined figure، يمكننا فهم كيفية وضع المتغيرات المتعلقة بالمنطقة بين ملعب كرة السلة، وكيفية توزيعها (أي عدد التسديدات التي حدثتفي كل منطقة)، وكيف تؤثر هذه المناطق على المتغير الثنائي التابع shot_made_flag.

بالإضافة إلى ذلك، تم تصنيف أنواع مختلفة من التسديداتفي المتغيرات combined_shot_type وaction_type. هنا نفحص هذه الميزات، ونقدم تأثيرها على دقة مقياس التسديد.

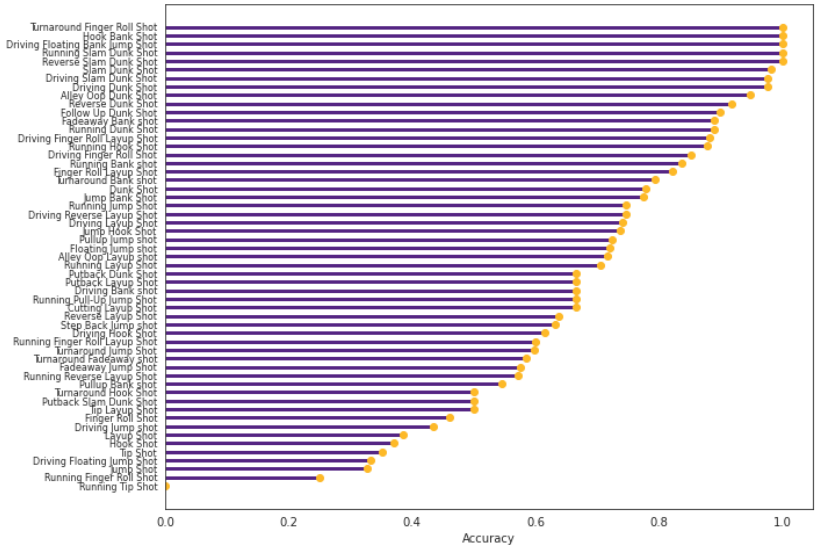
```
fig, ax = plt.subplots(1, 1, figsize=(10, 8))
jump_shot_df = df[df["combined_shot_type"] == "Jump Shot"]
scatter_jumpshots = sns.scatterplot(x=jump_shot_df["lon"],
y=jump_shot_df["lat"],
                                    alpha=0.1, ax=ax, color="grey")

not_jump_shot_df = df[df["combined_shot_type"] != "Jump Shot"]
scatter = sns.scatterplot(x=not_jump_shot_df["lon"],
y=not_jump_shot_df["lat"],
                          hue=not_jump_shot_df["combined_shot_type"],
                          palette=["C8", "#552583", "C3", "#000000",
"#FDB927"],
                          ax=ax)

scatter.set_xlim(left=-118.54, right=-118)
scatter.set_ylim(bottom=33.65, top=34.1);
ax.set_title("Shots made by type/kind")
ax.set_xlabel("")
ax.set_ylabel("")
plt.legend(ncol=len(df["combined_shot_type"].unique())+1, fontsize='small',
fancybox=True);
```



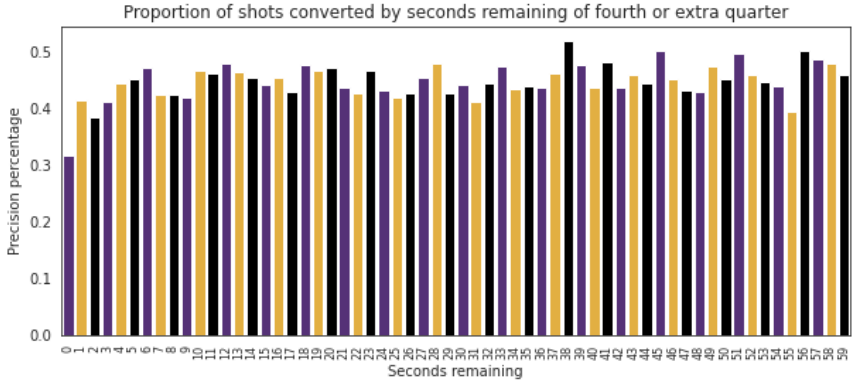

```
f, ax = plt.subplots(figsize=(10,8))
make_acc_lollipop("action_type", ax)
ax.set_xlim([0, 1.05])
ax.tick_params(axis="y", labelsize=8)
```



نواصل استكشاف الدقة، ونأخذ الآنفي الاعتبار الثواني المتبقية من الربع الرابع الأخير أو الثواني الإضافية. من المفترض أن تكون هذه التسديدات مصحوبة بقدر كبير من الضغط، وهو ما قد يؤدي إلى أداء أسوأ، لكن كوبي يتمتع بإحصائيات جيدة بشكل عام. على الرغم من وجود انخفاضفي الدقة بعد الثواني الخمس المتبقية.

```
subset = df[df["period"]<=4][["seconds_remaining",
"shot_made_flag"]].dropna()
subset = pd.DataFrame(subset, dtype=int)
df_grouped3 = subset.groupby("seconds_remaining").agg({"shot_made_flag":
"mean"}).reset_index()

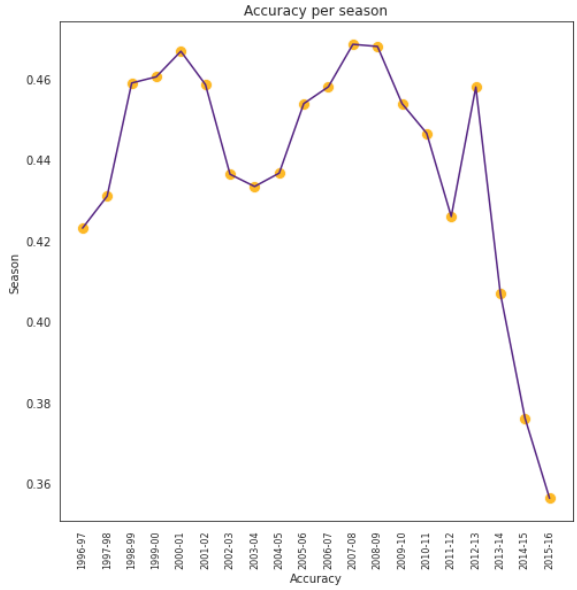
fig, ax = plt.subplots(1, 1, figsize=(10, 4))
sns.barplot(x="seconds_remaining", y="shot_made_flag", data=df_grouped3,
            palette=("#552583", "#FDB927", "#000000"))
ax.set_title("Proportion of shots converted by seconds remaining of fourth
or extra quarter")
ax.set_xlabel("Seconds remaining")
ax.set_ylabel("Precision percentage")
plt.xticks(fontsize=8, rotation=90);
```



انخفض أداء كوبيفي آخر ثلاث سنوات لهفي الدوري. على الأقل من حيث دقة التسديد، يوضح الرسم البياني التالي ذلك:

```
subset4 = df[["season", "shot_made_flag"]].dropna()
subset4["shot_made_flag"] = pd.to_numeric(subset4["shot_made_flag"])
df_grouped4 = subset4.groupby("season").agg({"shot_made_flag":
"mean"}).reset_index()

f, ax = plt.subplots(1, 1, figsize=(8,8))
sns.lineplot(x="season", y="shot_made_flag", data=df_grouped4,
color="#552583", ax=ax);
sns.scatterplot(x="season", y="shot_made_flag", data=df_grouped4, s=100,
color="#FDB927", ax=ax)
ax.set_title("Accuracy per season")
ax.set_xlabel("Accuracy")
ax.set_ylabel("Season")
plt.xticks(fontsize=8, rotation=90);
```

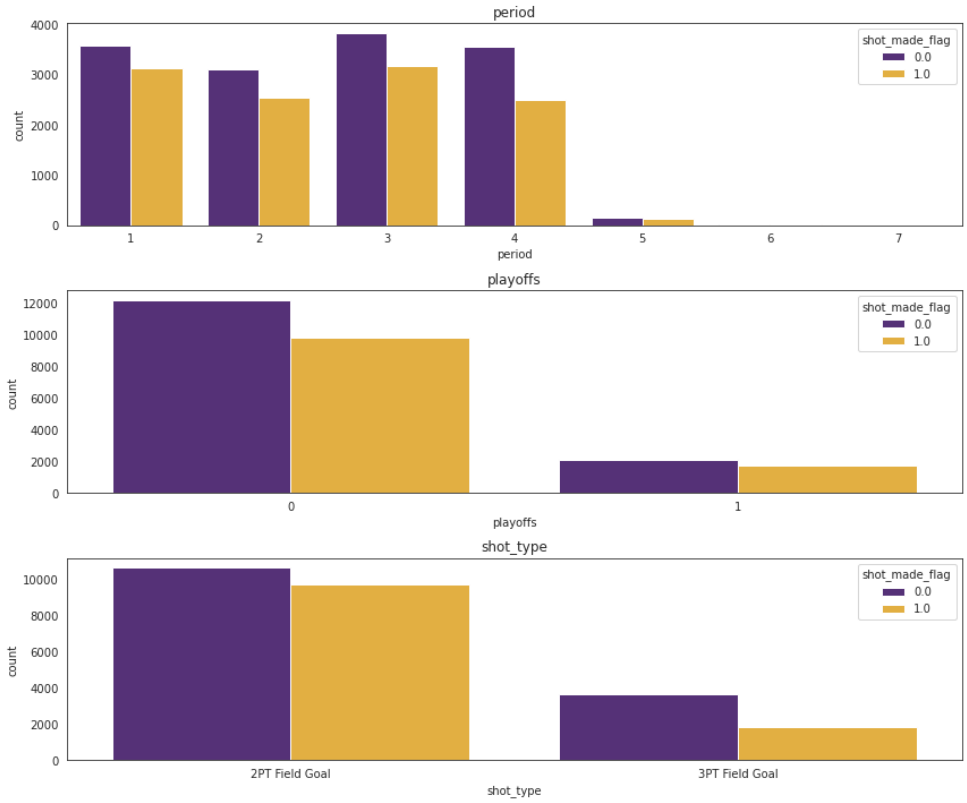


أخيرًا، سنستعرض ثلاث ميزات إضافية: الفترة period، والتصفيات playoffs، ونوع التسديد shot_type. ويمكننا استخلاص بعض الأفكار المدهشة والقيمة منها: كان لدى كوبي إحصائيات دقة مذهلةفي التصفيات وفي الأزمنة الإضافية، وكان لاعبًا رائعًا.

```
f, ax = plt.subplots(3, figsize=(12, 10))

for var, i in zip(["period", "playoffs", "shot_type"], range(0,3)):
    sns.countplot(x=var, hue="shot_made_flag", data=df, ax=ax[i],
palette=("#552583", "#FDB927"))
    ax[i].set_title(var)

plt.tight_layout()
plt.show()
```



معالجة البيانات مسبقًا

سنقوم الآن بإجراء بعض التعديلات على البيانات. وللحفاظ على سلامة إطار البيانات الأصلي، سننسخه إلى إطار جديد يسمى: copy_df. ويعتبر هذا ممارسة جيدة ويمكن أن يكون مفيدًافي منع المشكلات غير المرغوب فيها.

```
copy_df = df.copy()
target = copy_df['shot_made_flag'].copy()
```

إزالة الأعمدة غير المفيدة

لنبدأفي إزالة بعض الأعمدة التي لا تقدم أي فائدة.

- team_id و team_name هما ميزتان عديمتا الفائدة تمامًا نظرًا لأن Kobe لعبفي فريق واحد فقط وهو فريق L.A. Lakers: قيمهما لها قيمة فريدة واحدة فقط.
- بالنسبة إلى game_id و game_event_id، فهما متغيران مستقلان لهما علاقة null فيما إذا كانت التسديدة ناجحة أم لا. سيضيفان ضوضاء إلى نموذجنا.
- lat و long مرتبطان ارتباطًا وثيقًا بـ loc_x و loc_y، يمكننا إضافة مشكلات التعدد الخطي multicollinearity problems إلى مجموعتنا.

- في النهاية، shot_made_flag هو المتغير التابع لدينا، وقد قمنا بالفعل بتخزينهفي سلسلة الهدف.

```
vars_to_remove = ["team_id", "team_name", "game_id", "game_event_id",
                  "lat", "lon", "shot_made_flag"]

for var in vars_to_remove:
    copy_df = copy_df.drop(var, axis=1)
```

تحويل المتغيرات

أنواع الإجراءات

هناك الكثير من أنواع الإجراءات action. نحتاج إلى ترميز هذه القيم بعدد أقل من المرات كفئة جديدة: "Other" أو "Rare actions". وإلا، فعندما نستخدم الترميز واحد ساخن one-hot-encode، سنشهد زيادة كبيرةفي أبعاد الأعمدة.

```
pd.DataFrame({"counts":
copy_df["action_type"].value_counts().sort_values()[:25]})
```

	counts
Cutting Finger Roll Layup Shot	1
Running Slam Dunk Shot	1
Driving Floating Bank Jump Shot	1
Turnaround Fadeaway Bank Jump Shot	1
Turnaround Finger Roll Shot	2
Putback Slam Dunk Shot	2
Running Tip Shot	2
Tip Layup Shot	2
Running Finger Roll Shot	4
Running Pull-Up Jump Shot	4
Driving Floating Jump Shot	5
Driving Bank shot	5
Hook Bank Shot	5
Putback Dunk Shot	5
Cutting Layup Shot	6
Running Finger Roll Layup Shot	6
Running Reverse Layup Shot	11
Pullup Bank shot	12
Turnaround Hook Shot	14
Driving Hook Shot	14
Follow Up Dunk Shot	15
Putback Layup Shot	15
Reverse Slam Dunk Shot	16
Running Dunk Shot	19
Jump Hook Shot	24

```
rare_action_types =
copy_df["action_type"].value_counts().sort_values().index.values[:20]
copy_df.loc[copy_df["action_type"].isin(rare_action_types), "action_type"]
= "Other"
```

تاريخ المباراة

سنفصل الشهر والسنة عن التاريخ. وكما سنرى لاحقًا، سيساهم هذافي توضيح الهدف.

```
copy_df["game_date"] = pd.to_datetime(copy_df["game_date"])
copy_df["game_year"] = copy_df["game_date"].dt.year
copy_df["game_month"] = copy_df["game_date"].dt.month
copy_df = copy_df.drop("game_date", axis=1)
```

الثواني الأخيرة

كما لاحظنافي قسم التحليل الاستكشافي exploratory analysis، كان هناك انخفاض كبيرفي التسديدات التي تم التقاطها قبل أقل من 5 ثوانٍ متبقية. وعلى نحو مماثل، كانت الدقة مع المزيد من الثواني موحدة تمامًا. سنقوم بإجراء تحويل لتضمين هذه الظاهرة وتقليل عدد الأعمدة المستقبلية.

```
copy_df["seconds_from_period_end"] = 60 * copy_df["minutes_remaining"] +
copy_df["seconds_remaining"]
copy_df["last_5_sec_in_period"] = copy_df["seconds_from_period_end"] < 5

# We can drop the rest of time related fields
copy_df = copy_df.drop("minutes_remaining", axis=1)
copy_df = copy_df.drop("seconds_remaining", axis=1)
copy_df = copy_df.drop("seconds_from_period_end", axis=1)
```

مناطق x وy

لقد قمنا بالفعل بشيء مماثلفي قسم تصور البيانات. الآن سنقوم بتضمين هذه الشرائطفي مجموعة التدريب الخاصة بنا لمحور x ومحور y. لكننا لن نتخلى عن loc_x وloc_y.

```
copy_df["x_zones"] = pd.cut(copy_df["loc_x"], bins=25)
copy_df["y_zones"] = pd.cut(copy_df["loc_y"], bins=25)
```

المباريات على أرضنا

سيكون الأمر أكثر وضوحًا إذا قمنا بتعيين متغير ثنائي سيحدد ما إذا كانت المباراة تُلعب على أرضنا أو خارجها بالقيم الكلاسيكية 1 أو 0.

```
copy_df["home_play"] = copy_df["matchup"].str.contains("vs").astype("int")
copy_df = copy_df.drop("matchup", axis=1)
```

ترميز المتغيرات الفئوية

أخيرًا أصبحنافي وضع يسمح لنا بترميز متغيراتنا الفئوية categorical variables بطريقة الترميز واحد ساخن.

```
pd.get_dummies(copy_df["action_type"]).add_prefix("{}#".format("action_type
"))

categorial_vars = [
    'action_type', 'combined_shot_type', 'period', 'season', 'shot_type',
    'shot_zone_area', 'shot_zone_basic', 'shot_zone_range', 'game_year',
    'game_month', 'opponent', 'loc_x', 'loc_y', 'x_zones', 'y_zones']

for var in categorial_vars:
    dummies = pd.get_dummies(copy_df[var])
    dummies = dummies.add_prefix("{}#".format(var))
    copy_df.drop(var, axis=1, inplace=True)
```

```
    copy_df = copy_df.join(dummies)
```

مجموعات منفصلة للتدريب والاختبار

```
missing = target.isnull()

data_submit = copy_df[missing]
X = copy df[~missing]
Y = target[~missing]
```

```
print(X.shape, Y.shape)
```

```
(25697, 1154) (25697,)
```

```
copy_df.shape
```

```
(30697, 1154)
```

اختيار الميزة

عندما قمنا بالترميز واحد ساخن، قمنا بزيادة أعمدة مجموعتنا بشكل مفاجئ، وقمنا بالرسم من أقل من 1106 ميزة، والآن لدينا 208. وقد حدث هذا حتى مع كل العمل الشاق المتمثلفي تجاهل المتغيرات وتحويلها الذي قمنا به من قبل.

حسنًا، هذا أمر طبيعي إلى حد ما عندما يكون عدد الفئاتفي المتغيرات مرتفعًا. لحسن الحظ، لدينا ملاحظات كافية للتعامل مع كل هذه الأعمدة؛ والأهم من ذلك، مع التقنيات لتقليلها. وسنفعل ذلكفي هذا القسم من خلال تحديد تلك المتغيرات الأكثر إفادة.

لنبدأ بهذا التقليل للميزات. سنطبق تقنيات مختلفة ونجمعهافي مرحلة الاختيار النهائية.

عتبة التباين

سنجد جميع الميزات التي تحتوي على تباين مجموعة تدريب أكبر من 90%.

```
threshold = 0.9
vt = VarianceThreshold().fit(X)

# Find feature names
feat_var_threshold = copy_df.columns[vt.variances_ > threshold * (1-
threshold)]
feat_var_threshold
```

```
Index(['playoffs', 'shot_distance', 'home_play', 'action_type#Jump Shot',
       'combined_shot_type#Jump Shot', 'combined_shot_type#Layup',
'period#1',
       'period#2', 'period#3', 'period#4', 'shot_type#2PT Field Goal',
       'shot_type#3PT Field Goal', 'shot_zone_area#Center(C)',
       'shot_zone_area#Left Side Center(LC)', 'shot_zone_area#Left
Side(L)',
       'shot_zone_area#Right Side Center(RC)', 'shot_zone_area#Right
Side(R)',
       'shot_zone_basic#Above the Break 3',
```

```
       'shot_zone_basic#In The Paint (Non-RA)', 'shot_zone_basic#Mid-
Range',
       'shot_zone_basic#Restricted Area', 'shot_zone_range#16-24 ft.',
       'shot_zone_range#24+ ft.', 'shot_zone_range#8-16 ft.',
       'shot_zone_range#Less Than 8 ft.', 'game_month#1', 'game_month#2',
       'game_month#3', 'game_month#4', 'game_month#11', 'game_month#12',
       'loc_x#0', 'loc_y#0', 'x_zones#(-10.96, 8.96]', 'y_zones#(-10.6,
22.8]',
       'y_zones#(22.8, 56.2]', 'y_zones#(123.0, 156.4]'],
      dtype='object')
```

أهم الميزات

تتيح لنا RandomForestClassifier معرفة أهمية الميزة feature's importances. وفقًا لها، سنختار أفضل 30 ميزة.

```
model = RandomForestClassifier()
model.fit(X, Y)

feature_imp = pd.DataFrame(model.feature_importances_, index=X.columns,
columns=["importance"])
feat_imp_30 = feature_imp.sort_values("importance",
ascending=False).head(30).index
feat_imp_30
```

```
Index(['action type#Jump Shot', 'shot distance', 'home play',
       'action_type#Layup Shot', 'period#3', 'period#1', 'period#2',
       'period#4', 'combined_shot_type#Dunk', 'game_month#1',
'game_month#3',
       'game_month#12', 'game_month#2', 'game_month#4',
       'action type#Driving Layup Shot', 'game month#11', 'playoffs',
       'action_type#Running Jump Shot', 'opponent#PHX', 'opponent#DEN',
       'opponent#SAC', 'opponent#HOU', 'opponent#SAS',
       'x_zones#(-10.96, 8.96]', 'opponent#POR', 'opponent#MIN',
       'y_zones#(-10.6, 22.8]', 'game_month#5', 'opponent#UTA',
       'opponent#LAC'],
      dtype='object')
```

اختيار السمات أحادية المتغير

باستخدام هذا الإجراء، سنختار أيضًا أفضل 30 سمة ولكن باستخدام اختبار كاي2. يجب أن تكون السمات موجبة قبل تطبيق هذا الاختبار.

```
X_minmax = MinMaxScaler(feature_range=(0,1)).fit_transform(X)
X_scored = SelectKBest(score_func=chi2, k="all").fit(X_minmax, Y)
feature_scoring = pd.DataFrame({
    "feature": X.columns,
    "score": X_scored.scores_
})

feat scored 30 = feature scoring.sort values("score",
ascending=False).head(30)["feature"].values
feat_scored_30
```

```
array(['combined_shot_type#Dunk', 'action_type#Jump Shot',
       'shot_zone_basic#Restricted Area', 'x_zones#(-10.96, 8.96]',
```

```
       'action_type#Driving Layup Shot', 'loc_x#0', 'loc_y#0',
       'shot_zone_range#Less Than 8 ft.', 'y_zones#(-10.6, 22.8]',
       'action_type#Slam Dunk Shot', 'shot_type#3PT Field Goal',
       'action_type#Driving Dunk Shot', 'shot_zone_area#Center(C)',
       'action_type#Running Jump Shot', 'shot_zone_range#24+ ft.',
       'shot_zone_basic#Above the Break 3', 'combined_shot_type#Layup',
       'combined_shot_type#Jump Shot', 'last_5_sec_in_period',
       'action_type#Jump Bank Shot', 'action_type#Pullup Jump shot',
       'shot_zone_area#Left Side Center(LC)', 'action_type#Dunk Shot',
       'action_type#Alley Oop Dunk Shot', 'y_zones#(223.2, 256.6]',
       'shot_distance', 'action_type#Turnaround Jump Shot',
       'shot_type#2PT Field Goal', 'y_zones#(189.8, 223.2]',
       'shot_zone_basic#Mid-Range'], dtype=object)
```

إزالة الميزة المتكررة

نقوم الآن باختيار أفضل 30 ميزة باستخدام إزالة الميزة المتكررة recursive feature elimination (RFE) مع نموذج الانحدار اللوجستي logistic regression model.

```
# Running time can take several minutes
# You can ignore this method and don't include it in the final feature
selection
rfe = RFE(LogisticRegression(), 30)
rfe.fit(X, Y)

feature_rfe_scoring = pd.DataFrame({
    "feature": X.columns,
    "score": rfe.ranking_
})

feat_rfe_30 = feature_rfe_scoring[feature_rfe_scoring["score"] ==
1]["feature"].values
feat_rfe_30
```

```
array(['action_type#Driving Dunk Shot',
       'action_type#Driving Finger Roll Layup Shot',
       'action_type#Dunk Shot', 'action_type#Fadeaway Bank shot',
       'action_type#Hook Shot', 'action_type#Jump Shot',
       'action_type#Layup Shot', 'action_type#Running Hook Shot',
       'action_type#Slam Dunk Shot', 'combined_shot_type#Dunk',
       'combined_shot_type#Tip Shot', 'shot_zone_area#Back Court(BC)',
       'shot_zone_range#Back Court Shot', 'loc_x#-229', 'loc_x#-137',
       'loc_x#-118', 'loc_x#-53', 'loc_x#-43', 'loc_x#-22', 'loc_x#63',
       'loc_x#122', 'loc_x#185', 'loc_x#210', 'loc_x#211', 'loc_x#245',
       'loc_y#65', 'loc_y#67', 'y_zones#(290.0, 323.4]',
       'y_zones#(356.8, 390.2]', 'y_zones#(390.2, 423.6]'], dtype=object)
```

اختيار الميزة النهائية

أخيرًا، سنحصل على اختيارنا للميزات من خلال دمج كل الطرق المذكورة أعلاه. باختصار، سنحتفظ بالمتغيرات التي تظهر على الأقل كأفضل متغيرفي إحدى التقنيات.

```
features = np.hstack([
    feat_var_threshold,
    feat_imp_30,
    feat scored 30,
    feat_rfe_30
])
```

```
features = np.unique(features)
print("Final features set:\n")
for f in features:
    print("\t-{}".format(f))
```

```
Final features set:

          -action_type#Alley Oop Dunk Shot
          -action_type#Driving Dunk Shot
          -action_type#Driving Finger Roll Layup Shot
          -action_type#Driving Layup Shot
          -action type#Dunk Shot
          -action_type#Fadeaway Bank shot
          -action_type#Hook Shot
          -action_type#Jump Bank Shot
          -action_type#Jump Shot
          -action_type#Layup Shot
          -action_type#Pullup Jump shot
          -action_type#Running Hook Shot
          -action_type#Running Jump Shot
          -action_type#Slam Dunk Shot
          -action_type#Turnaround Jump Shot
          -combined shot type#Dunk
          -combined_shot_type#Jump Shot
          -combined_shot_type#Layup
          -combined_shot_type#Tip Shot
          -game_month#1
          -game_month#11
          -game month#12
          -game_month#2
          -game_month#3
          -game_month#4
          -game_month#5
          -home_play
          -last 5 sec in period
          -loc_x#-118
          -loc_x#-137
          -loc_x#-22
          -loc_x#-229
          -loc_x#-43
          -loc x#-53
          -loc_x#0
          -loc_x#122
          -loc_x#185
          -loc_x#210
          -loc_x#211
          -loc_x#245
          -loc_x#63
          -loc_y#0
          -loc_y#65
          -loc_y#67
          -opponent#DEN
          -opponent#HOU
          -opponent#LAC
          -opponent#MIN
          -opponent#PHX
          -opponent#POR
          -opponent#SAC
          -opponent#SAS
          -opponent#UTA
```

```
          -period#1
          -period#2
          -period#3
          -period#4
          -playoffs
          -shot_distance
          -shot_type#2PT Field Goal
          -shot_type#3PT Field Goal
          -shot_zone_area#Back Court(BC)
          -shot_zone_area#Center(C)
          -shot_zone_area#Left Side Center(LC)
          -shot_zone_area#Left Side(L)
          -shot_zone_area#Right Side Center(RC)
          -shot_zone_area#Right Side(R)
          -shot_zone_basic#Above the Break 3
          -shot_zone_basic#In The Paint (Non-RA)
          -shot_zone_basic#Mid-Range
          -shot_zone_basic#Restricted Area
          -shot_zone_range#16-24 ft.
          -shot_zone_range#24+ ft.
          -shot_zone_range#8-16 ft.
          -shot_zone_range#Back Court Shot
          -shot_zone_range#Less Than 8 ft.
          -x_zones#(-10.96, 8.96]
          -y_zones#(-10.6, 22.8]
          -y_zones#(123.0, 156.4]
          -y_zones#(189.8, 223.2]
          -y_zones#(22.8, 56.2]
          -y_zones#(223.2, 256.6]
          -y_zones#(290.0, 323.4]
          -y_zones#(356.8, 390.2]
          -y_zones#(390.2, 423.6]
```

إعداد مجموعة البيانات لمزيد من التحليل

```
copy_df = copy_df.loc[:, features]
data_submit = data_submit.loc[:, features]
X = X.loc[:, features]

print("Clean dataset shape: {}".format(copy_df.shape))
print("Subbmitable dataset shape: {}".format(data_submit.shape))
print("Train features shape: {}".format(X.shape))
print("Target label shape: {}".format(Y.shape))
```

```
Clean dataset shape: (30697, 85)
Subbmitable dataset shape: (5000, 85)
Train features shape: (25697, 85)
Target label shape: (25697,)
```

هنا أعرض لك الإصدار الفعلي من sklearn المستخدم للمساعدةفي حل مشكلات التوافق. بعد ذلك، نحدد بعض المتغيرات التي سنستخدمها خلال بناء النموذج. الأول هو البذرة العشوائية random seed: للحصول على نتائج قابلة للتكرار أمر لا بد منه.

تم تعيين عدد المعالجات على ـ 1، وهذا يعني أن جهاز الكمبيوتر الخاص بك سيستخدم جميع أنويته لمعالجة الكود بالتوازي. n_folds هو عدد الأقسام التي نريدها عندما نجري التحقق المتبادل cross-validation. خطأ السجل Log loss هي المقياس المختار للحصول على أداء تسجيل النماذج.

```
seed = 2666
processors = -1
num_folds = 3
scoring="neg_log_loss"

kfold = KFold(n_splits=num_folds, random_state=seed)
```

فحص عشوائي للخوارزميات

سنقوم الآن بإعداد بعض النماذج الأساسية بسرعة ورؤية كيفية سلوكهافي مجموعة البيانات الخاصة بنا.

```
models = []
models.append(("LR", LogisticRegression()))
models.append(("LDA", LinearDiscriminantAnalysis()))
models.append(("K-NN", KNeighborsClassifier(n_neighbors=5)))
models.append(("CART", DecisionTreeClassifier()))
models.append(("NB", GaussianNB()))


results = []
names = []
for name, model in models:
    cv_results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
    results.append(cv_results)
    names.append(name)
    print("{0}:({1:.3f}) +/- ({2:.3f})".format(name, cv_results.mean(),
cv_results.std()))
```

```
LR:(-0.613) +/- (0.002)
LDA:(-0.613) +/- (0.002)
K-NN:(-2.102) +/- (0.053)
CART:(-12.638) +/- (0.418)
NB:(-6.527) +/- (1.896)
```

من خلال النظر إلى هذه النتائج، نجد أن الانحدار اللوجستي Logistic Regression وتحليل التمييز الخطي Linear Discriminant Analysis يقدمان نتائج جيدة ويستحقان مزيداً من الفحص.

ولكن بصرف النظر عن هذه الخوارزميات البسيطة، فلنلق نظرة على بعض نماذج المجموعة قبل أن نرى ما إذا كان بوسعنا العثور على بعض النماذج الأكثر إثارة للاهتمام:

تجميع النماذج

التجميع (التجميع التمهيدي)

يتضمن التجميع (التجميع التمهيدي) Bagging (Bootstrap Aggregation) أخذ عينات متعددة مع الاستبدال من مجموعة البيانات التدريبية، وتدريب نموذج لكل منها. يتم حساب متوسط التنبؤ بالإخراج النهائي عبر تنبؤات جميع النماذج المستندة إلى العينات sampled-based-models.

Bagged Decision Trees

```
cart = DecisionTreeClassifier()
num_trees = 100

model = BaggingClassifier(base_estimator = cart, n_estimators = num_trees,
random_state=seed)
```

```
result = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

```
(-4.498) +/- (4.692)
```

Random Forest

```
num_features = 10

model = RandomForestClassifier(n_estimators=num_trees,
max_features=num_features)

results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

Extra Trees

```
model = ExtraTreesClassifier(n_estimators=num_trees,
max_features=num_features)
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
result = np.array(result)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

```
(-2.923) +/- (0.414)
```

التعزيز

تسعى خوارزميات التعزيز Boosting algorithms إلى تحسين قوة التنبؤ من خلال تدريب سلسلة من النماذج الضعيفة weak models، حيث يعوض كل منها نقاط الضعف في النماذج السابقة. لفهم التعزيز، من الأهمية بمكان إدراك أن التعزيز عبارة عن خوارزمية عامة وليست نموذجًا محددًا. يتطلب التعزيز منك تحديد نموذج ضعيف (على سبيل المثال الانحدار regression، وأشجار القرار الضحلة shallow decision trees، وما إلى ذلك) ثم تحسينه.

AdaBoost

```
model = AdaBoostClassifier(n_estimators=100, random_state=seed)
results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

```
(-0.691) +/- (0.000)
```

Stochastic Gradient Boosting

```
model = GradientBoostingClassifier(n_estimators=100, random_state=seed)

results = cross_val_score(model, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

```
(-0.608) +/- (0.003)
```

ضبط المعلمات الفائقة

لقد بقي لدينا كل النماذج التي حصلت على نتائج أفضل. ولكن كان من الممكن أن تحصل على أداء أفضل إذا حددنا البُنية المثلى للنماذج. وهذا ما سنفعله هنا: الاختيار من قائمة محددة من المعلمات الفائقة hyperparameters لكل نموذج المعلمات التي تعمل بشكل أفضل لبياناتنا.

تُعرف عملية الاختيار هذه للمعلمات الفائقة باسم ضبط المعلمات الفائقة Hyperparameter Tuning، وستكون GridSearchCV() أفضل صديق لنا.

Logistic Regression

```
lr_grid = GridSearchCV(
    estimator = LogisticRegression(random_state=seed),
    param_grid = {
        'penalty': ['l1', 'l2'],
        'C': [0.001, 0.01, 1, 10, 100, 1000]
    },
    cv = kfold,
    scoring=scoring,
    n_jobs=processors)

lr_grid.fit(X, Y)

print(lr_grid.best_score_)
print(lr_grid.best_params_)
```

```
-0.611906360356922
{'C': 1000, 'penalty': 'l2'}
```

Linear Discriminant Analysis

```
lda_grid = GridSearchCV(
    estimator = LinearDiscriminantAnalysis(),
    param_grid = {
        'solver': ['lsqr'],
        'shrinkage':[0, 0.25, 0.5, 0.75, 1],
        'n_components':[None, 2, 5, 10]
    },
    cv=kfold,
    scoring=scoring,
    n_jobs=processors)

lda_grid.fit(X, Y)

print(lr_grid.best_score_)
print(lr_grid.best_params_)
```

```
-0.611906360356922
{'C': 1000, 'penalty': 'l2'}
```

K-NN

```
knn_grid = GridSearchCV(
    estimator = Pipeline([
        ('min_max_scaler', MinMaxScaler()),
        ('knn', KNeighborsClassifier())
    ]),
    param_grid = {
```

```
        'knn__n_neighbors': [25],
        'knn__algorithm': ['ball_tree'],
        'knn__leaf_size': [2, 3, 4],
        'knn__p': [1]
    },
    cv = kfold,
    scoring = scoring,
    n_jobs=processors
    )

knn_grid.fit(X, Y)

print(knn_grid.best_score_)
print(knn_grid.best_params_)
```

```
-0.6517169770479284
{'knn__algorithm': 'ball_tree', 'knn__leaf_size': 2, 'knn__n_neighbors':
25, 'knn__p': 1}
```

Random Forest

```
rf_grid = GridSearchCV(
    estimator = RandomForestClassifier(warm_start=True, random_state=seed),
    param_grid = {
        'n_estimators': [100, 200],
        'criterion': ['gini', 'entropy'],
        'max_features': [18, 20],
        'max_depth': [8, 10],
        'bootstrap': [True]
    },
    cv = kfold,
    scoring = scoring,
    n_jobs = processors)

rf_grid.fit(X, Y)

print(rf_grid.best_score_)
print(rf_grid.best_params_)
```

```
-0.6073254159213145
{'bootstrap': True, 'criterion': 'entropy', 'max_depth': 10,
'max_features': 20, 'n_estimators': 200}
```

AdaBoost

```
ada_grid = GridSearchCV(
    estimator = AdaBoostClassifier(random_state=seed),
    param_grid = {
        'algorithm': ['SAMME', 'SAMME.R'],
        'n_estimators': [10, 25, 50],
        'learning_rate': [1e-3, 1e-2, 1e-1]
    },
    cv = kfold,
    scoring = scoring,
    n_jobs = processors)

ada_grid.fit(X, Y)

print(ada_grid.best_score_)
print(ada_grid.best_params_)
```

```
0.6378792171309785
{'algorithm': 'SAMME', 'learning_rate': 0.1, 'n_estimators': 50}
```

Gradient Boosting

```
gbm_grid = GridSearchCV(
    estimator = GradientBoostingClassifier(warm_start=True,
random_state=seed),
    param_grid = {
        'n estimators': [100, 200],
        'max_depth': [2, 3, 4],
        'max_features': [10, 15, 20],
        'learning_rate': [1e-1, 1]
    },
    cv = kfold,
    scoring = scoring,
    n_jobs = processors)

gbm_grid.fit(X, Y)

print(gbm_grid.best_score_)
print(gbm_grid.best_params_)
```

```
-0.6072089276104001
{'learning_rate': 0.1, 'max_depth': 4, 'max_features': 10, 'n_estimators':
200}
```

النموذج النهائي: تجميع التصويت

نحن الآنفي خطوتنا الأخيرةفي تطوير النموذج. نختار أفضل أربعة نماذج لدينا بناءً على تسجيل خطأ السجل، مع أفضل معلماتها الفائقة الممكنة، ونجمعهافي تجميع ensemble تسمى مصنف التصويت Voting classifier.

يعد التصويت أحد أبسط الطرق لدمج التنبؤات من خوارزميات التعلم الآلي المتعددة. لا يعد مصنف التصويت مصنفًا فعليًا ولكنه غلاف لمجموعة من الخوارزميات المختلفة التي يتم تدريبها وتقييمها بالتوازي، من أجل استغلال الخصائص المختلفة لكل منها.

في التصويت الناعم soft voting(النمط الذي اخترناه)، يتم تلخيص متجه الاحتمال لكل فئة متوقعة (لجميع المصنفات) ومتوسطها. الفئة الفائزة هي الفئة المقابلة لأعلى قيمة. كما نحدد أوزانًا مختلفة وفقًا لنتائج النماذج: على سبيل المثال، تعزيز التدرج gradient boosting والغابات العشوائية random forest، النموذجان اللذان حققا خطأ سجل أفضل، لهما وزن 3.

باستخدام طريقة المتابعة هذه، لدينا نماذج أكثر قوة.

```
estimators = []
estimators.append(('lr', LogisticRegression(penalty='l2', C=1)))
```

```
estimators.append(('gbm', GradientBoostingClassifier(n_estimators=200,
max_depth=3, learning_rate=0.1, max_features=15, warm_start=True,
random_state=seed)))
estimators.append(('rf', RandomForestClassifier(bootstrap=True,
max_depth=8, n_estimators=200, max_features=20, criterion='entropy',
random_state=seed)))
estimators.append(('ada', AdaBoostClassifier(algorithm='SAMME.R',
learning_rate=1e-2, n_estimators=10, random_state=seed)))


# Create the ensemble model
ensemble = VotingClassifier(estimators, voting='soft', weights=[2,3,3,1])

results = cross_val_score(ensemble, X, Y, cv=kfold, scoring=scoring,
n_jobs=processors)
print("({0:.3f}) +/- ({1:.3f})".format(np.mean(results), np.std(results)))
```

```
(-0.609) +/- (0.003)
```

التوقعات النهائية

```
model = ensemble
model.fit(X, Y)
preds = model.predict_proba(data_submit)
preds
```

```
array([[0.64890606, 0.35109394],
       [0.67166631, 0.32833369],
       [0.33057146, 0.66942854],
       ...,
       [0.26230402, 0.73769598],
       [0.23040411, 0.76959589],
       [0.60236878, 0.39763122]])
```

```
submission = pd.DataFrame()
submission["shot_id"] = data_submit.index
submission["shot_made_flag"]= preds[:,0]

submission.to_csv("sub.csv", index=False)
```

المصدر:

https://www.kaggle.com/code/marinovik/kobe-bryant-shot-selection-analysis

16) تحليل الدوري الهندي الممتاز للكريكيت باستخدام بايثون IPL 2024 RCB vs DC Analysis using Python

إذا كنت مهتمًا بمعرفة المزيد عن استخدام علم البيانات Data Scienceفي تحليلات لعبة الكريكيت cricket analytics، فهذه المقالة لك. كانت المباراة بين RCB وDelhi Capitals واحدة من مبارياتي المفضلةفي الدوري الهندي الممتاز Indian Premier League (IPL) هذا العام. لذا،في هذه المقالة، سأطلعك على تحليلي لمباراة IPL الكاملة RCB vs DC 2024 باستخدام بايثون.

مجموعة البيانات

تحتوي مجموعة البيانات التي جمعناها لتحليل مباراة IPL هذه على الأعمدة التالية:

- team: يشير إلى فريق الضرب.
- over: عدد الاشواطفي المباراة.
- batter: رجل الضرب الذي يواجه التسليم.
- bowler: لاعب البولينج الذي يسلم الكرة.
- non_striker: رجل الضربفي نهاية غير المهاجم.
- runs_batter: الأشواط التي سجلها الضارب من التسليم.
- runs_extras: الأشواط الإضافية (مثل الكرات العريضة، وعدم وجودكرات) التي تم التنازل عنها أثناء التسليم.
- runs_total: إجمالي الأشواط المسجلة من التسليم.
- player_out: اسم اللاعب الذي خرج من التسليم (إن وجد).
- wicket_kind: نوع الوكت (إن وجد).
- fielders: أسماء لاعبي الميدان المشاركينفي الوكت (إن وجد).

يمكنك تنزيل مجموعة البيانات من هنا.

تحليل IPL 2024 RCB Vs DC باستخدام بايثون

الآن، لنبدأ هذه المهمة عن طريق استيراد مكتبات بايثون الضرورية ومجموعة البيانات:

```
import pandas as pd
deliveries_df = pd.read_csv("innings_deliveries.csv")

print(deliveries_df.head())
```

```
                        team  over        batter    bowler   non_striker  \
0  Royal Challengers Bengaluru     0      V Kohli  I Sharma  F du Plessis
1  Royal Challengers Bengaluru     0      V Kohli  I Sharma  F du Plessis
2  Royal Challengers Bengaluru     0  F du Plessis  I Sharma       V Kohli
3  Royal Challengers Bengaluru     0      V Kohli  I Sharma  F du Plessis
4  Royal Challengers Bengaluru     0      V Kohli  I Sharma  F du Plessis

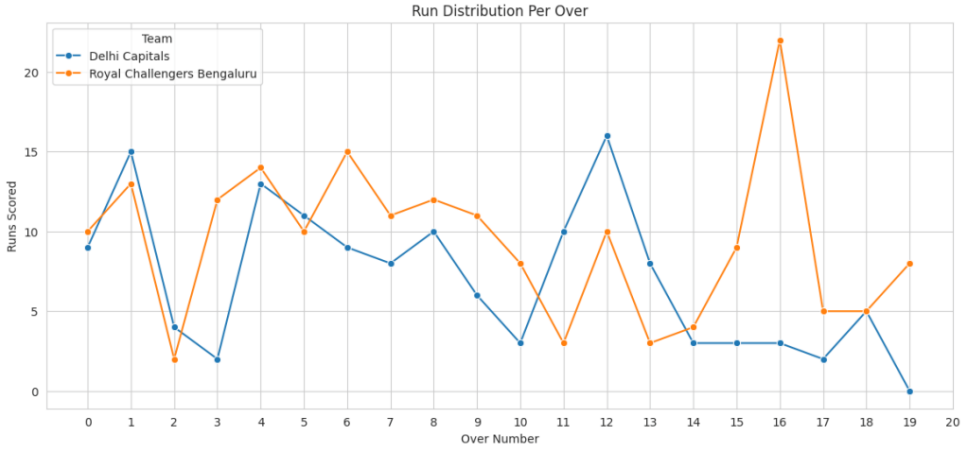
   runs_batter  runs_extras  runs_total player_out wicket_kind fielders
0            0            0           0        NaN         NaN       []
1            1            0           1        NaN         NaN       []
2            1            0           1        NaN         NaN       []
3            0            0           0        NaN         NaN       []
4            2            0           2        NaN         NaN       []
```

تحتوي مجموعة البيانات على عدة قيم فارغة null values ولكننا لسنا بحاجة إلى حذف أي صف يحتوي على قيم فارغةفي هذه الحالة أو ملء أي قيمة فارغة لأن ذلك سيؤثر على البيانات. لذا، دون إضاعة أي وقت، فلنبدأ بتصور توزيع الأشواط لكل جولة لكلا الفريقين لتوضيح اتجاهات التسجيل طوال الجولات:

```
import matplotlib.pyplot as plt
import seaborn as sns
sns.set_style("whitegrid")

# data preparation for run distribution per over
run_distribution = deliveries_df.groupby(['team',
'over']).agg({'runs_total': 'sum'}).reset_index()

# plotting run distribution per over for both teams
plt.figure(figsize=(14, 6))
sns.lineplot(data=run_distribution, x='over', y='runs_total', hue='team',
marker='o')
plt.title('Run Distribution Per Over')
plt.xlabel('Over Number')
plt.ylabel('Runs Scored')
plt.xticks(range(0, 21))  # over numbers from 0 to 20
plt.legend(title='Team')
plt.show()
```



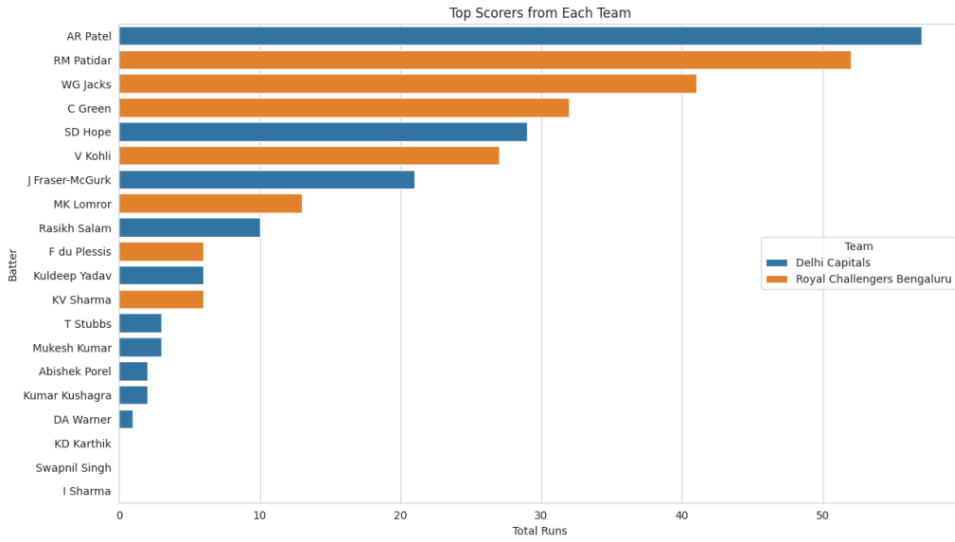
يوضح الرسم البياني أعلاه توزيع النقاط لكل جولة لكلا الفريقين. وفيما يلي بعض الأفكار:

- يُظهِر معدل التسجيل لكل فريق تقلبات طوال جولاتهم، مع ارتفاعات تشير إلى جولات بها نقاط عالية، ربما بسبب الحدود أو الضربات القوية.
- يبدو أن فريق Royal Challengers Bangalore (RCB) لديه شوطين بهما نقاط أعلى بشكل ملحوظ، مما يشير إلى الضرب العدواني.

بعد ذلك، سأحلل أفضل الهدافين من كل فريق لتسليط الضوء على الأداء الفردي. دعنا ننشئ مخططًا بيانيًا لتوضيح المساهمين الرئيسيين من حيث النقاط:

```
 #calculating top scorers for each team
top_scorers = deliveries_df.groupby(['team', 'batter']).agg({'runs_batter':
'sum'}).reset_index().sort_values(by='runs_batter', ascending=False)

plt.figure(figsize=(14, 8))
sns.barplot(data=top_scorers, x='runs_batter', y='batter', hue='team',
dodge=False)
plt.title('Top Scorers from Each Team')
plt.xlabel('Total Runs')
plt.ylabel('Batter')
plt.legend(title='Team', loc='center right')
plt.show()
```



تتضمن الملاحظات الرئيسية من الرسم البياني ما يلي:

- يعد AR Patel من فريق Delhi Capitals هو هداف المباراة، حيث تفوق بشكل كبير على الآخرين بأكثر من 50 نقطة.

- يعد RM Patidar هو هداف فريق Royal Challengers Bangalore، حيث يقترب من 50 نقطة.
- يعرض الرسم البياني مساهمة متنوعة من كلا الفريقين، حيث ساهم العديد من اللاعبين من كلا الجانبين بنتائج ملحوظة.

الآن، دعنا ننتقل إلى تحليل الرمي bowling analysis. سننظر إلى اللاعبين الذين أخذوا أكبر عدد من الوكتات ومعدلات الاقتصاد economy rates الخاصة بهم. سيتضمن ذلك حساب عدد الوكتات التي أخذها كل لاعب وعدد النقاط التي استقبلهافي كل جولة. سنعرض هذه البياناتفي رسم بياني شريطي وخطي مشترك للحصول على رؤية شاملة لأداء الرمي:

```
 #preparing data for bowling analysis
deliveries_df['wickets_taken'] =
deliveries_df['wicket_kind'].notna().astype(int)
bowling_stats = deliveries_df.groupby(['team',
'bowler']).agg({'runs_total': 'sum', 'wickets_taken': 'sum', 'over':
'nunique'}).reset_index()

 #calculating economy rate (total runs conceded / number of overs bowled)
bowling_stats['economy_rate'] = bowling_stats['runs_total'] /
bowling_stats['over']

 #sorting the data for better visualization
bowling_stats_sorted = bowling_stats.sort_values(by='wickets_taken',
ascending=False)

 #prepare the DataFrame for plotting
bowling_stats_sorted['wickets_taken'] =
deliveries_df['wicket_kind'].notna().astype(int)
bowling_stats = deliveries_df.groupby(['team',
'bowler']).agg({'runs_total': 'sum', 'wickets_taken': 'sum', 'over':
'nunique'}).reset_index()
bowling_stats['economy_rate'] = bowling_stats['runs_total'] /
bowling_stats['over']
bowling_stats_sorted = bowling_stats.sort_values(by='wickets_taken',
ascending=False)

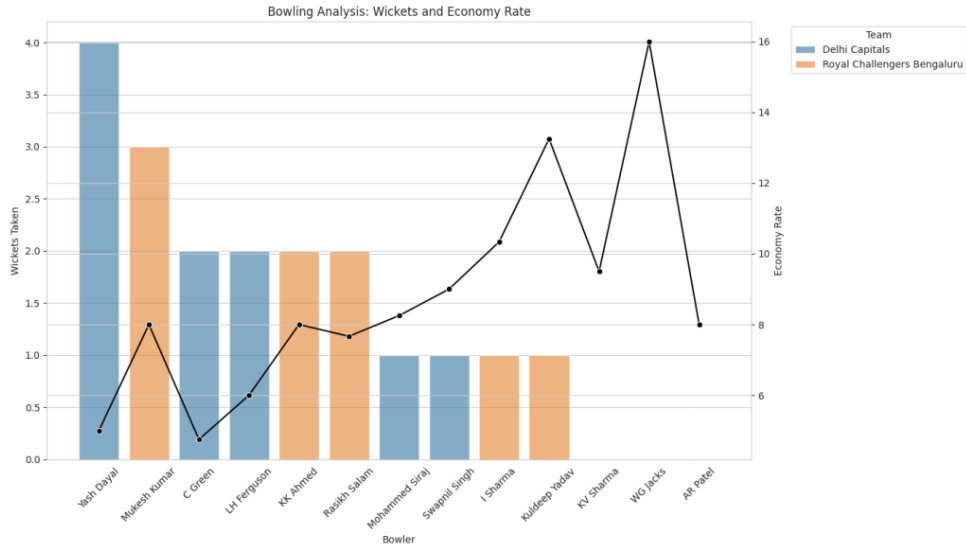
 #create the plot
fig, ax1 = plt.subplots(figsize=(14, 8))

 #Bar plot for wickets
sns.barplot(data=bowling_stats_sorted, x='bowler', y='wickets_taken',
hue='team', ax=ax1, alpha=0.6)
ax1.set_ylabel('Wickets Taken')
ax1.set_xlabel('Bowler')
ax1.set_title('Bowling Analysis: Wickets and Economy Rate')
ax1.legend(title='Team', bbox_to_anchor=(1.05, 1), loc='upper left')

for item in ax1.get_xticklabels:()
    item.set_rotation(45)

ax2 = ax1.twinx()
sns.lineplot(data=bowling_stats_sorted, x='bowler', y='economy_rate',
marker='o', sort=False, ax=ax2, color='black')
ax2.set_ylabel('Economy Rate')
```

```
plt.tight_layout()
plt.show()
```



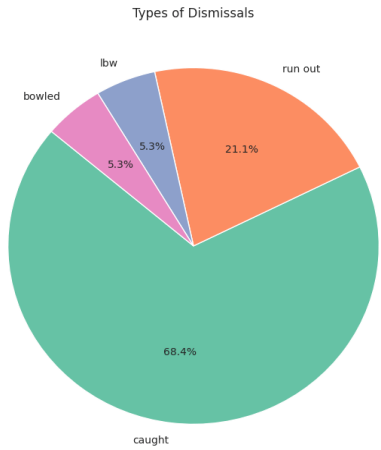
يوفر الرسم البياني المشترك للخطوط والأشرطة نظرة عامة شاملة على أداء كل فريقفي الرمي:

- **الوكتات المأخوذة Wickets Taken:** تشير الأشرطة إلى عدد الوكتات التي أخذها كل لاعب أثناء المباراة. يعكس ارتفاع الأشرطة مدى نجاح اللاعبين من حيث أخذ الوكتات. ساهم لاعبو الرمي من كلا الفريقينفي أخذ الوكتات، مع بعض الأداء الملحوظ الذي يبرز بسبب الأشرطة الأعلى.
- **معدل الاقتصاد Economy Rate:** يوضح الرسم البياني الخطي الموجود فوق الرسم البياني الشريطي معدل الاقتصاد (عدد النقاط التي استقبلها كل شوط) لكل لاعب. معدل الاقتصاد أمر بالغ الأهمية لأنه يشير إلى مدى اقتصاد اللاعبفي لعب الرمي من حيث النقاط التي تلقاها.

الآن، دعنا نحلل أنواع الوكتات التي حدثت أثناء المباراة لفهم كيفية أخذ معظم الوكتات (على سبيل المثال، الإمساك caught، الرمي bowled، الخروج run out). يمكن أن يوفر هذا رؤى حول طبيعة الملعب وظروف اللعب. سنصور ذلك باستخدام مخطط دائري pie chart:

```
# counting dismissal types
dismissal_types = deliveries_df['wicket_kind'].dropna().value_counts()

plt.figure(figsize=(8, 8))
plt.pie(dismissal_types, labels=dismissal_types.index, autopct='%1.1f%%',
startangle=140, colors=sns.color_palette("Set2"))
plt.title('Types of Dismissals')
plt.show()
```



الآن، دعنا نجري تحليل الشراكات Partnerships Analysis من خلال حساب وتصور الشراكات الأكثر إنتاجيةفي الضربفي المباراة. سننظرفي عدد النقاط المسجلة لكل شراكة ومدة استمرار كل شراكة من حيث الكرات التي واجهتها:

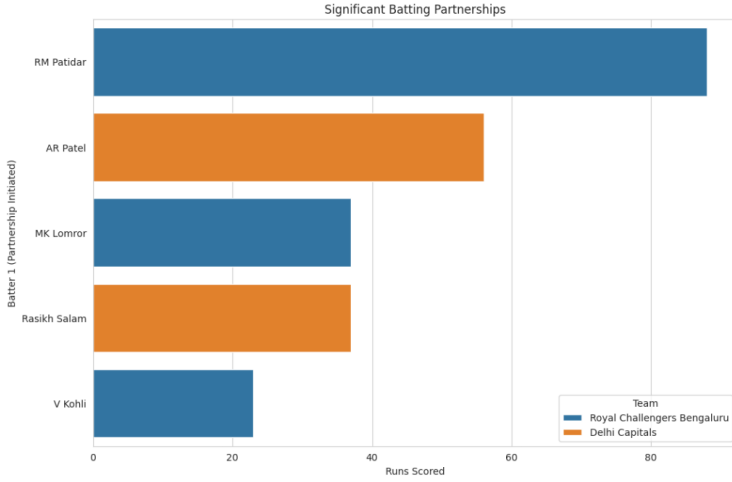
```
# function to calculate partnerships
def calculate_partnerships(df):
    partnerships = []
    current_partnership = {}
    for i, row in df.iterrows():
        if i == 0 or (row['batter'] not in current_partnership.values()):
            if current_partnership:
                partnerships.append(current_partnership)
            current_partnership = {
                'team': row['team'],
                'batter1': row['batter'],
                'batter2': row['non_striker'],
                'runs': 0,
                'balls': 0
            }
        current_partnership['runs'] += row['runs_total']
        current_partnership['balls'] += 1
        if 'player_out' in row and pd.notna(row['player_out']):
            if row['player_out'] == current_partnership['batter1'] or
row['player_out'] == current_partnership['batter2']:
                partnerships.append(current_partnership)
                current_partnership = {}
    # append the last partnership if not ended by a wicket
    if current_partnership:
        partnerships.append(current_partnership)
    return partnerships

# calculate partnerships
partnerships_data = calculate_partnerships(deliveries_df)
partnerships_df = pd.DataFrame(partnerships_data)
```

```
# filter out significant partnerships (e.g., partnerships with more than 20
runs)
significant_partnerships = partnerships_df[partnerships_df['runs'] > 20]

# sort by highest runs
significant_partnerships = significant_partnerships.sort_values(by='runs',
ascending=False)

plt.figure(figsize=(12, 8))
sns.barplot(data=significant_partnerships, x='runs', y='batter1',
hue='team', dodge=False)
plt.title('Significant Batting Partnerships')
plt.xlabel('Runs Scored')
plt.ylabel('Batter 1 (Partnership Initiated)')
plt.legend(title='Team')
plt.show()
```



يعرض الرسم البياني الشريطي الشراكات المهمةفي الضرب خلال المباراة، مع تسليط الضوء على الشراكات التي سجلت أكثر من 20 نقطة. وفيما يلي كيفية مساهمة هذه الرؤىفي تحليلنا:

- يحدد الرسم البياني الشراكات الرئيسية التي كان لها على الأرجح تأثير كبير على نتيجة المباراة، مما يوضح فعالية أزواج الضرب batting pairs.
- يوفر نظرة ثاقبة حول اللاعبين الذين شاركوافي المواقف المحورية pivotal stands، والتي يمكن أن تساعدفي تقييم شكل اللاعب واستراتيجية الفريق.

بعد ذلك، دعنا نجري تحليلًا للمرحلة لفحص أداء الفرق خلال مراحل مختلفة من أدوارها؛ اللعب بقوة (أول 6 أشواط)، والأشواط المتوسطة (7–15)، والأشواط القاتلة (16–20). سننظرفي معدلات التسجيل وخسارة الوكت خلال هذه المراحل. يمكن أن يوفر هذا نظرة ثاقبة حول النهج التكتيكي للفريق وتنفيذهفي ظل ظروف مختلفة:

```
# function to classify the phase of the game based on the over number
```

```
def classify_phase(over):
    if over < 6:
        return 'Powerplay'
    elif over < 16:
        return 'Middle'
    else:
        return 'Death'

# adding phase information to the dataframe
deliveries_df['phase'] = deliveries_df['over'].apply(classify_phase)

# grouping data by phase and team to calculate runs and wickets
phase_analysis = deliveries_df.groupby(['team',
'phase']).agg({'runs_total': 'sum', 'wickets_taken': 'sum', 'over':
'count'}).rename(columns={'over': 'balls'}).reset_index()

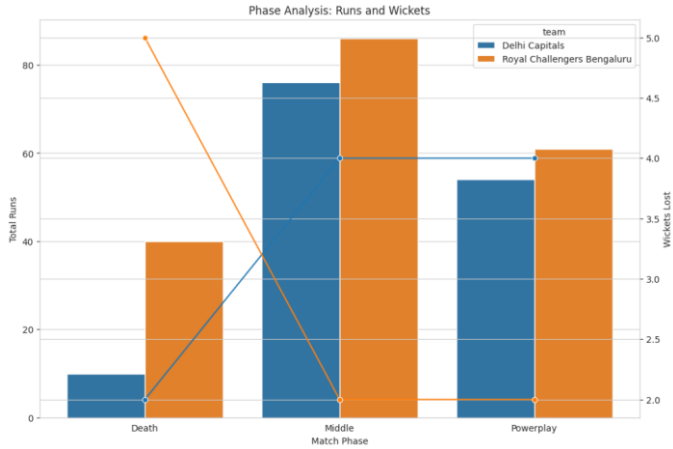
# calculating the run rate
phase_analysis['run_rate'] = (phase_analysis['runs_total'] /
phase_analysis['balls']) * 6

# plotting the phase analysis
fig, ax1 = plt.subplots(figsize=(12, 8))

# bar plot for runs scored in each phase
sns.barplot(data=phase_analysis, x='phase', y='runs_total', hue='team',
ax=ax1)
ax1.set_title('Phase Analysis: Runs and Wickets')
ax1.set_ylabel('Total Runs')
ax1.set_xlabel('Match Phase')

# line plot for wickets lost
ax2 = ax1.twinx()
sns.lineplot(data=phase_analysis, x='phase', y='wickets_taken', hue='team',
marker='o', ax=ax2, legend=False)
ax2.set_ylabel('Wickets Lost')

plt.show()
```



يقدم الرسم البياني أعلاه تفصيلاً واضحًا للمباراة إلى مراحل مختلفة؛ Powerplay وMiddle وDeath، ويوضح أداء كل فريق خلال هذه الأجزاء:

- **Powerplay**: يتمتع كلا الفريقين بإجمالي منخفض نسبيًا من الأشواط، حيث يخسر RCB عددًا أكبر من الوكتات مقارنة بـ DCفي هذه المرحلة، كما هو موضح من ارتفاع الخط البرتقالي.
- **Middle**: تُظهر هذه المرحلة أعلى معدل تسجيل للأشواط لكلا الفريقين، حيث سجل DC عددًا أكبر قليلاً من RCB. تظل الوكتات المفقودة خاضعة للسيطرة، مما يشير إلى أدوار مستقرة من كلا الفريقين.
- **Death**: يشهد RCB انخفاضًا حادًافي الأشواط مقارنة بمرحلة Middle، بينما يحافظ DC على معدل أشواط مرتفع. زادت الوكتات التي خسرها RCB بشكل كبيرفي هذه المرحلة، والتي تميزت بارتفاع الخط البرتقالي بالقرب من 4.5، مما يشير إلى انهيار محتمل أو ضرب عدواني لم يؤتي ثماره.

الآن، دعنا نحسب معدلات الضرب strike rates لجميع الضاربينفي هذه المباراة ثم نحلل البيانات لمعرفة اللاعبين الأكثر فعالية من حيث التسجيل السريع. بعد حساب معدلات الضربات، يمكننا النظر في الارتباطات مع متغيرات أخرى مثل عدد النقاط المسجلة أو مرحلة اللعب التي تم خلالها تسجيل النقاط. يمكن أن يمنحنا هذا رؤى حول اللاعبين الذين يسرعونفي التسجيلفي الأوقات الحاسمة أو ضد لاعبين محددين. أولاً، سأحسب معدل الضربات لكل ضارب:

```
# calculate runs and balls faced for each batter
batter_stats = deliveries_df.groupby('batter').agg({'runs_batter': 'sum',
'over': 'count'}).rename(columns={'over': 'balls faced'}).reset index()

# calculate strike rate for each batter (runs per 100 balls)
batter_stats['strike_rate'] = (batter_stats['runs_batter'] /
batter_stats['balls_faced']) * 100

# sorting batters by their strike rate
batter_stats_sorted = batter_stats.sort_values(by='strike_rate',
ascending=False)

# displaying calculated strike rates along with runs scored and balls faced
batter_stats_sorted.head(10)
```

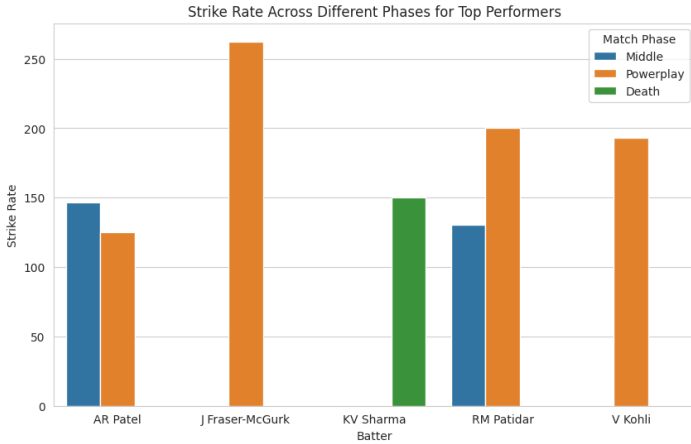
	batter	runs_batter	balls_faced	strike_rate
6	J Fraser-McGurk	21	8	262.500000
18	V Kohli	27	14	192.857143
13	RM Patidar	52	34	152.941176
8	KV Sharma	6	4	150.000000
0	AR Patel	57	40	142.500000
19	WG Jacks	41	30	136.666667
2	C Green	32	24	133.333333
11	MK Lomror	13	10	130.000000
15	SD Hope	29	24	120.833333
4	F du Plessis	6	7	85.714286

فيما يلي أفضل اللاعبين من حيث معدل الضرباتفي المباراة:

- حقق جيه فريزر_ماكجورك أعلى معدل ضربات عند 262.50، حيث سجل 21 نقطة من 8 كرات فقط.
- كما سجل فيرات كوهلي بكفاءة، بمعدل ضربات بلغ 192.86، حيث سجل 27 نقطة من 14 كرة.
- ساهم راجات باتيدار بشكل كبير بمعدل ضربات بلغ 152.94، حيث جمع 52 نقطة من 34 كرة.

الآن، دعنا نتعمق أكثر من خلال النظرفي كيفية تغير معدل الضربات مع مرحلة اللعبة لهؤلاء اللاعبين الأفضل أداءً. يمكن أن يعطي هذا نظرة ثاقبة حول التسجيل الاستراتيجي وديناميكيات اللعبة خلال مراحل الأدوار المختلفة:

```
# merging phase information with batter stats
batter_phase_stats = deliveries_df.groupby(['batter',
'phase']).agg({'runs_batter': 'sum', 'over':
'count'}).rename(columns={'over': 'balls_faced'}).reset_index()
# calculate strike rate for each batter-phase combination
batter_phase_stats['strike_rate'] = (batter_phase_stats['runs_batter'] /
batter_phase_stats['balls_faced']) * 100
# filtering for top performers based on overall strike rate
top_performers = batter_stats_sorted.head(5)['batter']
batter_phase_stats_top =
batter_phase_stats[batter_phase_stats['batter'].isin(top_performers)]
# plotting strike rate across different phases for top performers
plt.figure(figsize=(10, 6))
sns.barplot(data=batter_phase_stats_top, x='batter', y='strike_rate',
hue='phase')
plt.title('Strike Rate Across Different Phases for Top Performers')
plt.xlabel('Batter')
plt.ylabel('Strike Rate')
plt.legend(title='Match Phase')
plt.show()
```



يوضح الرسم البياني الشريطي كيف تختلف معدلات ضربات أفضل اللاعبين عبر مراحل مختلفة من المباراة:

- يتميز جيه فريزر–ماكجورك بمعدل ضربات مرتفع بشكل خاصفي المرحلة الوسطى Middle phase، وهو أعلى بكثير من أي مرحلة أخرى أو لاعب آخر، مما يشير إلى أداء ضارب عدواني وفعال للغاية خلال هذا الجزء من الأدوار.
- يتمتع كل منفي كوهلي و آر إم باتيدار بمعدلات ضربات عاليةفي مرحلة الموت Death phase، مما يشير إلى قدرتهما على تسريع التسجيل نحو نهاية الأدوار، وهو أمر بالغ الأهمية لتحديد الأهداف أو مطاردتها.
- يُظهر آر باتيل ثباتًافي مرحلتي اللعب بقوة Powerplay والوسط Middle بمعدل ضربات منخفض قليلاً ولكنه لا يزال تنافسيًا، مما يشير إلى دوره كلاعب افتتاحي ثابت أو ضاربفي منتصف الترتيب.
- يُظهر كيهفي شارما معدل ضربات أقلفي المرحلة الوسطى مقارنة بالآخرين، مما يشير إلى نهج أكثر تحفظًا خلال هذه المرحلة أو صعوبةفي التسارع.

تحديد نقاط التحول في المباراة

لتحديد نقطة التحول turning point حيث قد يخسر فريق دلهي كابيتالز Delhi Capitals (DC) المباراة ويكسب فريق رويال تشالنجرز بنغالور Royal Challengers Bangalore (RCB) اليد العليا، يمكننا تحليل مقارنة معدل الجري التراكمي طوال الأدوار وإلقاء نظرة على أحداث سقوط الويكيت. على وجه التحديد، يمكننا:

- **مقارنة معدلات الجري التراكمية Compare Cumulative Run Rates:** رسم معدلات الجري التراكمية لكلا الفريقين طوال أدوارهما لمعرفة أين بدأ RCBفي التفوق على DC بشكل كبير.
- **تحليل الوكت Wicket Analysis:** فحص توقيتات وتأثيرات سقوط الويكيت على معدل التسجيل والزخم لـ DC.
- **الأشواط ذات التأثير العالي High-Impact Overs:** حدد أي أشواط حيث أخذ RCB عدة وكتات أو كان معدل تسجيل DC منخفضًا بشكل كبير، مما قد يشير إلى فقدان الزخم.

دعنا نبدأ برسم معدل الجري التراكمي cumulative run rate لكلا الفريقين وتغطية أحداث الوكت لتحديد اللحظات الحرجةفي المباراة:

```
# calculate cumulative runs and wickets for each ball for both teams
deliveries_df['cumulative_runs'] =
deliveries_df.groupby('team')['runs_total'].cumsum()
```

```
deliveries_df['cumulative_wickets'] =
deliveries_df.groupby('team')['wickets_taken'].cumsum()

# separate data for both teams
rcb_deliveries = deliveries_df[deliveries_df['team'] == 'Royal Challengers
Bengaluru']
dc_deliveries = deliveries_df[deliveries_df['team'] == 'Delhi Capitals']

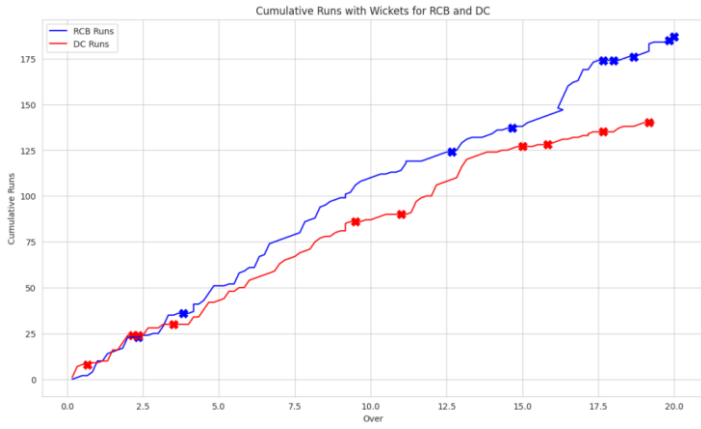
# calculating overs for cumulative analysis
rcb_deliveries['over_ball'] = rcb_deliveries['over'] +
(rcb_deliveries.groupby('over').cumcount() + 1) / 6
dc_deliveries['over_ball'] = dc_deliveries['over'] +
(dc_deliveries.groupby('over').cumcount() + 1) / 6

# plotting cumulative run rates and wickets
fig, ax = plt.subplots(figsize=(14, 8))

# plot for RCB
ax.plot(rcb_deliveries['over_ball'], rcb_deliveries['cumulative_runs'],
color='blue', label='RCB Runs')
ax.scatter(rcb_deliveries[rcb_deliveries['wickets_taken'] ==
1]['over_ball'], rcb_deliveries[rcb_deliveries['wickets_taken'] ==
1]['cumulative_runs'], color='blue', marker='X', s=100)

# plot for DC
ax.plot(dc_deliveries['over_ball'], dc_deliveries['cumulative_runs'],
color='red', label='DC Runs')
ax.scatter(dc_deliveries[dc_deliveries['wickets_taken'] == 1]['over_ball'],
dc_deliveries[dc_deliveries['wickets_taken'] == 1]['cumulative_runs'],
color='red', marker='X', s=100)

ax.set_title('Cumulative Runs with Wickets for RCB and DC')
ax.set_xlabel('Over')
ax.set_ylabel('Cumulative Runs')
ax.legend()
plt.show()
```



يوضح الرسم البياني إجمالي عدد النقاط التي سجلها كل فريق طوال جولاته، مع وجود علامات تشير إلى الوكتات:

- **تحولات الزخم Momentum Shifts:** تعد النقاط التي خسر فيها الفريق الوكتات بالغة الأهمية. وعلى الرغم من خسارة الوكتات، لا يظهر خط ركض RCB أي انخفاضات جذرية، مما يشير إلى التعافي الفعال من قبل الضاربين اللاحقين.
- **تحليل الأداء Performance Analysis:** قد تشير قدرة RCB على الحفاظ على معدل الركض مرتفعًا على الرغم من خسارة الوكتات إلى قوة ضاربة أعمق أو استراتيجيات ناجحة لتسريع الجولات. على النقيض من ذلك، بينما يزيد DC أيضًا من نقاطه، فإنه يفعل ذلك بمعدل أقل حدة، مما يشير ربما إلى عدد أقل من الجولات الكبيرة.

الآن، دعنا نحسب معدل النقاط لكل جولة لكلا الفريقين ونرى كيف تغيرت معدلات النقاط طوال الجولات، مع التركيز بشكل خاص على الجولات التي سقطت فيها الوكتات:

```
 #calculate runs and wickets per over for both teams
per_over_stats = deliveries_df.groupby(['team', 'over']).agg({'runs_total':
'sum', 'wickets_taken': 'sum'}).reset_index()

 #calculate run rate for each over
per_over_stats['run_rate'] = (per_over_stats['runs_total'] / 6)    # Runs
per over to runs per ball (standard rate)

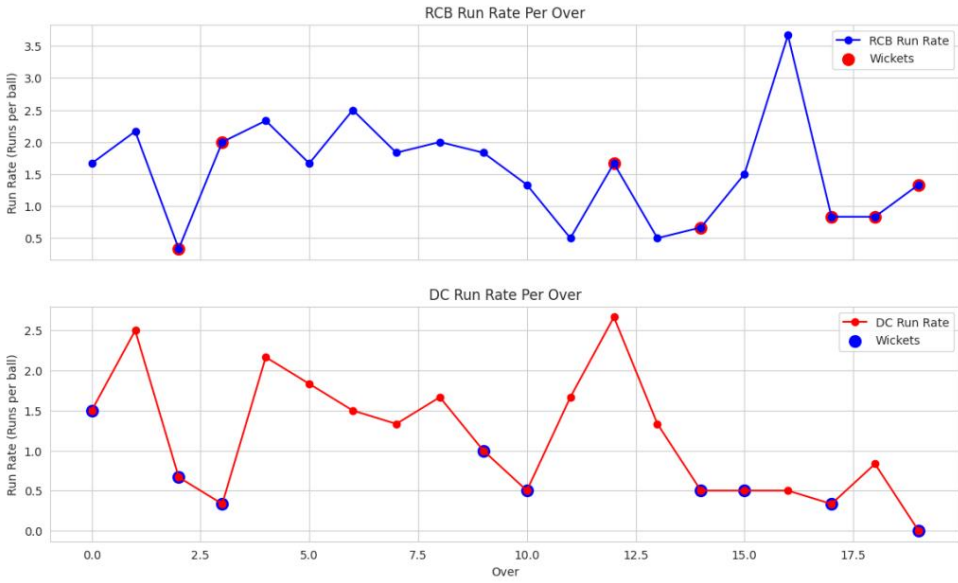
 #separate data for RCB and DC for plotting
rcb_per_over_stats = per_over_stats[per_over_stats['team'] == 'Royal
Challengers Bengaluru']
dc_per_over_stats = per_over_stats[per_over_stats['team'] == 'Delhi
Capitals']

 #plotting run rates and marking wickets for each team
fig, (ax1, ax2) = plt.subplots(2, 1, figsize=(14, 8), sharex=True)

 #RCB
ax1.plot(rcb_per_over_stats['over'], rcb_per_over_stats['run_rate'],
marker='o', color='blue', label='RCB Run Rate')
ax1.scatter(rcb_per_over_stats[rcb_per_over_stats['wickets_taken'] >
0]['over'], rcb_per_over_stats[rcb_per_over_stats['wickets_taken'] >
0]['run_rate'], color='red', s=100, label='Wickets')
ax1.set_title('RCB Run Rate Per Over')
ax1.set_ylabel('Run Rate (Runs per ball)')
ax1.legend()

 #DC
ax2.plot(dc_per_over_stats['over'], dc_per_over_stats['run_rate'],
marker='o', color='red', label='DC Run Rate')
ax2.scatter(dc_per_over_stats[dc_per_over_stats['wickets_taken'] >
0]['over'], dc_per_over_stats[dc_per_over_stats['wickets_taken'] >
0]['run_rate'], color='blue', s=100, label='Wickets')
ax2.set_title('DC Run Rate Per Over')
ax2.set_xlabel('Over')
ax2.set_ylabel('Run Rate (Runs per ball)')
ax2.legend()

plt.show()
```



تقدم معدلات النقاط المرسومة لكل جولة، جنبًا إلى جنب مع اللحظات التي تم فيها أخذ الوكتات (المميزة بنقاط كبيرة)، رؤى حول كيفية تطور ديناميكيات المباراة:

- **تقلبات معدل النقاطفي RCB (RCB Run Rate Fluctuations):** يظهر معدل النقاط في RCB تقلبات كبيرة، حيث بلغ ذروته عند حوالي 3.5 جولة لكل كرة نحو نهاية الجولة. يشير وجود علامات الوكت (الدوائر الحمراء) إلى أن الويكيتات تم أخذها أثناء الجولات حيث كان معدل النقاط أقل عمومًا، وهو أمر طبيعي حيث تميل الوكتات إلى تعطيل تدفق الضرب.
- **أنماط معدل النقاطفي DC (DC Run Rate Patterns):** يبدأ معدل النقاطفي DC قويًا ولكنه يشهد انخفاضًا حادًا بعد الجولات الأولية، ويستقر إلى حد مافي المنتصف قبل ذروة أخرى وانخفاض لاحق نحو النهاية. يتم أخذ الوكتات (الدوائر الزرقاء)في الجولات حيث ينخفض معدل النقاط، مما يشير إلى رمي فعال من RCB خلال هذه الأوقات.

لذا، هذه هي الطريقة التي يمكنك بها تحليل مباراة كريكيت باستخدام علم البيانات.

الاستنتاج

تميزت المباراة بين RCB وDC بمزيج من الضرب الاستراتيجي والرمي العدواني والشراكات الحاسمة. ساهمت قدرة RCB على الحفاظ على معدل تراكمي أعلى من النقاط وفعالية أخذ الوكتات في الأشواط الحاسمة بشكل كبيرفي فوزهم. لا يسلط التحليل التفصيلي لكل مرحلة واللاعبين الضوء

على ديناميكيات لعبة الكريكيت T20 فحسب، بل يساعد أيضًافي فهم كيفية تأثير تحولات الزخم والقرارات الاستراتيجية على نتيجة اللعبة.

المصدر:

https://thecleverprogrammer.com/2024/05/20/ipl-2024-rcb-vs-dc-analysis-/using-python

17) تحليل بيانات رياضة الكريكيت في الدوري الهندي الممتاز للكريكيت IPL 2022 Cricket Sports Data Analysis

المقدمة

إن الدوري الهندي الممتاز للكريكيت IPL هو دوري كريكيت احترافي من فئة العشرين عامًا، أطلقه مجلس الكريكيت الهنديفي عام 2008، ويضم 10 فرق بقيمة علامة تجاريةفي عام 2022 تبلغ 11 مليار دولار.

دعونا نحللها إحصائيًا.

للحصول على تحليل مفصل، يمكنك الوصول إلى التقرير الكامل على مستودع GitHub الخاص بي.

إعداد البيانات وتنظيفها

معالجة مسبقة بـ Pandas، وتصور بـ Plotly.

```
# Import necessary libraries

import pandas as pd
import plotly.express as px
```

قراءة ملف البيانات في جوبيتر

```
# Read IPL match data from a CSV file

data=pd.read_csv("IPL_Match_2022.csv")
```

استكشاف مجموعة البيانات

```
# Get the dimensions of the dataset

data.head()
```

المخرجات:

	match_id	date	venue	team1	team2	stage	toss_winner	toss_decision	first_ings_score	first_ings_wkts	second_ings_score	second_ings
0	1	March 26,2022	Wankhede Stadium, Mumbai	Chennai	Kolkata	Group	Kolkata	Field	131	5	133	
1	2	March 27,2022	Brabourne Stadium, Mumbai	Delhi	Mumbai	Group	Delhi	Field	177	5	179	
2	3	March 27,2022	Dr DY Patil Sports Academy, Mumbai	Banglore	Punjab	Group	Punjab	Field	205	2	208	
3	4	March 28,2022	Wankhede Stadium, Mumbai	Gujarat	Lucknow	Group	Gujarat	Field	158	6	161	
4	5	March 29,2022	Maharashtra Cricket Association Stadium,Pune	Hyderabad	Rajasthan	Group	Hyderabad	Field	210	6	149	

الملاحظة:

- يعرض الناتج أول خمسة صفوف من مجموعة بيانات مباراة الدوري الهندي الممتاز IPL مع معلومات حول الفرق والأماكن ونتائج المباريات والأداء الملحوظ.
- يوفر لمحة عامة عن بُنية البيانات ومحتواها للفحص الأولي.

الحصول على أبعاد مجموعة البيانات

```
data.shape
```

المخرجات:

```
(74, 20)
```

الملاحظات:

- تحتوي مجموعة البيانات على 74 صفًا و20 عمودًا
- في موسم الدوري الهندي الممتاز لعام 2022، تم لعب 74 مباراة

إظهار معلومات مجموعة البيانات باستخدام أنواع البيانات

المخرجات:

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 74 entries, 0 to 73
Data columns (total 20 columns):
 #   Column               Non-Null Count  Dtype 
---  ------               --------------  ----- 
 0   match_id             74 non-null     int64 
 1   date                 74 non-null     object
 2   venue                74 non-null     object
 3   team1                74 non-null     object
 4   team2                74 non-null     object
 5   stage                74 non-null     object
 6   toss_winner          74 non-null     object
 7   toss_decision        74 non-null     object
 8   first_ings_score     74 non-null     int64 
 9   first_ings_wkts      74 non-null     int64 
 10  second_ings_score    74 non-null     int64 
 11  second_ings_wkts     74 non-null     int64 
 12  match_winner         74 non-null     object
 13  won_by               74 non-null     object
 14  margin               74 non-null     int64 
 15  player_of_the_match  74 non-null     object
 16  top_scorer           74 non-null     object
 17  highscore            74 non-null     int64 
 18  best_bowling         74 non-null     object
 19  best_bowling_figure  74 non-null     object
dtypes: int64(7), object(13)
memory usage: 11.7+ KB
```

الملاحظات:

- تؤكد الأعداد غير الصفرية عدم وجود بيانات مفقودة.
- أنواع البيانات: 7 أعداد صحيحة، 13 كائنًا ـ مما يشير إلى مزيج من البيانات الرقمية والفئوية numeric and categorical data.

```
# to fetch all column names

data.columns
```

المخرجات:

```
Index(['match_id', 'date', 'venue', 'team1', 'team2', 'stage', 'toss_winner',
'toss decision', 'first ings score', 'first ings wkts', 'second ings score',
'second_ings_wkts', 'match_winner', 'won_by', 'margin', 'player_of_match',
'top_scorer', 'highscore', 'best_bowling', 'best_bowling_figure'],
      dtype='object')
```

الملاحظات:

- توضح الأعمدة تفاصيل مباريات الدوري الهندي الممتاز، بمافي ذلك الفرق، والأماكن، ومعلومات القرعة، والنتائج، والفائزين، وإنجازات اللاعبين.

عد القيم المفقودة في كل عمود

```
# Data Gap Analysis

data.isnull().sum()
```

المخرجات:

```
match_id                0
date                    0
venue                   0
team1                   0
team2                   0
stage                   0
toss_winner             0
toss_decision           0
first_ings_score        0
first_ings_wkts         0
second_ings_score       0
second_ings_wkts        0
match_winner            0
won_by                  0
margin                  0
player_of_the_match     0
top_scorer              0
highscore               0
best_bowling            0
best_bowling_figure     0
dtype: int64
```

ملاحظة: لا توجد قيم مفقودةفي أي عمود (المجموع = 0).

التحليل الاستكشافي والتصور

تحليل أداء الفريق

تحليل إحصائيات الفريق.

عدد المباريات التي خاضها الفريق 1.

```
data['team1'].value_counts()
```

المخرجات:

```
Banglore      16
Chennai       12
Delhi         12
Gujarat       10
Hyderabad      7
Kolkata        7
Lucknow        6
Mumbai         3
Punjab         1
Name: team1, dtype: int64
```

الملاحظات:

- يوفر ملخصًا لعدد المرات التي ظهر فيها كل فريق من فرق الدوري الهندي الممتاز باسم "team1"في مجموعة البيانات.
- على سبيل المثال، ظهر " Banglore " 16مرة باسم "team1"، مما يشير إلى أن الفريق شارك في 16 مباراة باعتباره الفريق الأول. وبالمثل، ظهر "Chennai" و "Delhi" 12 مرة لكل منهما، وهكذا.

عدد المباريات التي لعبها الفريق2:

```
data['team2'].value_counts()
```

المخرجات:

```
Rajasthan     17
Punjab        13
Mumbai        11
Lucknow        9
Kolkata        7
Hyderabad      7
Gujarat        6
Chennai        2
Delhi          2
Name: team2, dtype: int64
```

الملاحظة:

- توزيع الفرق التي حلتفي المركز الثاني (team2)في مباريات الدوري الهندي الممتاز
- ظهر اسم "Rajasthan" 17 مرةفي المركز الثاني، مما يشير إلى مشاركتهمفي المركز الثانيفي 17 مباراة. وعلى نحو مماثل، ظهر اسم "Punjab" 13 مرة، و"Mumbai" 11 مرة، وهكذا.

إجمالي عدد المباريات حسب الفريق:

```
data['team1'].value_counts() + data['team2'].value_counts()
```

المخرجات:

```
Banglore       NaN
Chennai       14.0
```

```
Delhi         14.0
Gujarat       16.0
Hyderabad     14.0
Kolkata       14.0
Lucknow       15.0
Mumbai        14.0
Punjab        14.0
Rajasthan      NaN
dtype: float64
```

الملاحظات:

- Rajasthan وBanglore لديهما قيمة NaN. وهذا يعني أنهما لعبا جميع مبارياتهما كـ "team1" أو "team2"، ولكن ليس كلاهما.
- تمثل القيم الرقمية العدد الإجمالي للمرات التي ظهر فيها كل فريق إما كـ "team1" أو "team2"في مجموعة البيانات، مما يشير إلى مشاركتهما الإجماليةفي مباريات الدوري الهندي الممتاز خلال موسم 2022.

استخدم NaNفي حساب إجمالي المباريات:

```
total_matches_played =
data['team1'].value_counts().add(data['team2'].value_counts(),
fill_value=0).astype(int)
```

المخرجات:

```
Banglore      16
Chennai       14
Delhi         14
Gujarat       16
Hyderabad     14
Kolkata       14
Lucknow       15
Mumbai        14
Punjab        14
Rajasthan     17
dtype: int32
```

شرح الكود :

- add(): لإضافة "قيم" سلسلتين.
- fill_value=0 يستخدم لتحديد أنه إذا كان الفريق مفقودًافي إحدى السلسلتين (مما يؤدي إلى قيمة NaN)، فيجب التعامل معه على أنه 0 أثناء الإضافة
- astype(int) : تحويل النتيجة إلى عدد صحيح

الملاحظات:

- في عام 2022، شاركت فرق الدوري الهندي الممتاز على النحو التالي:
- لعب فريق " Lucknow" 15 مرة، بينما تصدر فريق "Rajasthan" بـ 17 مباراة.

- وشاركت الفرق المتبقيةفي 14 مباراة لكل منها.

احسب إجمالي المباريات التي فاز بها كل فريق:

```
total_won= data['match_winner'].value_counts()
```

المخرجات:

```
Gujarat      12
Rajasthan    10
Banglore      9
Lucknow       9
Delhi         7
Punjab        7
Kolkata       6
Hyderabad     6
Chennai       4
Mumbai        4
Name: match_winner, dtype: int64
```

الملاحظات:

- عدد المباريات التي فاز بها كل فريق من فرق الدوري الهندي الممتاز خلال موسم 2022:
- حققت ولاية Gujarat أعلى عدد من الانتصارات، بواقع 12 انتصارًا.
- وتبعتها ولاية Rajasthan بفارق كبير بواقع 10 انتصارات.
- Banglore وLucknow (9)، وDelhi وPunjab (7)، وKolkata وHyderabad (6)، وChennai وMumbai (4).

احسب نسبة الفوز لكل فريق:

```
win_percentage = ((total_won / total_matches_played) *
100).sort_values(ascending=False).astype(int)
```

المخرجات:

```
Gujarat      75
Lucknow      60
Rajasthan    58
Banglore     56
Delhi        50
Punjab       50
Hyderabad    42
Kolkata      42
Chennai      28
Mumbai       28
dtype: int32
```

الملاحظات:

- تعكس هذه النسبة معدلات نجاح فرق الدوري الهندي الممتازفي تأمين الانتصارات.
- قائد نسبة الفوز: Gujarat (75%)
- تلي Lucknow عن كثب بنسبة فوز تبلغ 60%.

- تحافظ Rajasthan على نسبة فوز ثابتة تبلغ 58%
- Banglore وDelhi (56%)، Punjab وHyderabad(50%)، Kolkata وChennai (42%)، Mumbai (28%)

قم بإنشاء إطار بيانات لتخزين بيانات أداء الفريق وفرزها حسب نسبة الفوز (تنازليًا):

```
team_performance = pd.DataFrame({
'Total Matches Played': total_matches_played,
'Total Matches Won': total_won,
'Win Percentage (%)': win_percentage
}).sort_values(by='Win Percentage (%)', ascending=False)
```

المخرجات:

	Total Matches Played	Total Matches Won	Win Percentage (%)
Gujarat	16	12	75
Lucknow	15	9	60
Rajasthan	17	10	58
Banglore	16	9	56
Delhi	14	7	50
Punjab	14	7	50
Hyderabad	14	6	42
Kolkata	14	6	42
Chennai	14	4	28
Mumbai	14	4	28

الملاحظات:

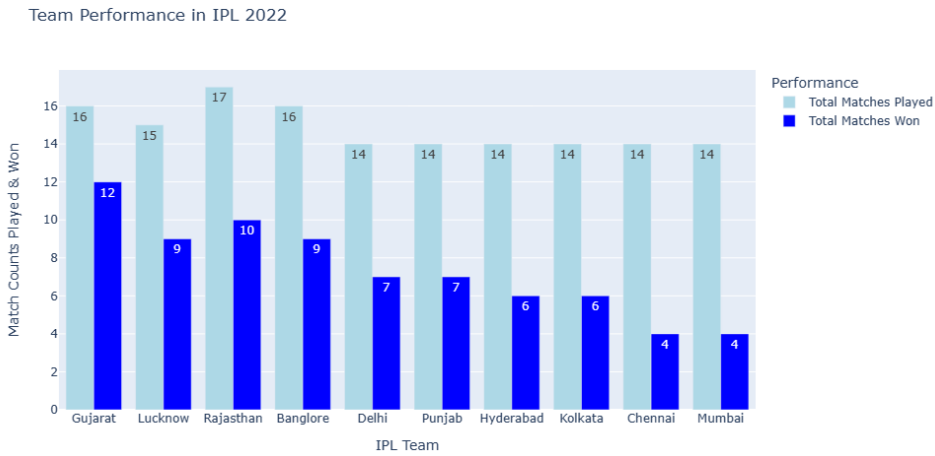
- يلخص إطار البيانات هذا أداء فرق الدوري الهندي الممتازفي عام 2022.
- لعبت Gujarat 16 مباراة وفازت بـ 12 منها، مما أدى إلى فوز بنسبة 75%.
- حققت Lucknow نسبة فوز 60%، حيث فازت بـ 9 من أصل 15 مباراة.
- لعبت Rajasthan أكبر عدد من المباريات (17) بنسبة فوز 58% (10 انتصارات)
- Banglore وDelhi: نسبة فوز 56% (9 انتصاراتفي 16 مباراة لكل منهما).
- Punjab: نسبة فوز 50% (7 من أصل 14).
- "Hyderabad" و"Kolkata": نسبة فوز 42% (6 انتصاراتفي 14 مباراة لكل منهما).
- "Chennai" و"Mumbai": نسبة فوز 28% (4 انتصاراتفي 14 مباراة لكل منهما).

مخطط شريطي Bar Chart لأداء الفريق:

```
px.bar( team_performance,
x=team_performance.index,
y=['Total Matches Played', 'Total Matches Won'],
text_auto=True, barmode='group',
title='Team Performance in IPL 2022',
```

```
labels={'index':'IPL Team'},
color_discrete_map={'Total Matches Played': 'lightblue', 'Total Matches
Won': 'blue'}
).update_layout(
legend_title_text='Performance',yaxis_title='Match Counts')
```

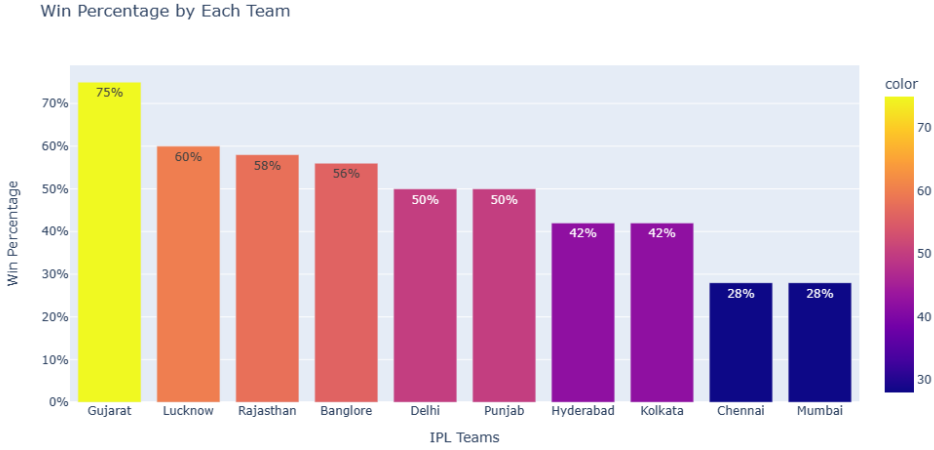
المخرجات:



تصور نسبة الفوز لكل فريق:

```
# Create a bar chart
px.bar(
win_percentage,
x=win_percentage.index,
y=win_percentage,
labels={'index': 'IPL Teams', 'y': 'Win Percentage'},
title='Win Percentage by Each Team',
text_auto=True, color=win_percentage
).update_layout(yaxis_ticksuffix='%')
```

المخرجات:



أفضل لاعب في المباراة

أفضل 10 لاعبينفي المباراة:

```
pom =
data.groupby('player_of_match')['match_id'].count().sort_values(ascending=F
alse)[:10]
```

المخرجات:

```
player_of_the_match
Kuldeep Yadav       4
Jos Buttler         3
Umesh Yadav         2
Rahul Tripathi      2
Shubman Gill        2
K L Rahul           2
Quinton de Kock     2
Jasprit Bumrah      2
Hardik Pandya       2
Yuzvendra Chahal    2
Name: match_id, dtype: int64
```

الملاحظات:

- أفضل 10 لاعبين حصلوا على جائزة "أفضل لاعبفي المباراة 'player_of_the_match award":
- حصل "Kuldeep Yadav" على الجائزة 4 مرات، مما يجعله اللاعب الأكثر حصولاً على جوائز "أفضل لاعبفي المباراة".
- حصل "Jos Buttler" على 3 جوائز، وحصل 8 لاعبين آخرين على جائزتين لكل منهم.

التحقق من النتائج:

احسب عدد المرات التي حصل فيها "player_of_the_match'" على جائزة "أفضل لاعبفي المباراة"

```
data[data['player_of_the_match']==('Jos Buttler')]['match_id'].count()
```

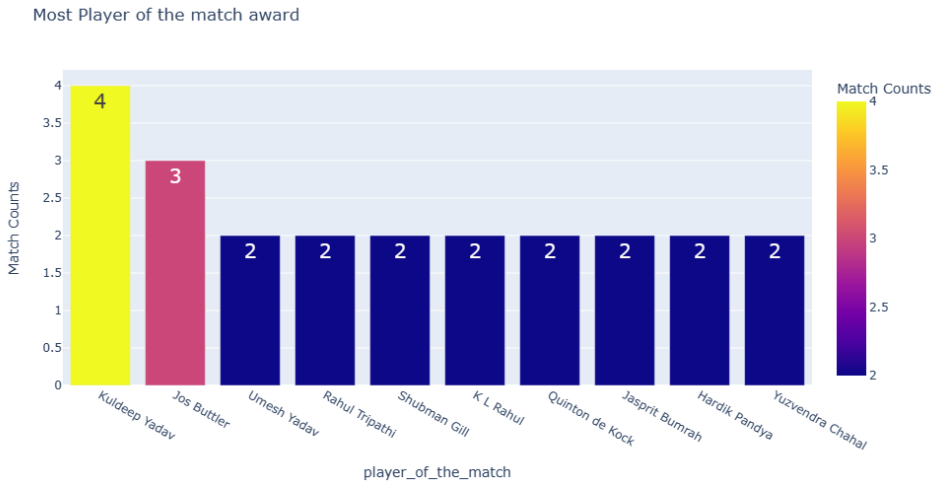
المخرجات:

```
3
```

رسم الرؤىفي المخطط الشريطي:

```
px.bar(pom, y ='match_id',
text='match_id',
color='match_id',
title='Most Player of the match award',
labels={'match_id':'Match Counts'}).update_traces(textfont_size=20)
```

المخرجات:



أفضل هداف في الدوري الهندي الممتاز 2022

احسب إجمالي أعلى الدرجات لكل لاعب واعرض أفضل 10 لاعبين:

```
score =
data.groupby('top_scorer')['highscore'].sum().sort_values(ascending=False)[
:10]
```

المخرجات:

```
top_scorer
Jos Buttler          651
Quinton de Kock      377
KL Rahul             351
Shubman Gill         288
Faf du Plessis       257
Liam Livingstone     239
David Warner         213
W. Saha              190
Ishan Kishan         180
Shikhar Dhawan       158
Name: highscore, dtype: int64
```

الملاحظات:

- أفضل هدافي الدوري الهندي الممتاز لعام 2022:
- يتصدر "Jos Buttler " القائمة بإجمالي 651 نقطة.
- يأتي "Quinton de Kock "في المركز الثاني بإجمالي 377 نقطة.
- "KL Rahul " بإجمالي 351 نقطة، من بين آخرين.

التحقق من النتائج:

احسب إجمالي النقاط العالية للاعب معين (على سبيل المثال، Jos Buttler):

```
data[data['top_scorer']==('Jos Buttler')]['highscore'].sum()
```

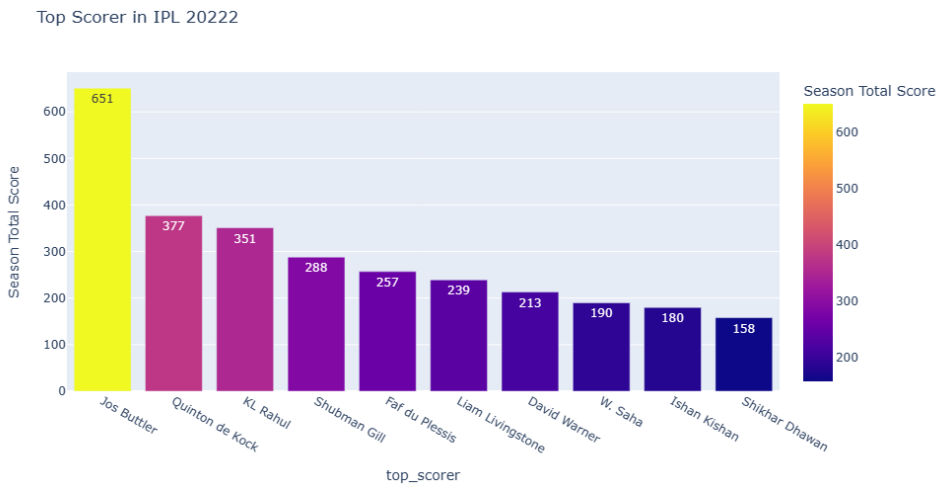
المخرجات:

```
651
```

رسم الرؤىفي المخطط الشريطي:

```
px.bar(score, y='highscore',color='highscore',
labels={'highscore':'Season Total Score'},
title='Top Scorer in IPL 20222',text='highscore')
```

المخرجات:



تحليل القرعة

أي فريق فاز بأكبر عدد من القرعة؟

عدد مرات فوزكل فريقفي الدوري الهندي الممتاز بالقرعة؟

```
data['toss_winner'].value_counts()
```

المخرجات:

```
Gujarat       10
Hyderabad     10
Mumbai         9
Kolkata        8
Delhi          8
Banglore       8
Lucknow        7
Chennai        6
Punjab         4
Rajasthan      4
Name: toss_winner, dtype: int64
```

الملاحظات:

- نجاح الفرقفي الفوز بقرعة العملة:

- حقق فريقا "Gujarat" و"Hyderabad" أكبر نجاحفي الفوز بقرعة العملة، حيث فاز كل منهما بها 10 مرات.
- حقق فريق "Mumbai" 9 انتصارات، وحقق العديد من الفرق الأخرى 8 انتصاراتفي قرعة العملة. بينما حقق فريقا "Punjab" و"راجاستان" أقل عدد من الانتصاراتفي قرعة العملة، حيث فاز كل منهما بـ 4 انتصارات، بينما حقق فريقا "Rajasthan" أقل عدد من الانتصارات في قرعة العملة.

التحقق من النتائج:

احسب عدد المباريات التي فاز بها فريق "Gujarat " بعد الفوز بقرعة العملة.

```
data[data['toss_winner']==('Gujarat')]['match_id'].count()
```

المخرجات:

```
10
```

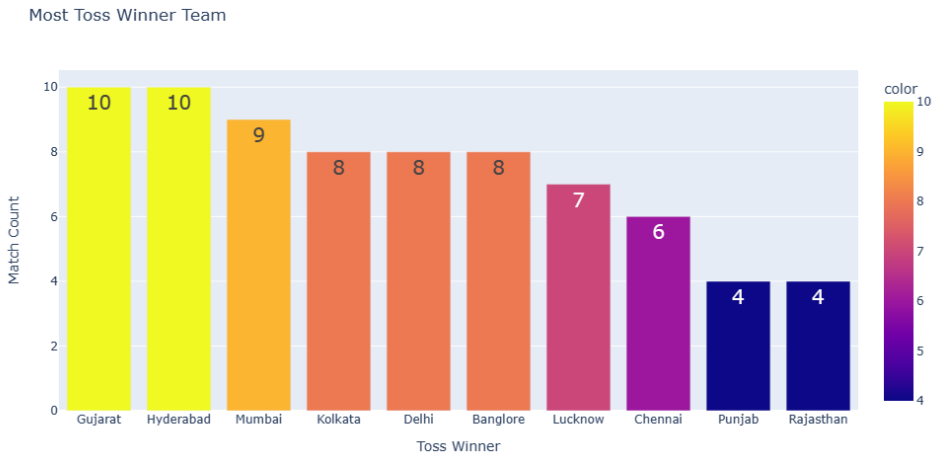
الرسم البيانيفي شريط المخطط الشريطي:

```
# visualize Toss Wins per Team

x = data['toss_winner'].value_counts().keys()
y = data['toss_winner'].value_counts()

px.bar(data, x=x, y =y, text=y, color=y, title='Most Toss Winner Team',
labels={'x':'Toss Winner', 'y':'Match
Count'}).update_traces(textfont_size=20)
```

المخرجات:



نسبة الفريق الفائز بالقرعة والفائز بالمباراة أيضًا

اكتشف عدد المرات التي فاز فيها الفريق بالمباراة بعد الفوز بالقرعة

```
toss_match_won = data[data['toss_winner'] ==
data['match_winner']]['match_winner'].value_counts()
```

المخرجات:

```
Gujarat      7
Kolkata      5
Hyderabad    5
Delhi        4
Banglore     4
Lucknow      4
Mumbai       3
Rajasthan    2
Punjab       1
Chennai      1
Name: match_winner, dtype: int64
```

الملاحظة: تصدر فريق Gujarat الترتيب بـ 7 انتصارات، يليه فريق Kolkata وHyderabad أباد بـ 5 انتصارات لكل منهما. كما سجلت العديد من الفرق الأخرى أعدادًا متفاوتة من الانتصارات.

احسب إجمالي عدد المباريات التي فازت بها؟

```
match_won = data['match_winner'].value_counts()
```

المخرجات:

```
Gujarat      12
Rajasthan    10
Banglore      9
Lucknow       9
Delhi         7
Punjab        7
Kolkata       6
Hyderabad     6
Chennai       4
Mumbai        4
Name: match_winner, dtype: int64
```

حساب نسبة المباريات التي فاز بها الفريق عندما يفوز بالقرعة والمباراة:

```
percentage_won = (toss_match_won / match_won * 100).astype(int)
```

المخرجات:

```
Hyderabad    83
Kolkata      83
Mumbai       75
Gujarat      58
Delhi        57
Banglore     44
Lucknow      44
Chennai      25
Rajasthan    20
Punjab       14
Name: match_winner, dtype: int32
```

الملاحظة:

- معدلات النجاح في الفوز بالمباريات بعد الفوز بالقرعة:

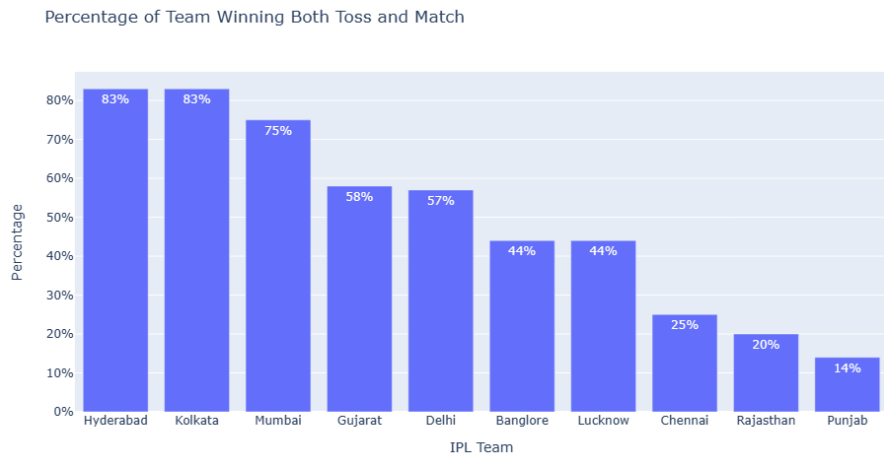
- "Hyderaba " و"Kolkata"في المقدمة بنسبة 83%، يليهما "Mumbai" بنسبة 75%، و" Punjab" بنسبة 14%، وهي الأقل.

تصور نسبة الفوز بالقرعة = الفوز بالمباراة

```
# Plotting with Bar Chart

px.bar(
percentage won,
x=percentage_won.index,
y=percentage_won,
text_auto=True,
title='% of Team Winning Both Toss & Match',
labels={'index': 'IPL Team', 'y': 'Percentage'}
).update_layout(yaxis_ticksuffix='%')
```

المخرجات:



تحليل الضرب والرمي

الدفاع مقابل مطاردة الانتصارات

مباريات الدفاع (الضرب أولاً) التي فاز بها الفريق بعدد الأشواط، ومباريات المطاردة (الضرب ثانياً أي أولًا) التي فاز بها الفريق بعدد الويكيتات.

إحصاء قيم "won_by"

```
# Runs vs. Wickets: Win Count

data["won_by"].value_counts()
```

المخرجات:

```
Wickets    37
Runs       37
Name: won_by, dtype: int64
```

الملاحظات:

- التوزيع المتساوي لنتائج المباريات: فازت الفرق التي تدافع عن نفسها بالنقاطفي 37 مباراة، وفازت الفرق التي تطارد المنافسين بأخذ الويكيتاتفي 37 مباراة.
- مما يوضح المنافسة المتوازنة بين الفرق التي تضرب أولاً وتلك التي تطارد المنافسينفي الموسم.

تحقق من النتائج المذكورة أعلاه

الضرب أولاً: عدد مرات الفوزفي المباريات

```
data[data['won_by']=='Runs']['match_id'].count()
```

المخرجات:

```
37
```

تصور الرؤيةفي مخطط دائري:

```
# distribution of matches won by defending (Runs) and chasing (Wickets)

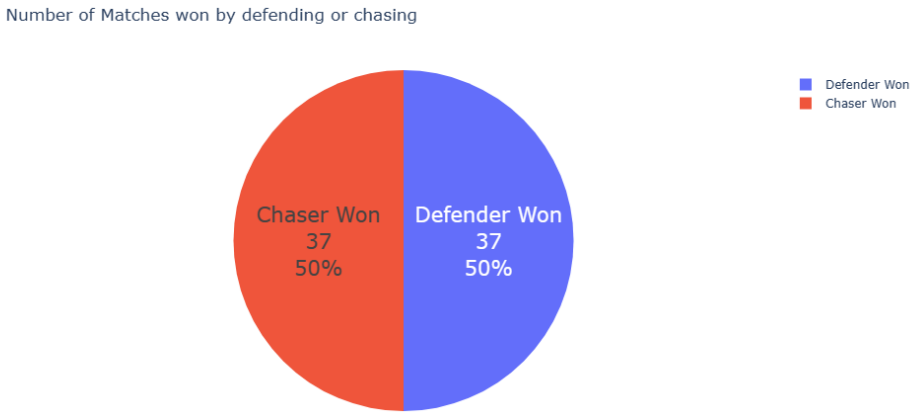
v = data["won_by"].value_counts()

fig= px.pie(data, values= v,
names=['Defender Won','Chaser Won'],
title='Number of Matches won by defending or chasing')

fig.update_traces(textposition='inside', textinfo='percent+label+value',
textfont_size=21)

fig.show()
```

المخرجات:



بيانات هامش فوز المدافع

```
# Retrieve & sort 'Runs' Win Margins Descending

data[data['won_by']=='Runs']['margin'].sort_values(ascending=False).values
```

المخرجات:

```
array([91, 75, 67, 62, 61, 54, 54, 54, 52, 44, 37, 36, 29, 24, 23, 23, 21,
20, 18, 18, 17, 16, 15, 14, 14, 13, 13, 12, 12, 11,  8,  7,  6,  5, 3,  3,
2], dtype=int64)
```

الملاحظات:

- هامش الفوز الذي حققته الفرق التي ضربت أولاً: تراوح من فوز بفارق هدفين إلى فوز ساحق بفارق 91 هدفًا.
- شهد الموسم مباريات متقاربة ومهيمنة، مما يؤكد على تنافسية البطولة.

قم بإنشاء إحصائيات وصفية لهامش الفوزفي المباريات التي فازت بها الفرق التي سجلت أهدافًا

```
data[data['won_by'] == 'Runs']['margin'].describe().astype(int)
```

المخرجات:

```
count    37
mean     27
std      23
min       2
25%      12
50%      18
75%      44
max      91
Name: margin, dtype: int32
```

الملاحظات:

- كانت هوامش الفوزفي المباريات التي خاضها الفريق الأول متباينة، حيث بلغ متوسطها 27 نقطة.
- وكان أوسع هامش هو 91 نقطة.
- يشير الانحراف المعياري المرتفع الذي بلغ 23 نقطة مقارنة بالمتوسط إلى هوامش فوز متباينة، تتراوح بين تقدم ضئيل إلى تقدم كبير. ويضيف هذا المزيج من النتائج الإثارة وعدم القدرة على التنبؤ بالبطولة.
- تراوحت هوامش الفوز من 2 إلى 91 نقطة، مما يدل على المباريات التنافسية.
- وكانت هوامش الفوزفي 75% من المباريات 44 نقطة أو أقل.

تصور هامش الفوز الأولفي الضرب

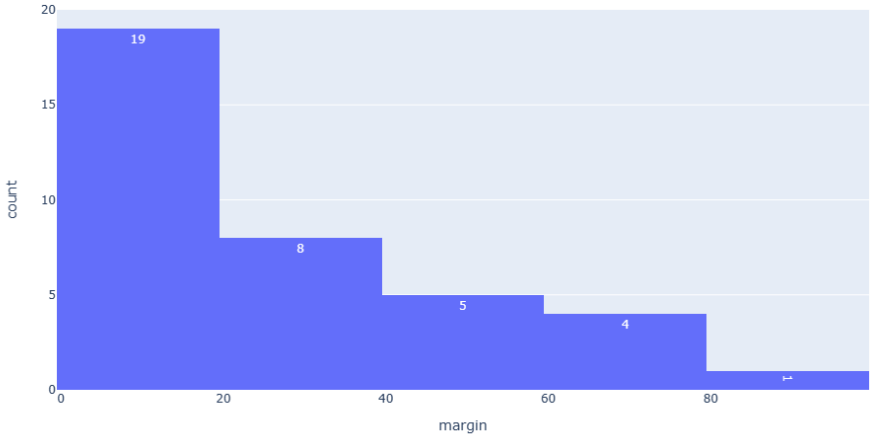
```
# Filter Matches Won by 'Runs'

Defender = data[data['won_by']=='Runs']

# Create a histogram

px.histogram(Defender, x="margin", text_auto=True)
```

المخرجات:



أفضل الرماة في الدوري الهندي الممتاز 2022

احصِ أفضل الرماةفي قائمة أفضل 10 لاعبين

```
bowler =
data.groupby('best_bowling')['match_id'].count().sort_values(ascending=Fals
e)[:10]
```

المخرجات:

```
best_bowling
Yuzvendra Chahal     5
Rashid Khan          4
Kuldeep Yadav        3
T Natarajan          3
Avesh Khan           3
Jasprit Bumrah       3
Josh Hazlewood       3
Kagiso Rabada        3
Wanindu Hasaranga    2
Umran Malik          2
Name: match_id, dtype: int64
```

الملاحظات:

- كان Yuzvendra Chaha لاعبًا متميزًا حيث فاز بخمس جوائز "أفضل لاعب بولينج".
- لم يكن Rashid Khan بعيدًا عنه، حيث فاز بلقب "أفضل رامي best bowler" أربع مرات.
- مع ذلك، قدم العديد من الرماة الآخرين مساهمات ملحوظة أيضًا

التحقق من النتائج

أفضل أداء لـKuldeep Yadav في الرمي: Count

```
data[data['best_bowling']==('Kuldeep Yadav')]['match_id'].count()
```

المخرجات:

```
3
```

تصور أفضل الرماة في IPL 2022

```
# Ploting the insight in Bar Graph

px.bar(bowler, y ='match_id', text='match_id',color='match_id',
       title='Best Bowler in IPL 2022',
       labels={'match_id':'Match count'}).update_traces(textfont_size=20)
```

المخرجات:

تحليل الملعب

أفضل ملعب لأكبر عدد من المباريات

```
# Venue-wise Match Total

data['venue'].value_counts()
```

المخرجات:

```
Wankhede Stadium, Mumbai              21
Dr DY Patil Sports Academy, Mumbai    20
Brabourne Stadium, Mumbai             16
MCA Stadium,Pune                      13
Eden Gardens, Kolkata                 2
Narendra Modi Stadium, Ahmedabad      2
Name: venue, dtype: int64
```

الملاحظات:

- استضاف ملعب Wankhede وDY Patil في مومباي أكثر من 20 مباراة لكل منهما، وهما الاختياران الأفضل في الدوري الهندي الممتاز.

- واستضاف ملعب Brabourne(مومباي) وملعب Pune's MCA 16 و13 مباراة، وهو حضور كبير.
- ولم يكن لدور Kolkata وAhmedabad سوى دور ضئيل، حيث استضافت كل منهما مباراتين فقط.
- يسلط توزيع الملاعب الضوء على الملاعب الرئيسيةفي مومباي، ويبرز تنوع المدن المضيفة، ويؤكد على مراكز الكريكيت المحوريةفي الدوري الهندي الممتاز 2022.

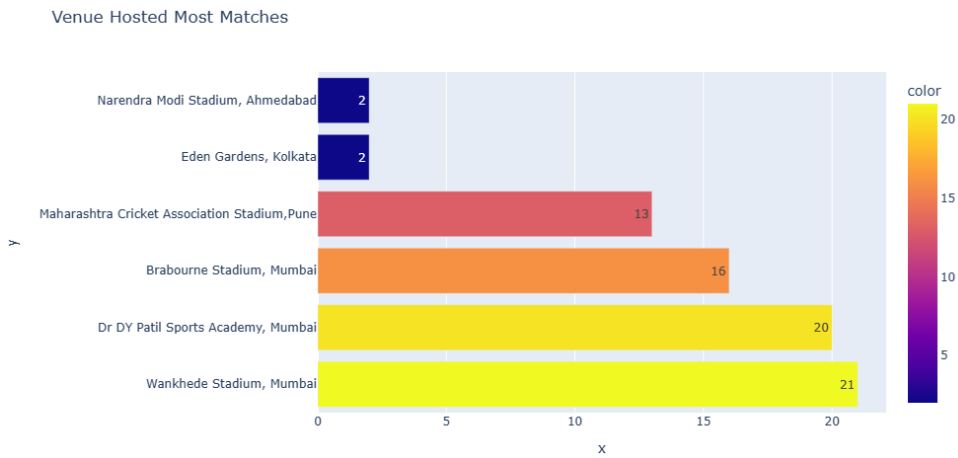
```
# Extract venue names and their respective match counts

x = data['venue'].value_counts().keys()
y = data['venue'].value_counts()

# Bar Chart: Top Match Hosting Venues

px.bar(data, x=y, y=x,text=y, color=y,
       title='Venue Hosted Most Matches')
```

المخرجات:



الاستنتاجات والاستنتاجات

النقاط الرئيسية من EDA:

- سيطرت جوجارات على المباراة بـ 12 فوزًا.
- تصدر Jos Buttle قائمة اللاعبين الذين حصلوا على نقاطفي المباريات.
- استضافت ملاعب Mumbai معظم المباريات.
- فازت Gujarat وHyderabad بالقرعة بشكل متكرر.
- تقاسمت الفرق الفوز بالتساوي بين الدفاع والمطاردة.
- أضافت هوامش الفوز المتنوعة الإثارة.

- برز Yuzvendra Chahal وRashid Khan بين افضل الرماة.

المصدر:

https://www.linkedin.com/pulse/python-practice-project-ipl-2022-cricket-sports-data-analysis-mishra

18) تحليل كأس العالم T20 2022 باستخدام بايثون T20 World Cup 2022 Analysis using Python

يولد كل حدث رياضي الكثير من البيانات التي يمكننا استخدامها لتحليل أداء اللاعبين والفرق والعديد من الأحداث البارزةفي اللعبة. نظرًا لأن بطولة كأس العالم للرجال T20 من المجلس الدولي للكريكيت ICC قد انتهت للتو، فقد ولدت الكثير من البيانات التي يمكننا استخدامها لتلخيص الحدث. لذا، إذا كنت تريد معرفة كيفية تحليل حدث رياضي مثل كأس العالم T20، فهذه المقالة لك. ستأخذك هذه المقالة خلال مهمة تحليل كأس العالم T20 2022 باستخدام بايثون.

مجموعة البيانات

يتم جمع مجموعة البيانات التي أستخدمها لتحليل كأس العالم T20 2022 يدويًا. يمكنك تنزيل مجموعة البيانات من هنا.

الآن دعنا نبدأ هذه المهمة باستيراد مكتبات بايثون الضرورية ومجموعة البيانات:

```
import pandas as pd
import plotly.express as px
import plotly.graph_objects as go
import plotly.io as pio
pio.templates.default = "plotly_white"

data = pd.read_csv("t20 world cup 22.csv")
print(data.head())
```

```
              venue       team1       team2     stage  toss winner  \
0               SCG  New Zealand   Australia  Super 12    Australia
1    Optus Stadium  Afghanistan     England  Super 12      England
2  Blundstone Arena      Ireland   Sri lanka  Super 12      Ireland
3               MCG     Pakistan       India  Super 12        India
4  Blundstone Arena   Bangladesh  Netherlands  Super 12  Netherlands

  toss decision  first innings score  first innings wickets  \
0         Field                200.0                    3.0
1         Field                112.0                   10.0
2           Bat                128.0                    8.0
3         Field                159.0                    8.0
4         Field                144.0                    8.0

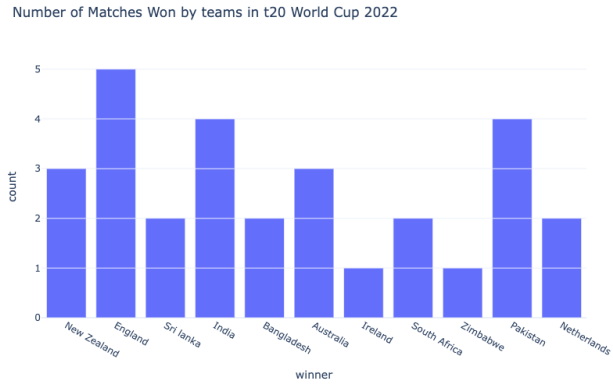
   second innings score  second innings wickets      winner   won by  \
0                 111.0                    10.0  New Zealand     Runs
1                 113.0                     5.0     England  Wickets
2                 133.0                     1.0   Sri lanka  Wickets
3                 160.0                     6.0       India  Wickets
4                 135.0                    10.0  Bangladesh     Runs

  player of the match       top scorer  highest score         best bowler  \
0       Devon Conway      Devon Conway           92.0        Tim Southee
1        Sam Curran   Ibrahim Zadran           32.0         Sam Curran
2       Kusal Mendis     Kusal Mendis           68.0  Maheesh Theekshana
3        Virat Kohli      Virat Kohli           82.0       Hardik Pandya
4       Taskin Ahmed  Colin Ackermann           62.0       Taskin Ahmed

  best bowling figure
0                3-6
1               5-10
2               2-19
3               3-30
4               4-25
```

الآن دعونا نلقي نظرة على عدد المباريات number of matches التي فاز بها كل فريق في كأس العالم:

```
figure = px.bar(data ,
                x=data["winner"],
                title="Number of Matches Won by teams in t20 World Cup
2022")
figure.show()
```

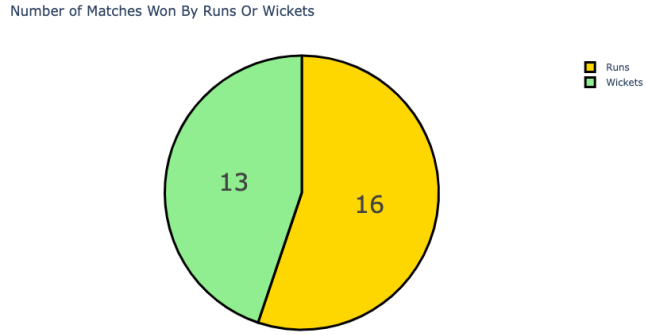


بما أن إنجلترا فازت بكأس العالم t20 2022، فقد فازت إنجلترا بخمس مباريات. وفازت كل من باكستان والهند بأربع مباريات.

الآن دعونا نلقي نظرة على عدد المباريات التي فاز بها الفريق الذي احتل المركز الأول أو الثاني في كأس العالم t20 2022:

```
won_by = data["won by"].value_counts()
label = won_by.index
counts = won_by.values
colors = ['gold','lightgreen']

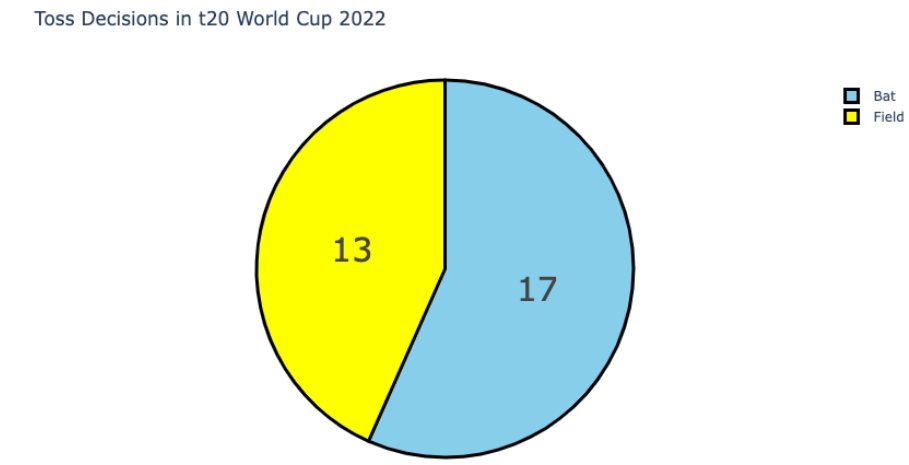
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Number of Matches Won By Runs Or Wickets')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black',
width=3)))
fig.show()
```



لذافي كأس العالم 2022، تم الفوز بـ 16 مباراة عن طريق الضرب أولاً، وتم الفوز بـ 13 مباراة عن طريق المطاردة. الآن، دعونا نلقي نظرة على قرارات القرعة toss decisions التي اتخذتها الفرقفي كأس العالم:

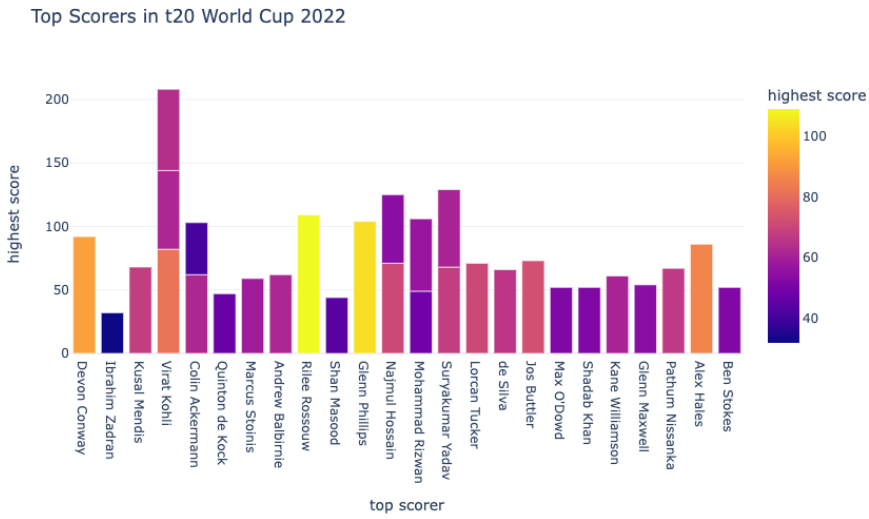
```
toss = data["toss decision"].value_counts()
label = toss.index
counts = toss.values
colors = ['skyblue','yellow']

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Toss Decisions in t20 World Cup 2022')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
textfont_size=30,
                  marker=dict(colors=colors, line=dict(color='black',
width=3)))
fig.show()
```



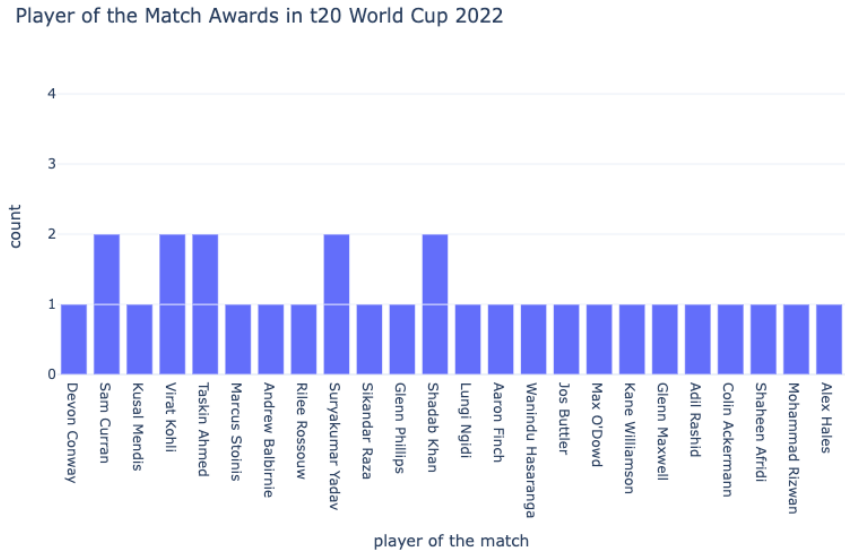
لذافي 17 مباراة، قررت الفرق أن تضرب أولاً bat first، وفي 13 مباراة، اختارت الفرق أن تلعب أولاً. الآن دعونا نلقي نظرة على أفضل الهدافينفي كأس العالم T20 2022:

```
figure = px.bar(data,
                x=data["top scorer"],
                y = data["highest score"],
                color = data["highest score"],
                title="Top Scorers in t20 World Cup 2022")
figure.show()
```



لذا، سجل فيرات كوهلي Virat Kohli أعلى نتيجةفي 3 مباريات. ولا شك أنه كان أفضل لاعبفي كأس العالم T20 2022. والآن دعونا نلقي نظرة على عدد جوائز أفضل لاعبفي المباراةفي كأس العالم:

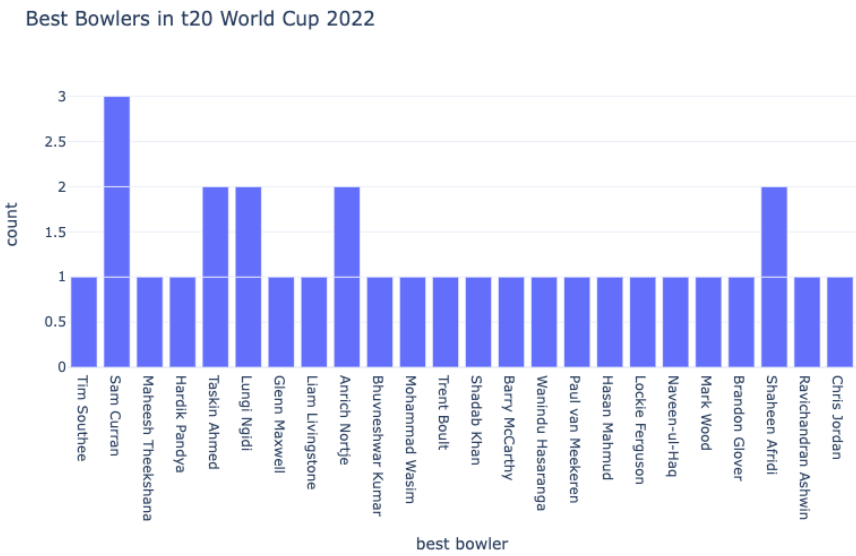
```
figure = px.bar(data,
                x = data["player of the match"],
                title="Player of the Match Awards in t20 World Cup 2022")
figure.show()
```



حصل فيرات كوهلي وسام كوران وتسكين أحمد وسورياكومار ياداف وشاداب خان على جائزة أفضل لاعبفي المباراةفي مباراتين. ولم يحصل أي لاعب على جائزة أفضل لاعبفي المباراةفي أكثر من مباراتين.

الآن دعونا نلقي نظرة على الرماة bowlers الذين حققوا أفضل أرقامفي الرمي bowlingفي نهاية المباريات:

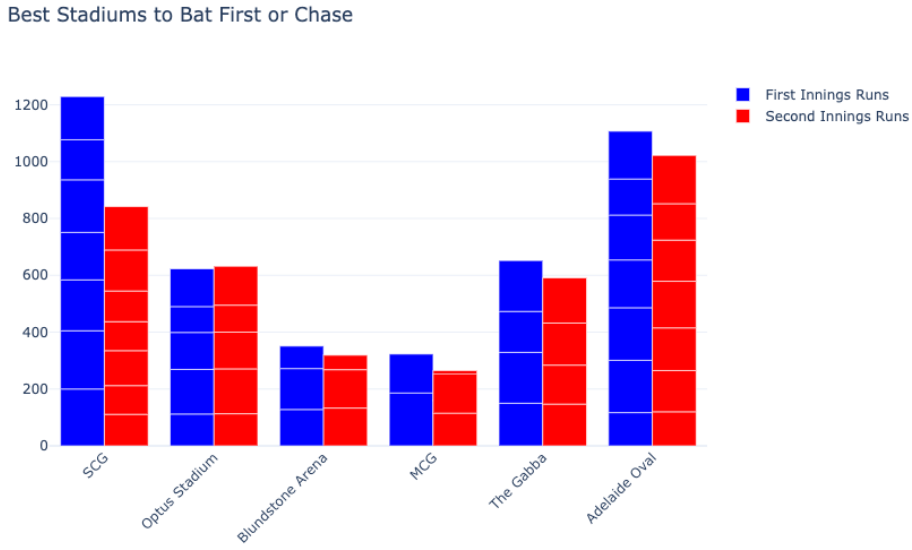
```
figure = px.bar(data,
                x=data["best bowler"],
                title="Best Bowlers in t20 World Cup 2022")
figure.show()
```



كان سام كوران هو أفضل لاعب بولينجفي 3 مباريات. بلا شك، كان يستحق أن يكون أفضل لاعبفي البطولة. الآن دعونا نقارن بين الأشواط المسجلةفي الشوط الأول first innings والشوط الثاني second inningsفي كل ملعب من ملاعب كأس العالم T20 2022:

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=data["venue"],
    y=data["first innings score"],
    name='First Innings Runs',
    marker_color='blue'
))
fig.add_trace(go.Bar(
    x=data["venue"],
    y=data["second innings score"],
    name='Second Innings Runs',
    marker_color='red'
))
fig.update_layout(barmode='group',
```

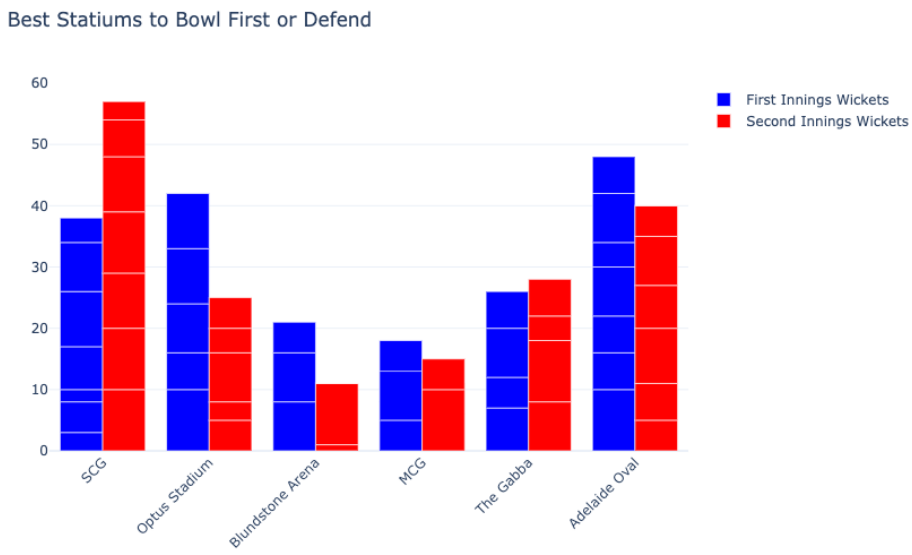
```
                  xaxis_tickangle=-45,
                  title="Best Stadiums to Bat First or Chase")
fig.show()
```



لذا كان ملعب سانت غالن SCG هو الملعب الوحيدفي كأس العالم الذي كان الأفضلفي ضرب الكرة أولاً. أما الملاعب الأخرى فلم يكن لها أي فارق كبيرفي ضرب الكرة أولاً أو مطاردة الكرة.

الآن دعونا نقارن عدد الويكيتات wickets المفقودةفي الشوط الأول والشوط الثانيفي كل ملعب من ملاعب كأس العالم T20 2022:

```
fig = go.Figure()
fig.add_trace(go.Bar(
    x=data["venue"],
    y=data["first innings wickets"],
    name='First Innings Wickets',
    marker_color='blue'
))
fig.add_trace(go.Bar(
    x=data["venue"],
    y=data["second innings wickets"],
    name='Second Innings Wickets',
    marker_color='red'
))
fig.update_layout(barmode='group',
                  xaxis_tickangle=-45,
                  title="Best Statiums to Bowl First or Defend")
fig.show()
```



كان ملعب SCG هو أفضل ملعب للعب البولينج أثناء الدفاع عن الهدف، بينما كان ملعب Optus هو أفضل ملعب للعب البولينج أولاً bowl first.

الخلاصة

إذن، بعض النقاط البارزةفي كأس العالم t20 2022 التي توصلنا إليها من تحليلنا هي:

- فازت إنجلترا بأكبر عدد من المباريات.
- حقق فيرات كوهلي أعلى نتيجةفي أكبر عدد من المباريات.
- كان سام كوران أفضل لاعب بولينجفي أكبر عدد من المباريات.
- فازت المزيد من الفرق بالضرب أولاً.
- قررت المزيد من الفرق الضرب أولاً.
- كان ملعب SCG هو أفضل ملعب للضرب أولاً.
- كان ملعب SCG هو أفضل ملعب للدفاع عن الهدففي كأس العالم.
- كان ملعب Optus هو أفضل ملعب للرمي أولاً.

المصدر:

https://thecleverprogrammer.com/2022/11/21/t20-world-cup-2022-analysis-/using-python

19) تحليل أداء فيرات كوهلي باستخدام بايثون Virat Kohli Performance Analysis using Python

يعد تحليل أداء اللاعب (player's performance) إحدى حالات استخدام علم البياناتفي التحليلات الرياضية (sports analytics). يعتبر فيرات كوهلي (Virat Kohli) أحد أشهر لاعبي الكريكيتفي العالم. لذلك سيكون مشروع علم بيانات رائعًا إذا قمنا بتحليل أداء الضرب batting performance لـ Virat Kohli على مر السنين. لذلك إذا كنت تريد معرفة كيفية تحليل أداء Virat Kohli، فهذه المقالة مناسبة لك.في هذه المقالة، سوف آخذك خلال مهمة تحليل أداء Virat Kohli باستخدام بايثون.

تحليل أداء Virat Kohli (دراسة حالة)

يعتبر Virat Kohli أحد أشهر لاعبي الكريكيتفي العالم. هنا تحصل على مجموعة بيانات لجميع مباريات ODI التي لعبها Virat Kohli من 18 أغسطس 2008 إلى 22 يناير 2017. أنت مطالب بتحليل أداء Virat Kohliفي مباريات ODI.

فيما يلي المعلومات الكاملة حول جميع الأعمدةفي مجموعة البيانات:

1. **Runs**: النقاطفي المباراة.
2. **BF**: الكرات التي تواجههافي المباراة.
3. **4s**: عدد 4sفي المباراة.
4. **6s**: عدد 6sفي المباراة.
5. **SR**: معدل الضرباتفي المباراة.
6. **Pos**: مركز الضربفي المباراة.
7. **Dismissal**: كيف خرج Virat Kohliفي المباراة.
8. **Inns**: الشوط الأول والثاني.
9. **Opposition**: من كان خصم الهند.
10. **Ground**: مكان المباراة.
11. **Start Date**: تاريخ المباراة.

يمكنك تنزيل مجموعة البيانات هذه من هنا.

تحليل أداء Virat Kohli باستخدام بايثون

لنبدأ الآن بمهمة تحليل أداء Virat Kohli باستخدام بايثون. سأبدأ هذه المهمة عن طريق استيراد مكتبات بايثون ومجموعة البيانات اللازمة:

```
import pandas as pd
import numpy as np
import plotly.express as px
import plotly.graph_objects as go

data = pd.read_csv("Virat_Kohli.csv")
print(data.head())
```

```
   Runs  BF  4s  6s     SR  Pos Dismissal  Inns   Opposition         Ground  \
0    12  22   1   0  54.54  2.0       lbw     1  v Sri Lanka       Dambulla
1    37  67   6   0  55.22  2.0    caught     2  v Sri Lanka       Dambulla
2    25  38   4   0  65.78  1.0   run out     1  v Sri Lanka  Colombo (RPS)
3    54  66   7   0  81.81  1.0    bowled     1  v Sri Lanka  Colombo (RPS)
4    31  46   3   1  67.39  1.0       lbw     2  v Sri Lanka  Colombo (RPS)

  Start Date
0  18-Aug-08
1  20-Aug-08
2  24-Aug-08
3  27-Aug-08
4  29-Aug-08
```

دعنا نلقي نظرة على ما إذا كانت مجموعة البيانات هذه تحتوي على أي قيم فارغة أم لا قبل المضي قدمًا:

```
print(data.isnull().sum())
```

```
Runs          0
BF            0
4s            0
6s            0
SR            0
Pos           0
Dismissal     0
Inns          0
Opposition    0
Ground        0
Start Date    0
dtype: int64
```

تحتوي مجموعة البيانات على المباريات التي لعبها Virat Kohli بين 18 أغسطس 2008 و 22 يناير 2017. لذلك دعونا نلقي نظرة على إجمالي النقاط التي سجلها Virat Kohli:

```
#Total Runs Between 18-Aug-08 - 22-Jan-17
data["Runs"].sum()
```

```
6184
```

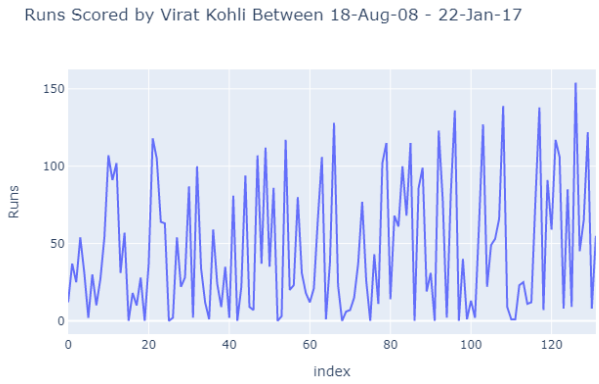
الآن دعونا نلقي نظرة على متوسط نقاط Virat Kohli خلال نفس الفترة:

```
#Average Runs Between 18-Aug-08 - 22-Jan-17
data["Runs"].mean()
```

```
46.84848484848485
```

في ODIs، يعتبر متوسط الضرب من 35 ـ 37 متوسطًا جيدًا. لذا فإن معدل الضرب في Virat Kohli جيد. الآن دعونا نلقي نظرة على اتجاه الجري الذي سجله Virat Kohli في مسيرته من 18 أغسطس 2008 إلى 22 يناير 2017:

```
matches = data.index
figure = px.line(data, x=matches, y="Runs”,
       title='Runs Scored by Virat Kohli Between 18-Aug-08 - 22-Jan-17’)
figure.show()
```



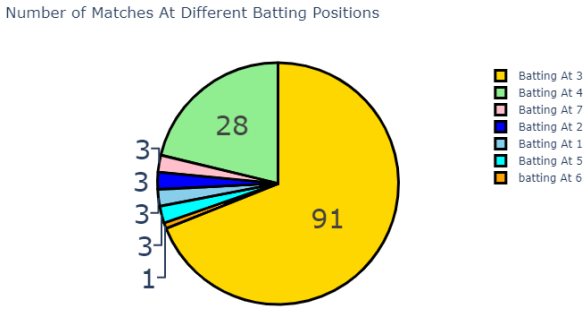
في العديد من الاشواط التي لعبها Virat Kohli ، سجل أكثر من 100 نقطة أو ما يقرب من ذلك. هذه علامة جيدة على الثبات. الآن دعونا نرى جميع مواقع الضرب (batting positions) التي لعبها Virat Kohli :

```
#Batting Positions
data["Pos"] = data["Pos"].map({3.0: "Batting At 3", 4.0: "Batting At 4",
2.0: "Batting At 2,"
":1.0             Batting At 1", 7.0:"Batting At 7", 5.0:"Batting At 5,"
":6.0             batting At 6”})

Pos = data["Pos"].value_counts()
label = Pos.index
counts = Pos.values
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
```

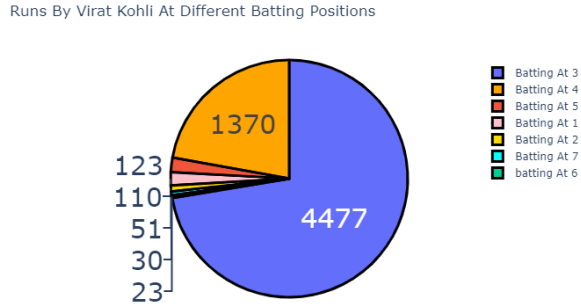
```
fig.update_layout(title_text='Number of Matches At Different Batting
Positions')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
textfont_size=30,
       marker=dict(colors=colors, line=dict(color='black', width=3))(
fig.show()
```



في أكثر من 68٪ من جميع الاشواط التي لعبها Virat Kohli ، احتل المركز الثالث. الآن دعونا نلقي نظرة على مجموع النقاط (total runs) التي سجلها Virat Kohliفي مراكز مختلفة:

```
label = data["Pos"]
counts = data["Runs"]
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

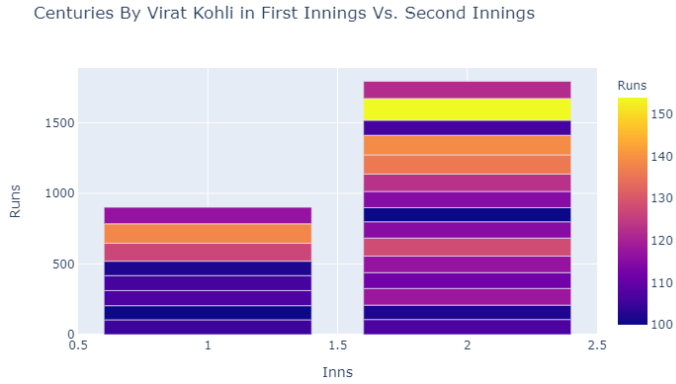
fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Runs By Virat Kohli At Different Batting
Positions')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
textfont_size=30,
       marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```



أكثر من 72٪ من مجموع النقاط التي سجلها Virat Kohli هيفي المركز الثالث. لذلك يمكننا القول أن الضربفي المركز الثالث مثالي لـVirat Kohli .

الآن دعونا نلقي نظرة على عدد القرون centuries (100 نقطةفي شوط واحد) التي سجلها Virat Kohli أثناء الضربفي الاشواط الأولى والثانية:

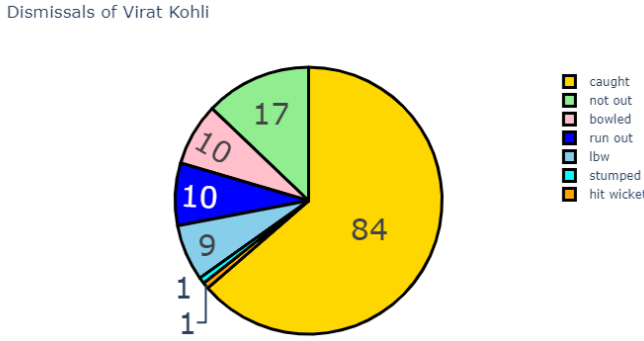
```
centuries = data.query("Runs >= 100")
figure = px.bar(centuries, x=centuries["Inns"], y = centuries["Runs”],
       color = centuries["Runs"],
       title="Centuries By Virat Kohli in First Innings Vs. Second Innings”)
figure.show()
```



لذلك يتم تسجيل معظم القرون أثناء الضربفي الأدوار الثانية. من خلال هذا، يمكننا القول إن Virat Kohli يحب مطاردة النتائج. دعنا الآن نلقي نظرة على نوع عمليات الطرد (dismissals) التي واجهها Virat Kohliفي معظم الأوقات:

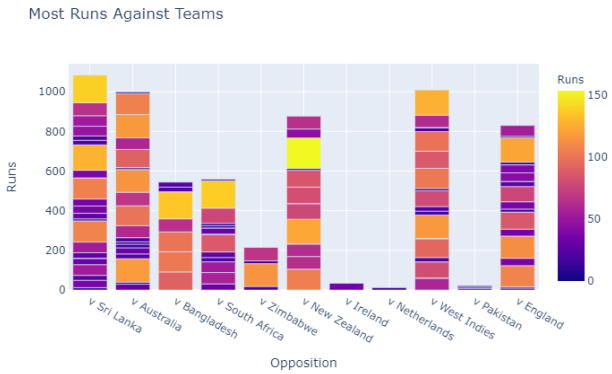
```
#Dismissals of Virat Kohli
dismissal = data["Dismissal"].value_counts()
label = dismissal.index
counts = dismissal.values
colors = ['gold','lightgreen', "pink", "blue", "skyblue", "cyan", "orange"]

fig = go.Figure(data=[go.Pie(labels=label, values=counts)])
fig.update_layout(title_text='Dismissals of Virat Kohli')
fig.update_traces(hoverinfo='label+percent', textinfo='value',
textfont_size=30,
       marker=dict(colors=colors, line=dict(color='black', width=3)))
fig.show()
```



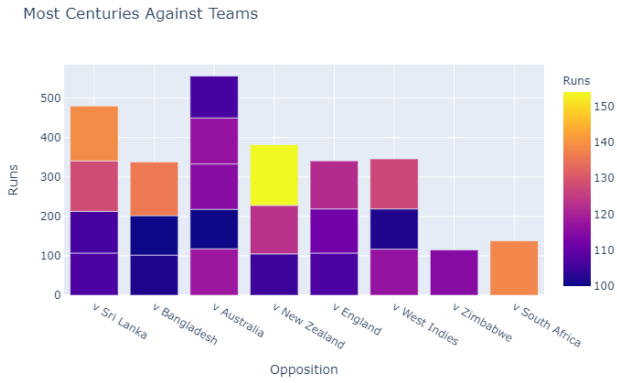
لذلكفي معظم الأوقات، يخرج Virat Kohli من خلال القبض عليه من قبل اللاعب أو الحارس. الآن دعونا نلقي نظرة على الفريق الذي سجل Virat Kohli معظم نقاطه:

```
figure = px.bar(data, x=data["Opposition"], y = data["Runs"], color =
data["Runs”],
     title="Most Runs Against Teams”)
figure.show()
```



وفقًا للرقم أعلاه، يحب Virat Kohli اللعب ضد سريلانكا وأستراليا ونيوزيلندا وجزر الهند الغربية وإنجلترا. لكنه سجل معظم أشواطه أثناء لعبه ضد سريلانكا. دعونا الآن نلقي نظرة على الفريق الذي سجل ضده Virat Kohli معظم قرونه:

```
figure = px.bar(centuries, x=centuries["Opposition"], y =
centuries["Runs”],
       color = centuries["Runs"],
       title="Most Centuries Against Teams”)
figure.show()
```



لذلك، كانت معظم القرون التي سجلها Virat Kohli ضد أستراليا. الآن دعونا نحلل معدل ضربات Virat Kohli. لتحليل معدل ضربات Virat Kohli ، سأقوم بإنشاء مجموعة بيانات جديدة لجميع المباريات التي لعبها Virat Kohli حيث كان معدل تسديده أكثر من 120:

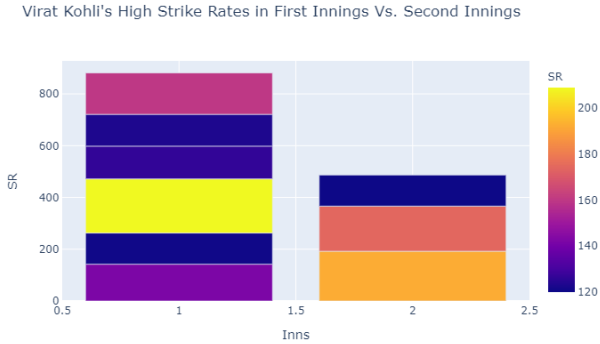
```
strike_rate = data.query("SR >= 120")
print(strike_rate)
```

```
     Runs  BF  4s  6s      SR           Pos Dismissal  Inns     Opposition  \
8      27  19   4   0  142.10  Batting At 7    bowled     1    v Sri Lanka
32    100  83   8   2  120.48  Batting At 4   not out     1   v Bangladesh
56     23  11   3   0  209.09  batting At 6   not out     1  v West Indies
76     43  34   4   1  126.47  Batting At 3    caught     1      v England
78    102  83  13   2  122.89  Batting At 3    caught     1  v West Indies
83    100  52   8   7  192.30  Batting At 3   not out     2    v Australia
85    115  66  18   1  174.24  Batting At 3   not out     2    v Australia
93     78  65   7   2  120.00  Batting At 3    caught     2  v New Zealand
130     8   5   2   0  160.00  Batting At 3    caught     1      v England

            Ground Start Date
8           Rajkot  15-Dec-09
32           Dhaka  19-Feb-11
56          Indore   8-Dec-11
76      Birmingham  23-Jun-13
78   Port of Spain   5-Jul-13
83          Jaipur  16-Oct-13
85          Nagpur  30-Oct-13
93        Hamilton  22-Jan-14
130        Cuttack  19-Jan-17
```

الآن دعونا نرى ما إذا كان Virat Kohli يلعب بمعدلات ضربات عاليةفي الاشواط الأولى أو الاشواط الثانية:

```
figure = px.bar(strike_rate, x = strike_rate["Inns”],
       y = strike_rate["SR"],
       color = strike_rate["SR"],
     title="Virat Kohli's High Strike Rates in First Innings Vs. Second
Innings”)
figure.show()
```

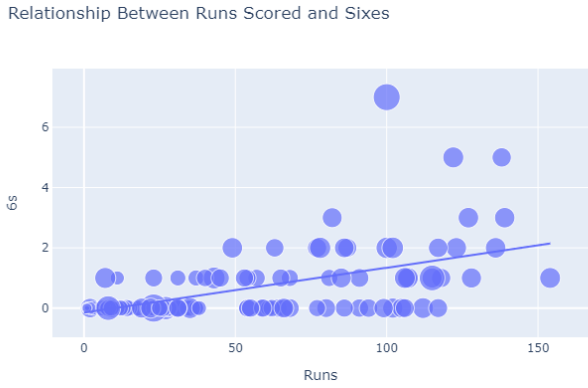


لذلك وفقًا للشكل أعلاه، يحب Virat Kohli اللعب بقوة أكبرفي الاشواط الأولى مقارنة بالاشواط الثانية. الآن دعونا نرى العلاقة بين الأشواط التي سجلها Virat Kohli والأربع (fours) التي لعبهافي كل شوط:

```
figure = px.scatter(data_frame = data, x="Runs,"
        y="4s", size="SR", trendline="ols,"
        title="Relationship Between Runs Scored and Fours”)
figure.show()
```

هناك علاقة خطية. هذا يعني أن Virat Kohli يحب اللعب الرباعي. كلما زاد عدد مرات الجري التي سجلهافي الأدوار، كلما لعب رباعيات أكثر. دعونا نرى ما إذا كانت هناك علاقة ما مع السداسيات (sixes):

```
figure = px.scatter(data_frame = data, x="Runs,"
        y="6s", size="SR", trendline="ols,"
        title= "Relationship Between Runs Scored and Sixes”)
figure.show()
```



لا توجد علاقة خطية قوية هنا. هذا يعني أن Virat Kohli يحب لعب رباعايات أكثر من سداسيات. هذه هي الطريقة التي يمكنك بها تحليل أداء Virat Kohli أو أي لاعب كريكيت آخرفي العالم.

الملخص

هذه هي الطريقة التي يمكنك بها إجراء تحليل أداء Virat Kohli باستخدام لغة برمجة بايثون. يعد تحليل أداء اللاعب إحدى حالات استخدام علم البياناتفي التحليلات الرياضية. آمل أن تكون قد أحببت هذه المقالة حول تحليل أداء Virat Kohli باستخدام بايثون.

المصدر:

https://thecleverprogrammer.com/2022/05/10/virat-kohli-performance-/analysis-using-python

20) الوقاية من الإصابات في لعبة البيسبول باستخدام التعلم الآلي injury prevention in baseball using machine learning

الإصابات Injuries هي جزء مؤسف ولكنه لا مفر منهفي أي رياضة احترافية، والبيسبول baseball ليس استثناءً. من تمزقات الرباط الجانبي الزندي readed ulnar collateral ligament (UCL) المروعة التي يمكن أن تؤدي إلى جراحة تومي جون Tommy John surgery للرامين pitchers، إلى إجهادات أوتار الركبة hamstring strains وإصابات الأنسجة الرخوة soft tissue injuries الأخرى التي يمكن أن تهمل اللاعبين، يمكن أن تؤثر المتطلبات البدنية للبيسبول سلبًا على حتى الرياضيين الأكثر لياقة. مع استثمار الفرق لملايين الدولاراتفي لاعبيها، فإن الحفاظ على صحتهم وضمان بقائهمفي الملعب له أهمية قصوى.

في السنوات الأخيرة، سمحت التطوراتفي تحليلات البيانات للفرق بتطوير استراتيجيات متطورة للوقاية من الإصابات لتحسين أداء اللاعبين وإطالة حياتهم المهنية. على سبيل المثال، كان فريق لوس أنجلوس دودجرز Los Angeles Dodgers استباقيًافي استخدام البيانات للوقاية من الإصابات. كانوا أول فريق رياضي أمريكي يشترك مع Kitman Labs، وهي شركة تكنولوجيا رياضية أيرلندية معروفة بتزويد الفرق الأوروبية ببيانات إدارة الرياضيين والوقاية من الإصابات.

من خلال العمل معًا، طور فريق دودجرز وKitman Labs برنامجًا شاملاً مصممًا خصيصًا للاعبي البيسبول. يستخدم هذا البرنامج القياسات الحيوية جنبًا إلى جنب مع مقاييس عبء العمل الأخرى لتحديد اللاعبين الذين قد يكونون معرضين لخطر الإصابة.في الوقت الفعلي، تلتقط كاميرا محمولة عالية الدقة أنماط الحركة، والتي يتم تحليلها بعد ذلك لتوفير رؤى قابلة للتنفيذ.

الهدف من النظام هو تنبيه الفريق إلى المشكلات المحتملة قبل حدوثها، مما يوفر فرصة لمنع الإصابات قبل أن تُبعد اللاعبين عن الملاعب. يدمج هذا النهج القائم على البيانات المعلومات الطبية وبيانات القوة واللياقة البدنية ومقاييس الأداءفي نظام شامل واحد، بهدف تعزيز فريق أكثر صحة وفعالية.

يستكشف هذهالمقالة كيف يمكن لمنظمات البيسبول الاستفادة من الأساليب القائمة على البيانات للتخفيف من مخاطر الإصابة وتعظيم عمر اللاعب، والتعمقفي الأساليب المستخدمة لتحديد عوامل خطر الإصابة، وتطوير برامج التدريب والتعافي الشخصية، وأهمية إدارة الحمل والمراقبة أثناء الموسم لمنع الإصابات أثناء موسم البيسبول الطويل والشاق.

جمع البيانات الحيوية من الأحداث الرياضية الحية

إن جمع البيانات البيومترية biometric data هو جانب بالغ الأهمية للوقاية من الإصابات المعتمدة على البياناتفي الرياضة، بمافي ذلك لعبة البيسبول. تشير البيانات البيومترية إلى المعلومات المتعلقة بالخصائص الجسدية والسلوكية للشخص.في الرياضة، قد يشمل ذلك معدل ضربات قلب اللاعب وأنماط حركته وغير ذلك من المقاييس الفسيولوجية.

لقد تحسنت التقنيات المستخدمة لالتقاط بيانات اللاعبين بسرعة على مدار السنوات العديدة الماضية، مما يسمح لأطباء الفريق والمدربين ببناء ملف صحي للاعبيهم بسهولة أكبر. وهذا ممكن حتىفي اللعب المباشر، وهو أمر لا يمكن تصورهفي الماضي القريب.

هناك العديد من الطرق لجمع بيانات اللاعبين، ولكن إليك أربع طرق تدعم حالة استخدام علم البيانات data science للحد من مخاطر الإصابة:

1. **التكنولوجيا القابلة للارتداء Wearable technology:** تعد التكنولوجيا القابلة للارتداء واحدة من أكثر الطرق شيوعًا لجمع البيانات البيومترية من الرياضيين. تشمل هذه الفئة أجهزة مثل أجهزة تتبع اللياقة البدنية والساعات الذكية وأجهزة تحديد المواقع العالمية وأجهزة مراقبة معدل ضربات القلب، بالإضافة إلى معدات أكثر تخصصًا مثل الملابس الذكية وأجهزة الاستشعار المضمنةفي المعدات الرياضية. يمكن لهذه الأجهزة جمع مجموعة واسعة من البيانات، مثل معدل ضربات القلب heart rate وضغط الدم blood pressure ودرجة حرارة الجسم body temperature ومستويات الأكسجين oxygen levels ومعدل التعرق sweat rate وحتى أنماط النوم sleep patterns.
   على سبيل المثال، يستخدم العديد من الرياضيين أجهزة مراقبة معدل ضربات القلب أثناء التدريب لتتبع حمولتهم القلبية الوعائية cardiovascular load والتعافي. تُستخدم أجهزة تحديد المواقع العالمية GPS بشكل شائعفي الرياضات الجماعية مثل كرة القدم أو الرجبي لتتبع حركات اللاعبين وتحديد حجم العمل.في الرياضات مثل كرة السلة، يمكن لأجهزة الاستشعار المضمنةفي الكرة أو أحذية اللاعبين تتبع أنماط الحركة التفصيلية ودقة الملعب ومقاييس الأداء الأخرى.
2. **الفحص البيومتري Biometric screening:** يشير هذا إلى الاختبارات الطبية التي تقيس الخصائص الفيزيائية مثل مؤشر كتلة الجسم body mass index (BMI) ونسبة الدهون في الجسم body fat percentage وكتلة العضلات muscle mass وكثافة العظام bone density وسعة الرئة lung capacity. تُجرى هذه الفحوصات عادةًفي بيئة سريرية ويمكن أن توفر بيانات أساسية مهمة لتتبع صحة ولياقة الرياضي بمرور الوقت.

3. **تكنولوجيا التصوير Imaging technology**: يمكن استخدام تقنيات مثل التصوير بالرنين المغناطيسي MRI والتصوير المقطعي المحوسب CT scans والموجات فوق الصوتية ultrasound لجمع البيانات الحيوية عن الهياكل الداخلية للرياضي مثل العظام والعضلات والأعضاء. يمكن أن يكون هذا مفيدًا بشكل خاص للوقاية من الإصابات والتعافي منها.
4. **الاختبارات الجينية Genetic testing**: يخضع بعض الرياضيين للاختبارات الجينية لاكتساب رؤى حول إمكاناتهمفي الأداء وقابليتهم للإصابة. يمكن أن يساعد هذافي توجيه خطط التدريب والتغذية الشخصية.

من المهم أن نضعفي الاعتبار أنهفي حين أن جمع البيانات الحيوية يمكن أن يوفر رؤى قيمة، فإنه يثير أيضًا قضايا مهمة تتعلق بالخصوصية وأمان البيانات. يجب على المنظمات الرياضية التأكد من أنها تتعامل مع هذه البيانات الحساسة وتخزنها بشكل مسؤول، بما يتوافق مع القوانين واللوائح ذات الصلة.

من المهم أيضًا ملاحظة أن استخدام البيانات الحيويةفي الرياضة هو مجال سريع التطور، مع تطوير تقنيات وطرق جديدة طوال الوقت. وعلى هذا النحو، فإن تفاصيل كيفية جمع البيانات البيومترية قد تختلف على نطاق واسع اعتمادًا على الرياضة ومستوى المنافسة والموارد المتاحة وعوامل أخرى.

كانت التكنولوجيا القابلة للارتداء Wearable technology موضوعًا ساخنًا للمناقشة بين دوري البيسبول الرئيسي MLB ورابطة لاعبي دوري البيسبول الرئيسي (MLBPA). إن الاهتمام بهذه التكنولوجيا مدفوع بالأهداف المشتركة لكل من المالكين واللاعبين: الصحة والأداء. ومع ذلك، قبل استخدام أي تقنية قابلة للارتداء، يجب أن تمر بعملية اختبار وتقييم مكثفة وفقًا لقواعد البيسبول الرسمية، والتي تتطلب الموافقة على أي تقنية جديدة قبل استخدامهافي الملعب. يمكنك قراءة المزيد حول هذا الموضوع هنا.

الاستفادة من البيانات البيومترية للوقاية من الإصابات

الآن بعد أن رأينا كيف يمكننا التقاط البيانات البيومترية للمساعدةفي منع الإصابات المحتملةفي لعبة البيسبول، فلنناقش كيف يمكن الاستفادة منها لمنع إصابات اللاعبين. توفر البيانات البيومترية رؤى تفصيلية حول الحالة البدنية والأداء الرياضي، وبالتالي فهي ضرورية لمساعدة الفرق على فهم أداء لاعبيها. من خلال مراقبة هذه المقاييس، يمكن للفرق اكتشاف التغييرات الدقيقة التي قد تشير إلى زيادة خطر الإصابة. على سبيل المثال، قد تشير التغييراتفي معدل ضربات قلب الرامي pitcher's heart rate أو أنماط حركته إلى التعب أو الإجهاد الذي قد يؤدي إلى الإصابة إذا لم تتم إدارته بشكل صحيح.

قد تبدو النسخة المبسطة من مجموعة البيانات هذه على هذا النحو:

```
import pandas as pd

# Hypothetical player data with a focus on biometric metrics
data = {
    'Player': ['Player A', 'Player B', 'Player C', 'Player D'],
    'Heart Rate': [80, 85, 90, 78],
    'Movement Pattern Risk Score': [5, 3, 7, 4],
    'Resting Metabolic Rate': [1500, 1600, 1550, 1520],
    'Sleep Hours': [8, 7, 6, 7.5],
    'Stress Level': [3, 4, 5, 2],  # on a scale of 1-10
}

df = pd.DataFrame(data)
```

في هذه المجموعة من البيانات، قمنا بتضمين بعض المقاييس الحيوية الإضافية التي قد تكون ذات صلة بمخاطر الإصابة، مثل معدل الأيض أثناء الراحة resting metabolic rate، وساعات النوم hours of sleep، ومستوى التوتر stress level. تمنحنا هذه البيانات صورة أكثر اكتمالاً للحالة البدنية لكل لاعب وتسمح لنا بتقييم مخاطر الإصابة بشكل أفضل.

على سبيل المثال، قد يُعتبر اللاعب الذي يعاني من معدل ضربات قلب مرتفع، ومعدل مخاطر نمط الحركة المرتفع، ومعدل الأيض أثناء الراحة المنخفض، وساعات نوم منخفضة، ومستوى توتر مرتفع، معرضًا لخطر الإصابة بشكل أكبر. يمكن أن يساعد هذا النوع من التحليل القائم على البيانات الفرقفي منع الإصابات من خلال تحديد اللاعبين المعرضين للخطرفي وقت مبكر واتخاذ الإجراءات المناسبة، مثل تعديل جداول التدريب، أو تنفيذ استراتيجيات التعافي المستهدفة، أو إجراء تغييرات على عوامل التغذية ونمط الحياة.

قد يستخدم الفريق مجموعة البيانات هذه لتحديد اللاعبين المعرضين لخطر الإصابة. على سبيل المثال، قد يعتبر الفريق المذكور اللاعبين الذين ألقوا عددًا كبيرًا من الكرات، ولديهم متوسط سرعة رمي مرتفع high average pitch speed ، ومعدل ضربات قلب مرتفع، ومعدل مخاطر نمط الحركة مرتفعًا معرضين لخطر إصابة بشكل أكبر.

قد يستخدمون نظام تسجيل مخاطر risk scoring system بسيط مثل ما يلي:

```
# Define a function to calculate risk score
def calculate_risk_score(row):
    pitch_score = row['Pitches Thrown'] / 100
    speed_score = row['Average Pitch Speed'] / 100
    heart_rate_score = row['Heart Rate'] / 100
    movement_score = row['Movement Pattern Risk Score'] / 10

    return pitch_score + speed_score + heart_rate_score + movement_score

# Calculate risk score for each player
df['Risk Score'] = df.apply(calculate_risk_score, axis=1)
```

سيؤدي هذا إلى إنشاء إطار بيانات dataframe جديد بدرجة مخاطرة محسوبة لكل لاعب:

```
print(df)

# Output:
#     Player  Pitches Thrown  Average Pitch Speed  Heart Rate  Movement Pattern Risk Score  Risk Score
# 0  Player A             100                   90          80                         5        2.75
# 1  Player B              80                   85          85                         3        2.53
# 2  Player C             120                   95          90                         7        3.12
# 3  Player D              90                   88          78                         4        2.50
```

في هذا السيناريو الافتراضي، سيُعتبر اللاعب C الأكثر عرضة للخطر، نظرًا لعدد الكرات المرتفعة التي سددها، ومتوسط سرعة الكرة المرتفع، ومعدل ضربات القلب المرتفع، ودرجة المخاطرة المرتفعة لنمط الحركة.

هذا مثال مبسط ومن المرجح أن تكون العمليات الفعلية التي يستخدمها فريق دودجرز أكثر تعقيدًا، وتتضمن المزيد من المتغيرات وخوارزميات التعلم الآلي المتطورة. ومع ذلك، فإنه يوضح النهج العام لاستخدام البيانات لتحديد عوامل الخطر والتنبؤ باحتمالية الإصابات.

يعتمد الاستخدام الفعال للبيانات الحيويةفي الوقاية من الإصابات على وجود بيانات دقيقة وفي الوقت المناسب، والقدرة على تحليل هذه البيانات وتفسيرها، والقدرة على ترجمة هذه الأفكار إلى تدخلات قابلة للتنفيذ. وبهذه الطريقة، يمكن لعلم البيانات أن يدعم ليس فقط الوقاية من الإصابات ولكن أيضًا تحسين الأداءفي لعبة البيسبول وغيرها من الرياضات.

إحداث ثورة في تدريب وتطوير اللاعبين

إن عالم البيسبول، كما هو الحالفي الرياضات الأخرى، أصبح مدفوعًا بالبيانات بشكل متزايد. تستغل الفرق قوة البيانات والتكنولوجيا ليس فقط لاتخاذ القرارات الاستراتيجية أثناء المباريات، ولكن أيضًا لإحداث ثورةفي تدريب اللاعبين وتطويرهم. دعونا نتعمقفي كيفية حدوث ذلكفي عالم دوري البيسبول الرئيسي.

التدريب بالواقع الافتراضي

يتم الاستفادة من تقنية الواقع الافتراضي Virtual Reality (VR)في مجموعة متنوعة من الرياضات لأغراض التدريب، والبيسبول ليس استثناءً. يسمح الواقع الافتراضي للاعبين بمحاكاة مواقف اللعبة الحقيقية دون الإجهاد البدني للتواجدفي الملعب. على سبيل المثال، يمكن للضاربين مواجهة رماة افتراضيين يرمون مجموعة متنوعة من الكرات بسرعات مختلفة وفي مواقع مختلفة، مما يساعدهم على تحسين مهارات التعرف على الكرات وأوقات رد الفعل.

قد يبدو الكود لمحاكاة تدريب الواقع الافتراضي شيئًا مثل:

```
# Sample code for a VR training simulation
class VRTrainingSimulation:
    def __init__(self, player):
        self.player = player
```

```
    def simulate_pitch(self, pitch_type, pitch_speed, pitch_location):
        # Simulate a pitch and return the player's reaction time and
decision
        pass
```

العلوم الرياضية واللياقة البدنية

تلعب العلوم الرياضية Sports science وبرامج اللياقة البدنية المعتمدة على البيانات data-driven conditioning programs دورًا متزايد الأهميةفي الحفاظ على صحة اللاعبين ووصولهم إلى قمة أدائهم. وكما ناقشنا سابقًا، يمكن للتقنيات القابلة للارتداء مراقبة معدل ضربات قلب اللاعبين وأنماط نومهم وتعافيهم، مما يوفر بيانات قيمة يمكن استخدامها لتحسين برامج اللياقة البدنية conditioning programs وروتين التعافي recovery routines.

قد يبدو الكود الخاص بمثال العلوم واللياقة البدنية على هذا النحو:

```
# Sample code for monitoring player's heart rate
class HeartRateMonitor:
    def __init__(self, player):
        self.player = player

    def monitor_heart_rate(self):
        # Monitor the player's heart rate and return a report
        pass
```

اتخاذ القرارات بناءً على البيانات

لا تُستخدم البيانات التي يتم جمعها من هذه التقنيات لتحسين أداء اللاعب الفردي فحسب، بل تُستخدم أيضًا لإعلام عملية اتخاذ القرارات الاستراتيجية. يمكن للمدربين وإدارة الفريق استخدام الرؤى المستمدة من البيانات لاتخاذ قرارات مستنيرة حول برامج تدريب اللاعبين واستراتيجيات اللعب وجداول الراحة والتعافي.

قد يبدو الكود المستخدمفي عملية اتخاذ القرار هذه على هذا النحو:

```
# Sample code for decision making based on player data
class DecisionMaker:
    def __init__(self, player_data):
        self.player_data = player_data

    def make_decision(self):
        # Make a decision based on player data and return the decision
        pass
```

حالات الاستخدام

لفهم تأثير التكنولوجيا على تدريب اللاعبين وتطويرهم حقًا، دعنا نستكشف بعض حالات الاستخدام في العالم الحقيقي.

تصميم الملعب

يعد تصميم الملعب Pitch design مثالاً رائعًا للمكان الذي تحدث فيه التكنولوجيا تأثيرًا كبيرًا. تسمح الكاميرات عالية السرعة وتقنية تتبع الملعب مثل Rapsodo وTrackMan للفرق بتحليل معدل الدوران ومحور الدوران وحركة كل ملعب. يمكن بعد ذلك استخدام هذه المعلومات لمساعدة الرماة على تحسين ملعبهم أو حتى تطوير ملعب جديد.

على سبيل المثال، قد يكون لدى الرامي كرة منحنية لا تحتوي على نفس القدر من الانخفاض الرأسي مثل الكرات المنحنية النموذجية. باستخدام تقنية تتبع الملعب pitch tracking ، يمكنهم رؤية معدل الدوران الدقيق ومحور الكرة المنحنية ومقارنتها بالرماة الآخرين. ثم باستخدام هذه المعلومات، يمكنهم تعديل قبضتهم أو زاوية ذراعهم لمحاولة زيادة الانخفاض الرأسي للكرة المنحنية.

دعنا نوضح كيف يمكن التعامل مع تصميم الملعب باستخدام التكنولوجيا من خلال مثال بايثون. بالنسبة لهذا الرسم التوضيحي، سأفترض أن لدينا مجموعة من بيانات الملعب للاعب معين، وسنقوم بتحليل هذه البيانات لمساعدتهم على تحسين الكرة المنحنية الخاصة بهم.

```
import pandas as pd
import matplotlib.pyplot as plt

# Assuming we have a csv file 'pitch_data.csv' with columns: 'pitch_type',
'spin_rate', 'spin_axis', 'vertical_drop'
pitch_data = pd.read_csv('pitch_data.csv')

# Filter out the data for curveballs
curveballs = pitch_data[pitch_data['pitch_type'] == 'curveball']

# Calculate average spin rate, spin axis and vertical drop
avg_spin_rate = curveballs['spin_rate'].mean()
avg_spin_axis = curveballs['spin_axis'].mean()
avg_vertical_drop = curveballs['vertical_drop'].mean()

print(f"Average Spin Rate: {avg_spin_rate}")
print(f"Average Spin Axis: {avg_spin_axis}")
print(f"Average Vertical Drop: {avg_vertical_drop}")

# Compare this pitcher's average spin rate and vertical drop with typical
values
typical_spin_rate = 2500  # This is just an illustrative value
typical_vertical_drop = -15  # This is just an illustrative value

if avg_spin_rate < typical_spin_rate:
    print("The spin rate of the curveball is less than typical. Consider
working on your grip to increase spin rate.")
if avg_vertical_drop > typical_vertical_drop:
    print("The curveball doesn't have as much vertical drop as typical.
Consider adjusting your release point to increase vertical drop.")

# Visualize the spin rate and vertical drop
plt.scatter(curveballs['spin_rate'], curveballs['vertical_drop'])
plt.xlabel('Spin Rate')
plt.ylabel('Vertical Drop')
```

```
plt.title('Curveball Spin Rate vs Vertical Drop')
plt.show()
```

في هذا المثال، نقوم أولاً بتحميل بيانات الملعب من ملف CSV. ثم نقوم بتصفية الكرات المنحنية وحساب متوسط معدل الدوران average spin rate ومحور الدوران pin axis والهبوط الرأسي vertical drop لهذه الكرات pitches. ثم نقارن هذه المتوسطات بالقيم النموذجية للكرة المنحنية. بناءً على كيفية مقارنة متوسطات الرامي بالقيم النموذجية، فقد نقدم له نصائح حول كيفية ضبط قبضته أو نقطة الإطلاق لتحسين الكرة المنحنية.

أخيرًا، نرسم معدل الدوران مقابل الهبوط الرأسي لكل كرة منحنية. يمكن أن يساعدنا هذا في تصور أي اتجاهات أو أنماط في البيانات.

الوقاية من الإصابات

كما ذكرنا سابقًا، تعد الوقاية من الإصابات injury prevention أحد أهم تطبيقات التكنولوجيا في الرياضة. تُستخدم التكنولوجيا القابلة للارتداء، مثل WHOOP أو Zephyr Bioharness، لتتبع معدل ضربات قلب اللاعبين وجودة نومهم وغيرها من البيانات البيومترية. يمكن أن تساعد هذه البيانات الفرق في تحديد علامات التعب أو الإجهاد التي قد لا تكون ملحوظة بالعين المجردة.

في حالة الرماة pitchers، على سبيل المثال، قد يكون التغيير البسيط في آليات حركة الرمي (والتي يمكن اكتشافها باستخدام تقنية التقاط الحركة motion capture technology) علامة على التعب أو إصابة طفيفة، والتي قد تؤدي، إذا لم يتم معالجتها، إلى إصابة أكثر خطورة.

لنفكر في سيناريو حيث لدينا بيانات بيومترية تم جمعها من أجهزة يمكن ارتداؤها على الرماة، ونتطلع إلى تحديد علامات التعب أو خطر الإصابة المحتملة. لهذا، سنستخدم بايثون جنبًا إلى جنب مع بعض مكتبات علم البيانات مثل pandas وscikit-learn.

```
import pandas as pd
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split
from sklearn.metrics import classification_report

# Assuming we have a CSV file 'biometric_data.csv' with columns:
'heart_rate', 'sleep_quality', 'pitch_speed', 'pitch_count', 'fatigue'
biometric_data = pd.read_csv('biometric_data.csv')

# Let's say fatigue is our target variable and it's binary - 'Yes' or 'No'
X = biometric_data[['heart_rate', 'sleep_quality', 'pitch_speed',
'pitch_count']]
y = biometric_data['fatigue']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a Random Forest classifier and fit it to our training data
clf = RandomForestClassifier(random_state=42)
```

```
clf.fit(X_train, y_train)

# Predict fatigue on the test set
y_pred = clf.predict(X_test)

# Print a classification report to evaluate our model
print(classification_report(y_test, y_pred))

# Now, if we have a new pitcher's biometric and pitch data, we can predict
if they're likely to be fatigued
new_pitcher_data = pd.DataFrame({
    'heart_rate': [75],
    'sleep_quality': [0.85],
    'pitch_speed': [92],
    'pitch_count': [100]
})

fatigue_prediction = clf.predict(new_pitcher_data)
print("Fatigue Prediction for New Pitcher: ", fatigue_prediction)
```

في هذا المثال، نستخدم مصنف الغابة العشوائية Random Forest Classifier للتنبؤ بالتعب استنادًا إلى معدل ضربات القلب heart rate وجودة النوم sleep quality وسرعة الملعب pitch speed وعدد الملعبات pitch count. نقوم بتدريب النموذج على البيانات التاريخية ثم نستخدمها للتنبؤ بما إذا كان من المحتمل أن يشعر رامٍ جديد بالتعب استنادًا إلى بياناته البيومترية وبيانات الملعب.

هذا مثال مبسط، ولكنفي سيناريو واقعي، قد نستخدم نماذج أكثر تعقيدًا أو ندرج ميزات إضافيةفي نموذجنا. من المهم أيضًا تقييم نموذجنا وتحديثه بانتظام مع توفر بيانات جديدة.

تطوير اللاعبين

يعد تطوير اللاعبين Player development مجالاً آخر حيث يكون للتكنولوجيا تأثير كبير. على سبيل المثال، تسمح تقنية HitTrax للضاربين برؤية البياناتفي الوقت الفعلي عن تأرجحتهم، مثل سرعة الخروج exit velocity وزاوية الإطلاق launch angle ونقطة الاتصال point of contact. يمكن استخدام هذه التقنيةفي التدريب لمساعدة الضاربين على فهم تأرجحتهم بشكل أفضل وإجراء التعديلات لتحسين أدائهم.

بالإضافة إلى ذلك، يتم استخدام تقنية الواقع الافتراضي لمساعدة الضاربين على تدريب مهارات التعرف على الملعب. باستخدام الواقع الافتراضي، يمكن للضاربين مواجهة مئات الملعباتفي فترة زمنية قصيرة، مما يساعدهم على التعرف على أنواع الملعب المختلفة والمواقع بشكل أسرع.

لنفكرفي موقف حيث نريد التنبؤ بأداء اللاعب بناءً على بياناته البيومترية ومقاييس الأداء السابقة. لغرض هذا المثال، سنستخدم بايثون وpandas للتلاعب بالبيانات وscikit-learn للتعلم الآلي.

```
import pandas as pd
from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error
```

```
# Assuming we have a CSV file 'player_data.csv' with columns: 'heart_rate',
'sleep_quality', 'batting_average', 'on_base_percentage',
'slugging_percentage', 'player_performance'
player_data = pd.read_csv('player_data.csv')

# Let's say player_performance is our target variable and it's continuous
X = player_data[['heart_rate', 'sleep_quality', 'batting_average',
'on_base_percentage', 'slugging_percentage']]
y = player_data['player_performance']

# Split the data into training and test sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2,
random_state=42)

# Create a Linear Regression model and fit it to our training data
model = LinearRegression()
model.fit(X_train, y_train)

# Predict player performance on the test set
y_pred = model.predict(X_test)

# Print the root mean squared error of our predictions to evaluate our
model
rmse = mean_squared_error(y_test, y_pred, squared=False)
print(f"Root Mean Squared Error: {rmse}")

# Now, if we have a new player's biometric and performance data, we can
predict their performance
new_player_data = pd.DataFrame({
    'heart_rate': [70],
    'sleep_quality': [0.90],
    'batting_average': [0.300],
    'on_base_percentage': [0.400],
    'slugging_percentage': [0.500]
})

performance_prediction = model.predict(new_player_data)
print("Performance Prediction for New Player: ", performance_prediction)
```

في هذا المثال، نستخدم نموذج الانحدار الخطي linear regression model للتنبؤ بأداء اللاعب بناءً على معدل ضربات القلب وجودة النوم ومتوسط الضرب ونسبة الوصول إلى القاعدة on-base percentage ونسبة الضربات القوية slugging percentage. نقوم بتدريب النموذج على البيانات التاريخية ثم نستخدمه للتنبؤ بالأداء المحتمل للاعب جديد بناءً على بياناته البيومترية وبيانات الأداء.

هذه مجرد أمثلة قليلة لكيفية إحداث التكنولوجيا ثورةفي تدريب اللاعبين وتطويرهمفي دوري البيسبول الرئيسي. ومع استمرار التكنولوجيافي التقدم، يمكننا أن نتوقع رؤية المزيد من الاستخدامات المبتكرة للتكنولوجيافي هذه الرياضة.

الخلاصة

في عصر حيث المعلومات هي القوة، تبنت لعبة البيسبول بوضوح ثورة البيانات. من استخدام البيانات البيومتريةفي الوقاية من الإصابات إلى تطبيق بيانات الأداء على أرض الملعبفي تحسين أداء اللاعبين، أصبح علم البيانات جزءًا لا يتجزأ من هذه الرياضة. تُظهر الأمثلة التي ناقشناها، مثل شراكة فريق لوس

أنجلوس دودجرز مع Kitman Labs واستخدام التكنولوجيا القابلة للارتداءفي دوري البيسبول الرئيسي، كيف تستخدم الفرق البيانات للحصول على ميزة تنافسية.

من خلال استخدام نماذج بايثون ووالتعلم الالي، أظهرنا كيف يمكن الاستفادة من البياناتفي الوقاية من الإصابات وتحسين التدريب والتنبؤ بأداء اللاعبين. ومع ذلك، فهذه ليست سوى أمثلة قليلة للتطبيقات العديدة لعلم البياناتفي لعبة البيسبول. إن إمكانات البياناتفي هذه الرياضة هائلة، ولا تقتصر إلا على الأسئلة التي نطرحها والرؤى التي نسعى إليها.

من المهم أن نتذكر أنهفي حين يمكن للبيانات والتحليلات أن توفر ميزة حاسمة، فإن مزيج الرؤى القائمة على البيانات والخبرة البشرية هو الذي يدفع حقًا حدود ما هو ممكن. يمكن للبيانات أن تساعدنافي تحديد الأنماط والتنبؤ، لكن قرارات الحكم والاستراتيجيات والشغف باللعبة لا يمكن الاستغناء عنها.

في نهاية المطاف، لا يغير ظهور علم البياناتفي لعبة البيسبول جوهر هذه الرياضة ـ بل يوفر فقط أدوات جديدة لفهمها وتعزيزها. ومع تطور أساليب جمع البيانات وزيادة قوة أدواتنا التحليلية، فمن المثير أن نتخيل كيف ستستمر هذه التطوراتفي تشكيل مستقبل لعبة البيسبول.

المصدر:

https://medium.com/data-science-at-microsoft/data-driven-injury-prevention-in-baseball-maximizing-player-performance-and-longevity-b9fade8992ed

21) تحليل بيانات لاعبي التنس Tennis player Data Analysis

سيتكون هذا المشروع من جميع المراحل السابقة: من معالجة البيانات إلى تحليل البيانات، إلى تصور البيانات، إلى الحصول على البيانات جاهزة تقريبًا لوقت النمذجة. لذا توقع أن يتحرك التحليل نحو اتجاه العثور على الميزات ذات الصلة لاستخدامهافي نموذج تنبؤي.

سأستخدم بعد ذلك، على انفراد، الأفكار التي سأكتشفها لإنشاء النموذج وآمل أن أجعله مربحًا بدرجة كافية.

الاستيراد والدوال

قد لا تكون هذه الممارسة الأكثر كفاءة، ولكنني أحب أن أحافظ على تنظيم الأشياء ولا أستطيع التفكير في طريقة أفضل للقيام بذلك.

فيما يلي، ستتمكن من رؤية الحزم التي أستخدمهافي المستند بالكامل، بالإضافة إلى بعض الدوال الخارجية التي ستساعدنيفي تسهيل هذا المسار.

```
################################################################################
############
# IMPORTS
################################################################################
############

# Math
import numpy as np
import math, statistics

# Data manipulation
import pandas as pd
from fuzzywuzzy import fuzz, process
!pip install xlrd
!pip install openpyxl

# Data visualization
import seaborn as sns
import matplotlib
import matplotlib.pyplot as plt
from matplotlib.pyplot import figure
plt.style.use('ggplot')
%matplotlib inline
matplotlib.rcParams['figure.figsize'] = (12,8)

# Machine learning
from sklearn.ensemble import RandomForestClassifier
from sklearn.impute import SimpleImputer

# Extras
from tqdm import tqdm
import time
```

```
# Shutting down warnings, just to make things cleaner
import warnings
warnings. simplefilter(action='ignore', category=Warning)
pd.set_option("display.max_columns", None)
pd.set_option("display.max_rows", 20)
```

```
Collecting xlrd
  Downloading xlrd-2.0.1-py2.py3-none-any.whl (96 kB)
     |████████████████████████████████| 96 kB 925 kB/s
Installing collected packages: xlrd
Successfully installed xlrd-2.0.1
WARNING: Running pip as root will break packages and permissions. You should install packa
ges reliably by using venv: https://pip.pypa.io/warnings/venv
Collecting openpyxl
  Downloading openpyxl-3.0.7-py2.py3-none-any.whl (243 kB)
     |████████████████████████████████| 243 kB 805 kB/s
Collecting et-xmlfile
  Downloading et_xmlfile-1.1.0-py3-none-any.whl (4.7 kB)
Installing collected packages: et-xmlfile, openpyxl
Successfully installed et-xmlfile-1.1.0 openpyxl-3.0.7
WARNING: Running pip as root will break packages and permissions. You should install packa
ges reliably by using venv: https://pip.pypa.io/warnings/venv
```

```
################################################################################
############
# FUNCTIONS
################################################################################
############

def divider(ini, stop, denominator):
    '''Acts as a range() but instead of adding up or subtracting, it
multiplies/divides'''
    l = []
    if ini > stop:
        while ini > stop:
            l.append(int(ini))
            ini /= denominator
    else:
        while ini < stop:
            l.append(int(ini))
            ini *= denominator

    return l

def compute_diff(df, cols):
    '''This computes the difference of certain features between player1 and
player2'''
    # ALWAYS player 1 - player 2
    # cols shouldn't have player suffixes
    for col in cols:
        df[col + "_diff"] = df['p1_' + col] - df['p2_' + col]
    return df

def myround(x, base=.5):
    '''This rounds to the desired base (default to .5)'''
    return base * round(x/base)

def f(x):
```

```
    '''This will be used to display information when grouping by odds or
confidence'''
    d = {}
    d["Games"] = x["Right"].count()
    d['Accuracy'] = x['Right'].mean()
    d['Mean Odds'] = x['Odd'].mean()
    d['Profits (unit)'] = x['Profits_unit'].sum()
    d['Profits (odd)'] = x['Profits_odd'].sum()
    d['ROI (unit)'] = x['Profits_unit'].sum() / x['Odd'].count()
    d['ROI (odd)'] = x['Profits_odd'].sum() / x['Odd'].sum()
    return pd.Series(d, index=d.keys())
```

البيانات

تتكون البيانات المتاحة من نوعين مختلفين من الملفات، يتم استخراج كلا النوعين من مصادر مختلفة، ويصل عدد الملفات لكل نوع إلى عدد السنوات بين عامي 2005 و2021 (كلاهما متضمن).

- data/atpmatches{year}.csv: تحتوي على معلومات نوعية قيمة (مثل يد كل لاعب player's hand، والسطح surface، ومستوى البطولة tourney level...) بالإضافة إلى بيانات كمية من هذا الحدث المعين (إحصائيات المباراة النموذجية).
- data/{year}.xls[x]: تحتوي هذه بشكل أساسي على معلومات تتعلق بالاحتمالات بالإضافة إلى نتائج المباراة والبيانات الوصفية.

دعنا نلقي نظرة سريعة على كلا النوعين:

```
pd.read_csv("/kaggle/input/atp-masters-tennis-
dataset/atp_matches_2021.csv").head()
```

	tourney_id	tourney_name	surface	draw_size	tourney_level	tourney_date	match_num	winner_id	winner_seed	winner_entr
0	2021-0499	Delray Beach	Hard	32	A	20210104	300	128034	4.0	NaN
1	2021-0499	Delray Beach	Hard	32	A	20210104	299	128034	4.0	NaN
2	2021-0499	Delray Beach	Hard	32	A	20210104	298	200624	NaN	NaN
3	2021-0499	Delray Beach	Hard	32	A	20210104	297	106328	NaN	Q
4	2021-0499	Delray Beach	Hard	32	A	20210104	296	128034	4.0	NaN

```
pd.read_excel("/kaggle/input/tennis-2021/2021.xlsx").head()
```

	ATP	Location	Tournament	Date	Series	Court	Surface	Round	Best of	Winner	Loser	WRank	LRank	WPt
0	1	Antalya	Antalya Open	2021-01-07	ATP250	Indoor	Hard	1st Round	3	Basilashvili N.	Arnaboldi A.	40	267.0	139
1	1	Antalya	Antalya Open	2021-01-07	ATP250	Indoor	Hard	1st Round	3	Celikbilek A.	Zuk K.	309	262.0	150
2	1	Antalya	Antalya Open	2021-01-07	ATP250	Indoor	Hard	1st Round	3	Ruusuvuori E.	Vesely J.	87	67.0	806
3	1	Antalya	Antalya Open	2021-01-07	ATP250	Indoor	Hard	1st Round	3	Bublik A.	Caruso S.	49	76.0	109
4	1	Antalya	Antalya Open	2021-01-07	ATP250	Indoor	Hard	1st Round	3	Goffin D.	Herbert P.H.	16	83.0	255

إعداد البيانات

ستكون هذه عملية ميكانيكية إلى حد كبير تتكون من سلسلة من الخطوات:

1. إنشاء التحويلات اللازمة لضم مجموعات البيانات (أي تطبيع أسماء اللاعبين والبطولات وتواريخها...)
2. ضم مجموعات البيانات، وإنشاء مجموعة واحدة سنويًا.
3. التخلص من الأعمدة غير الضرورية.
4. إعادة تنظيم مجموعة البيانات Reorganize dataset: من الفائز والخاسر إلى اللاعب 1 واللاعب 2، موزعة بالتساوي (وهذا يعني أن اللاعب 1 فازفي نصف الوقت واللاعب 2 فاز في النصف الآخر). قم أيضًا بإنشاء عمود جديد باسم "فوز اللاعب 1 Player1Win".
5. إنشاء ميزات جديدة؟

تحويل أعمدة المعرف

الخطة هي ربط كلا النوعين من إطارات البيانات، وللقيام بذلك، سأستخدم الأعمدة التالية كمعرفات (التنسيق: column_name_df1 = column_name_df2)، وبعضها سأضطر إلى إنشائه:

- Month = month
- Year = year
- winner = winner_name
- loser = loser_name

إذا نظرنا عن كثب إلى كلا النوعين من إطارات البيانات، فإن أحدهما يحتوي على أسماء كاملة لكلا اسمي اللاعبين بينما يحتوي الآخر على الاسم الأخير فقط متبوعًا بالحرف الأول من اسم اللاعب. سأستخدم حزمة fuzzywuzzy لإجراء بعض المطابقة بين السلاسل.

```
years = [y for y in range(2005, 2022)]
d = {}

for year in tqdm(years):
    # Type of XLS file changed in 2013. We'll read them accordingly.
    if year < 2013:
        df1 = pd.read_excel("/kaggle/input/tennis-data-
atp/{}.xls".format(year))
    elif year < 2019:
        df1 = pd.read_excel("/kaggle/input/tennis-data-
atp/{}.xlsx".format(year))
    elif year < 2021:
        df1 = pd.read_excel("/kaggle/input/atp-
mens/ATP_Data/{}.xlsx".format(year))
    else:
        df1 = pd.read_excel("/kaggle/input/tennis-2021/2021.xlsx")
    df2 = pd.read_csv("/kaggle/input/atp-masters-tennis-
dataset/atp_matches_{}.csv".format(year))
```

```
    # Converting dates into two columns: year and month -> year will always
be the same as the iteration variable
    # but let's do things right just in case
    df1["Year"] = df1["Date"].dt.year
    df1["Month"] = df1["Date"].dt.month
    df2["year"] = df2["tourney_date"].astype(str).str[:4].astype(int)
    df2["month"] = df2["tourney_date"].astype(str).str[4:6].astype(int)

    # Formatting strings in player names to match the other df
    l = []
    w = []
    players_w = pd.unique(df2["winner_name"])
    players_l = pd.unique(df2["loser_name"])
    for i,row in df1.iterrows():
        winner = row["Winner"]
        loser = row["Loser"]

        w1 = process.extract(winner, players_w)
        l1 = process.extract(loser, players_l)

        if len(w1) > 0:
            w.append(w1[0][0])
        else:
            w.append("")

        if len(l1) > 0:
            l.append(l1[0][0])
        else:
            l.append("")

    df1["winner"] = w
    df1["loser"] = l

    d[year] = [df1, df2]
```

ربط مجموعات البيانات

```
for year in years:
    d[year] = d[year][0].merge(d[year][1], left_on=["Month", "Year",
"winner", "loser"], right_on=["month", "year", "winner_name",
"loser_name"])
```

كما هو متوقع، فقدنا بعض المباريات في هذه العملية، وبلغ متوسط النسبة 13.24%. هل هذا كثير؟ أليس كذلك؟ أود أن أقول إنه أعلى من المثالي ولكنه لا يزال جيداً بما فيه الكفاية.

إزالة الأعمدة غير المرغوب فيها

أولاً وقبل كل شيء، لا تحتوي جميع إطارات البيانات على نفس الأعمدة، لذا يجب أن نضيف الاتساق هنا. دعنا نراجعها جميعًا بسرعة لمعرفة أيها يحتوي على أقل عدد من الأعمدة.

```
for year in d:
    print("{} has {} columns".format(year, len(d[year].columns)))
```

```
2005 has 93 columns
2006 has 93 columns
2007 has 93 columns
2008 has 93 columns
2009 has 93 columns
2010 has 97 columns
2011 has 97 columns
2012 has 97 columns
```

```
2013 has 97 columns
2014 has 97 columns
2015 has 95 columns
2016 has 95 columns
2017 has 95 columns
2018 has 95 columns
2019 has 91 columns
2020 has 91 columns
2021 has 91 columns
```

2021 هو العام الذي يحتوي على أقل عدد من الأعمدة. فلنستخدمها كأساس ونرى ما إذا كانت السنوات الأخرى تحتوي على نفس الأعمدة.

بعد ذلك، من بين جميع الأعمدة المتاحة حاليًا، سأقوم بإزالة بعض الأعمدة التي أعتقد أنها لن تكون ذات صلة. سأحتفظ ببعض هذه الأعمدة لفهم البيانات بشكل أفضل.

سأتأكد من اكتمال جميع الألعاب المتاحة (لم تنته بسبب الإصابة أو أسباب أخرى)، وأيضًا سأحتفظ بتلك الصفوف التي تتطابق فيها رتب رابطة محترفي التنس ATP ونقاط ATPفي كل من اطر البيانات dfs ولإزالة تلك الصفوف التي لا تحتوي على احتمالات.

أخيرًا، سأقوم بإنشاء إطار بيانات واحد من كل هذه الأعمدة، حتى يكون لدينا جميع البياناتفي بُنية بيانات واحدة.

```
cols = d[2021].columns.tolist() # Choosing 2021 because it's the one with
less columns

for year in d:
    cols = [col for col in cols if col in d[year].columns]
print("Final number of columns: {}".format(len(cols)))

# Unwanted columns
remove = ["ATP", "Location", "Tournament", "Date", "Series", "Best of",
"Winner", "Loser", "Year", "Month",
          "tourney_id", "tourney_name", "surface", "draw_size",
"tourney_level", "tourney_date", "match_num",
          "winner_id", "winner_entry", "winner_seed", "winner", "loser",
"winner_ioc", "loser_id", "loser_seed",
          "loser_entry", "loser_ioc", "score", "best_of", "round",
"winner_rank", "loser_rank", "winner_rank_points",
          "loser_rank_points", "Comment"
         ]
wanted_cols = [col for col in cols if col not in remove]

# Making sure ranks and points coincide, also checking that the game had
been completed and odds are not missing.
for year in d:
    d[year] = d[year][(d[year]["WRank"] == d[year]["winner_rank"]) &
(d[year]["LRank"] == d[year]["loser_rank"]) &
            (d[year]["WPts"] == d[year]["winner_rank_points"]) &
(d[year]["LPts"] == d[year]["loser_rank_points"])
            & (d[year]["Comment"] == "Completed") &
(~d[year]["B365W"].isna()) & (~d[year]["B365L"].isna())
           ][wanted_cols]
```

```
# Creating a single dataframe for all data, using just the desired columns
df = pd.concat([d[year] for year in d])
df.reset_index(drop=True, inplace=True)
df = df[['Court', 'Surface', 'Round', 'year', 'month', 'minutes',
         'winner_name', 'WRank', 'WPts', 'W1', 'W2', 'W3', 'W4', 'W5',
'Wsets', 'B365W', 'winner_hand', 'winner_ht',
         'winner_age', 'w_ace', 'w_df', 'w_svpt', 'w_1stIn', 'w_1stWon',
'w_2ndWon', 'w_SvGms', 'w_bpSaved',
         'w_bpFaced',
         'loser_name', 'LRank', 'LPts', 'L1', 'L2', 'L3', 'L4', 'L5',
'Lsets', 'B365L', 'loser_hand', 'loser_ht',
         'loser_age', 'l_ace', 'l_df', 'l_svpt', 'l_1stIn', 'l_1stWon',
'l_2ndWon', 'l_SvGms', 'l_bpSaved',
         'l_bpFaced']]

# I'll keep working with the same variable, df, all the time (aesthetics).
It's a good practice to save
# the current work in another variable, just in case I mess up later.
df_v1 = df.copy()

df.head()
```

```
Final number of columns: 85
```

	Court	Surface	Round	year	month	minutes	winner_name	WRank	WPts	W1	W2	W3	W4	W5	Wsets
0	Outdoor	Clay	1st Round	2005	7	69.0	Tommy Robredo	20.0	1425.0	7.0	6	NaN	NaN	NaN	2.0
1	Outdoor	Clay	1st Round	2005	7	50.0	Mikhail Youzhny	27.0	1200.0	6.0	6	NaN	NaN	NaN	2.0
2	Outdoor	Clay	1st Round	2005	7	61.0	Juan Carlos Ferrero	31.0	1150.0	6.0	6	NaN	NaN	NaN	2.0
3	Outdoor	Clay	1st Round	2005	7	67.0	Tomas Berdych	42.0	811.0	6.0	6	NaN	NaN	NaN	2.0
4	Outdoor	Clay	1st Round	2005	7	66.0	Nicolas Almagro	88.0	469.0	6.0	6	NaN	NaN	NaN	2.0

إعادة تنظيم مجموعة البيانات

أعتقد أن الوقت مناسب لبدء تشكيل البيانات بالشكل الذي سنحتاجه. فبدلاً من وجود فائز وخاسر، سأقوم بإنشاء لاعب 1 ولاعب 2. ولأغراض مستقبلية، سأجعل نصف الوقت يفوز فيه اللاعب 1، والنصف الآخر سيكون للاعب 2 (مع إبقاء الأمر كما هو الآن، سيتعلم الناتج أن اللاعب الأول يفوز دائمًا وسيكون ذلك خطأً فادحًا).

من الواضح أنني سأحتاج إلى عمود إضافي يوضح اللاعب الفائز: "p1_win".

```
data = []
p1_wins = []

inverted_cols = df.columns[:6].tolist() + df.columns[28:].tolist() +
df.columns[6:28].tolist()

final_cols = ['Court', 'Surface', 'Round', 'year', 'month', 'minutes',
              'p1_name', 'p1_Rank', 'p1_Pts', 'p1_1', 'p1_2', 'p1_3',
'p1_4', 'p1_5', 'p1_sets', 'p1_B365',
              'p1_hand', 'p1_ht', 'p1_age', 'p1_ace', 'p1_df', 'p1_svpt',
'p1_1stIn', 'p1_1stWon', 'p1_2ndWon',
              'p1_SvGms', 'p1_bpSaved', 'p1_bpFaced',
```

```
                'p2_name', 'p2_Rank', 'p2_Pts', 'p2_1', 'p2_2', 'p2_3',
'p2_4', 'p2_5', 'p2_sets', 'p2_B365',
                'p2_hand', 'p2_ht', 'p2_age', 'p2_ace', 'p2_df', 'p2_svpt',
'p2_1stIn', 'p2_1stWon', 'p2_2ndWon',
                'p2_SvGms', 'p2_bpSaved', 'p2_bpFaced']

for i,row in tqdm(df.iterrows()):
    if len(p1_wins) == 0 or statistics.mean(p1_wins) <= 0.5: # More 0 than
1 (or same) -> we add 1 (that means, player1 is the one who won)
        p1_wins.append(1)
        data.append(row.tolist())
    else:
        p1_wins.append(0)
        data.append(row[inverted_cols].tolist())

df = pd.DataFrame(data, columns = final_cols)
df["p1_2"] = df["p1_2"].astype(float)
df["p1_3"] = df["p1_3"].replace(" ", np.nan)
df["p1_3"] = df["p1_3"].astype(float)
df["p2_2"] = df["p2_2"].astype(float)
df["p2_3"] = df["p2_3"].replace(" ", np.nan)
df["p2_3"] = df["p2_3"].astype(float)
df["p1_win"] = p1_wins
df
```

	Court	Surface	Round	year	month	minutes	p1_name	p1_Rank	p1_Pts	p1_1	p1_2	p1_3	p1_4	p1
0	Outdoor	Clay	1st Round	2005	7	69.0	Tommy Robredo	20.0	1425.0	7.0	6.0	NaN	NaN	Na
1	Outdoor	Clay	1st Round	2005	7	50.0	Jerome Haehnel	109.0	391.0	1.0	1.0	NaN	NaN	Na
2	Outdoor	Clay	1st Round	2005	7	61.0	Juan Carlos Ferrero	31.0	1150.0	6.0	6.0	NaN	NaN	Na
3	Outdoor	Clay	1st Round	2005	7	67.0	Kevin Kim	71.0	541.0	2.0	3.0	NaN	NaN	Na
4	Outdoor	Clay	1st Round	2005	7	66.0	Nicolas Almagro	88.0	469.0	6.0	6.0	NaN	NaN	Na
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
32071	Outdoor	Clay	Quarterfinals	2021	4	87.0	Filip Krajinovic	33.0	1683.0	4.0	4.0	NaN	NaN	Na
32072	Outdoor	Clay	Quarterfinals	2021	4	75.0	Novak Djokovic	1.0	11873.0	6.0	6.0	NaN	NaN	Na
32073	Outdoor	Clay	Quarterfinals	2021	4	84.0	Gianluca Mager	98.0	813.0	3.0	4.0	NaN	NaN	Na
32074	Outdoor	Clay	Semifinals	2021	4	205.0	Aslan Karatsev	28.0	1946.0	7.0	4.0	6.0	NaN	Na
32075	Outdoor	Clay	The Final	2021	4	147.0	Aslan Karatsev	28.0	1946.0	1.0	6.0	6.0	NaN	Na

معالجة البيانات

هنا تبدأ الأمورفي أن تصبح مثيرة للاهتمام. بعد قليل، ستراني أقوم بتنظيف البيانات وإنشاء ميزات جديدة...

التحقق من الاتساق في Odds

```
# Visualize them
for i,row in df.iterrows():
    if (1/row["p2_B365"]) + (1/row["p1_B365"]) < 1:
        print(row["p2_B365"], row["p1_B365"])
    if row["p2_B365"]<1 or row["p1_B365"]<1:
```

```
        print(row["p2_B365"], row["p1_B365"])

# Drop them
df = df[(df["p2_B365"]>1) & (df["p1_B365"]>1)]
```

```
  0.967 29.0
  0.971 34.0
  0.967 29.0
```

كل شيء على ما يرام الآن. دعنا ننشئ بعض الميزات الأكثر تقدمًا.

إنشاء الميزات

إذا كنت من الأشخاص الذين يحبون التنس ويحبون الإحصائيات، فأنا متأكد من أنك على دراية بموقع Ultimate Tennis Statistics. إذا لم تكن كذلك، فلا تترددفي إلقاء نظرة عليه (إنه موقع ويب مليء بإحصائيات التنس، كما استنتجت بالفعل).

أشير إلى هذه الخدمة لأنها تحتوي على إحصائيات متقدمة يمكنني استخدامها لهذا التحليل. بشكل ملموس، يوجد هذا القاموس الذي يحتوي على الصيغ، والتي سأقوم بإنشاء بعضها الآن. لا تقلق إذا لم يكن هناك شيء واضح، فهو مجرد كود يقوم بإجراء حسابات رياضية بسيطة لإنشاء هذه الميزات المتقدمة.

```
# First, turn the missing values into 0 within the games-won-per-set
columns
df["p1_3"].fillna(0, inplace=True)
df["p1_4"].fillna(0, inplace=True)
df["p1_5"].fillna(0, inplace=True)
df["p2_3"].fillna(0, inplace=True)
df["p2_4"].fillna(0, inplace=True)
df["p2_5"].fillna(0, inplace=True)

# 1st Serve Effectiveness
df["p1_1stWon%"] = df["p1_1stWon"] / df["p1_1stIn"]
df["p1_2ndWon%"] = df["p1_2ndWon"] / (df["p1_svpt"] - df["p1_1stIn"])
df["p1_1stServeEffectiveness"] = df["p1_1stWon%"]/df["p1_2ndWon%"]

df["p2_1stWon%"] = df["p2_1stWon"] / df["p2_1stIn"]
df["p2_2ndWon%"] = df["p2_2ndWon"] / (df["p2_svpt"] - df["p2_1stIn"])
df["p2_1stServeEffectiveness"] = df["p2_1stWon%"]/df["p2_2ndWon%"]

# Return to Service Points Ratio
df["p1_Ret2ServPtsRatio"] = df["p2_svpt"] / df["p1_svpt"]
df["p2_Ret2ServPtsRatio"] = df["p1_svpt"] / df["p2_svpt"]

# Point Dominance Ratio
df["p1_ServeWon%"] = (df["p1_1stWon"] + df["p1_2ndWon"]) / df["p1_svpt"]
df["p1_ReturnWon%"] = 1 - df["p1_ServeWon%"]

df["p2_ServeWon%"] = (df["p2_1stWon"] + df["p2_2ndWon"]) / df["p2_svpt"]
df["p2_ReturnWon%"] = 1 - df["p2_ServeWon%"]

df["p1_PtsDominanceRatio"] = df["p1_ReturnWon%"] / df["p2_ReturnWon%"]
df["p2_PtsDominanceRatio"] = df["p2_ReturnWon%"] / df["p1_ReturnWon%"]
```

```
# Break Points Ratio
df["p1_BPConverted%"] = (df["p2_bpFaced"] - df["p2_bpSaved"]) /
df["p2_bpFaced"]
df["p2_BPConverted%"] = (df["p1_bpFaced"] - df["p1_bpSaved"]) /
df["p1_bpFaced"]

df["p1_BPRatio"] = df["p1_BPConverted%"] / df["p2_BPConverted%"]
df["p2_BPRatio"] = df["p2_BPConverted%"] / df["p1_BPConverted%"]

# Points to Sets Over-Performing Ratio
df["p1_SetWon%"] = df["p1_sets"] / (df["p1_sets"] + df["p2_sets"])
df["p1_PtsWon%"] = (df["p1_1stWon"] + df["p1_2ndWon"] + df["p2_1stIn"] -
df["p2_1stWon"] + (df["p2_svpt"] - df["p2_1stIn"]) - df["p2_2ndWon"]) /
(df["p1_svpt"] + df["p2_svpt"])
df["p1_Pts2Sets_OP_Ratio"] = df["p1_SetWon%"] / df["p1_PtsWon%"]

df["p2_SetWon%"] = df["p2_sets"] / (df["p1_sets"] + df["p2_sets"])
df["p2_PtsWon%"] = (df["p2_1stWon"] + df["p2_2ndWon"] + df["p1_1stIn"] -
df["p1_1stWon"] + (df["p1_svpt"] - df["p1_1stIn"]) - df["p1_2ndWon"]) /
(df["p1_svpt"] + df["p2_svpt"])
df["p2_Pts2Sets_OP_Ratio"] = df["p2_SetWon%"] / df["p2_PtsWon%"]

# Points to Games Over-Performing Ratio
df["p1_GmsWon%"] = (df["p1_1"] + df["p1_2"] + df["p1_3"] + df["p1_4"] +
df["p1_5"]) / (df["p1_1"] + df["p1_2"] + df["p1_3"] + df["p1_4"] +
df["p1_5"] + df["p2_1"] + df["p2_2"] + df["p2_3"] + df["p2_4"] +
df["p2_5"])
df["p1_Pts2Gms_OP_Ratio"] = df["p1_GmsWon%"] / df["p1_PtsWon%"]

df["p2_GmsWon%"] = (df["p2_1"] + df["p2_2"] + df["p2_3"] + df["p2_4"] +
df["p2_5"]) / (df["p1_1"] + df["p1_2"] + df["p1_3"] + df["p1_4"] +
df["p1_5"] + df["p2_1"] + df["p2_2"] + df["p2_3"] + df["p2_4"] +
df["p2_5"])
df["p2_Pts2Gms_OP_Ratio"] = df["p2_GmsWon%"] / df["p2_PtsWon%"]

# Games to Sets Over-Performing Ratio
df["p1_Gms2Sets_OP_Ratio"] = df["p1_SetWon%"] / df["p1_GmsWon%"]
df["p2_Gms2Sets_OP_Ratio"] = df["p2_SetWon%"] / df["p2_GmsWon%"]

# Break Points Over-Performing Ratio
df["p1_BPWon%"] = (df["p2_bpFaced"] - df["p2_bpSaved"] + df["p1_bpSaved"])
/ (df["p1_bpFaced"] + df["p2_bpFaced"])
df["p1_BP_OP_Ratio"] = df["p1_BPWon%"] / df["p1_PtsWon%"]

df["p2_BPWon%"] = (df["p1_bpFaced"] - df["p1_bpSaved"] + df["p2_bpSaved"])
/ (df["p1_bpFaced"] + df["p2_bpFaced"])
df["p2_BP_OP_Ratio"] = df["p2_BPWon%"] / df["p2_PtsWon%"]

# Break Points Saved Over-Performing Ratio
df["p1_BPSaved%"] = df["p1_bpSaved"] / df["p1_bpFaced"]
df["p1_BPSaved_OP_Ratio"] = df["p1_BPSaved%"] / df["p1_ServeWon%"]

df["p2_BPSaved%"] = df["p2_bpSaved"] / df["p2_bpFaced"]
df["p2_BPSaved_OP_Ratio"] = df["p2_BPSaved%"] / df["p2_ServeWon%"]

# Break Points Converted Over-Performing Ratio
df["p1_BPConverted_OP_Ratio"] = df["p1_BPConverted%"] / df["p1_ReturnWon%"]
df["p2_BPConverted_OP_Ratio"] = df["p2_BPConverted%"] / df["p2_ReturnWon%"]

 # Extras I might need
df["p1_Ace%"] = df["p1_ace"]/df["p1_svpt"]
df["p1_DF%"] = df["p1_df"]/df["p1_svpt"]
```

```
df["p1_1stServe%"] = df["p1_1stIn"] / df["p1_svpt"]
df["p1_1stReturnWon%"] = (df["p2_1stIn"] - df["p2_1stWon"]) /
df["p2_1stIn"]

df["p2_Ace%"] = df["p2_ace"]/df["p2_svpt"]
df["p2_DF%"] = df["p2_df"]/df["p2_svpt"]
df["p2_1stServe%"] = df["p2_1stIn"] / df["p2_svpt"]
df["p2_1stReturnWon%"] = (df["p1_1stIn"] - df["p1_1stWon"]) /
df["p1_1stIn"]

# Upsets
df["p1_UpsetScored"] = [1 if (row["p1_Rank"] < row["p2_Rank"] and
row["p1_win"] == 1) else 0 for i,row in df.iterrows()]
df["p2_UpsetScored"] = [1 if (row["p1_Rank"] > row["p2_Rank"] and
row["p1_win"] == 0) else 0 for i,row in df.iterrows()]
df["p1_UpsetAgainst"] = df["p2_UpsetScored"]
df["p2_UpsetAgainst"] = df["p1_UpsetScored"]

# Rank variation
r1, r2 = [], []
for i,row in tqdm(df.iterrows()):
    year = row["year"]
    month = row["month"]
    p1 = row["p1_name"]
    p2 = row["p2_name"]
    p1_rank = row["p1_Rank"]
    p2_rank = row["p2_Rank"]

    if month < 7:
        year -= 1
        month = 12 + month - 6
    else:
        month -= 6

    try:
        aux1 = df[((df["year"] == year) & (df["month"] <= month)) |
(df["year"] < year)].loc[(df["p1_name"] == p1) | (df["p2_name"] ==
p1)].iloc[-1]
        prev_rank1 = aux1["p1_Rank"] if aux1["p1_name"] == p1 else
aux1["p2_Rank"]
        r1.append(prev_rank1 - p1_rank)
    except:
        r1.append(0)

    try:
        aux2 = df[((df["year"] == year) & (df["month"] <= month)) |
(df["year"] < year)].loc[(df["p1_name"] == p2) | (df["p2_name"] ==
p2)].iloc[-1]
        prev_rank2 = aux2["p1_Rank"] if aux2["p1_name"] == p2 else
aux2["p2_Rank"]
        r2.append(prev_rank2 - p2_rank)
    except:
        r2.append(0)

df["p1_RankVariation"] = r1
df["p2_RankVariation"] = r2

df.head()
```

	Court	Surface	Round	year	month	minutes	p1_name	p1_Rank	p1_Pts	p1_1	p1_2	p1_3	p1_4	p1_5	p1_set
0	Outdoor	Clay	1st Round	2005	7	69.0	Tommy Robredo	20.0	1425.0	7.0	6.0	0.0	0.0	0.0	2.0
1	Outdoor	Clay	1st Round	2005	7	50.0	Jerome Haehnel	109.0	391.0	1.0	1.0	0.0	0.0	0.0	0.0
2	Outdoor	Clay	1st Round	2005	7	61.0	Juan Carlos Ferrero	31.0	1150.0	6.0	6.0	0.0	0.0	0.0	2.0
3	Outdoor	Clay	1st Round	2005	7	67.0	Kevin Kim	71.0	541.0	2.0	3.0	0.0	0.0	0.0	0.0
4	Outdoor	Clay	1st Round	2005	7	66.0	Nicolas Almagro	88.0	469.0	6.0	6.0	0.0	0.0	0.0	2.0

معالجة القيم الفارغة

دعونا نفحص نسبة القيم المفقودة missing valuesفي مجموعة البيانات بأكملها:

```
pd.set_option('display.max_rows', 50)
df.isnull().sum().sort_values(ascending=False).head(50)/df.shape[0]
```

```
p1_ht                          0.128018
p2_ht                          0.126363
p2_BPRatio                     0.101528
p1_BPRatio                     0.101528
p1_BPSaved_OP_Ratio            0.050826
p2_BPConverted%                0.050826
p2_BPConverted_OP_Ratio        0.050826
p1_BPSaved%                    0.050826
p1_BPConverted_OP_Ratio        0.048390
p2_BPSaved_OP_Ratio            0.048327
p2_BPSaved%                    0.048327
p1_BPConverted%                0.048327
minutes                        0.030614
p2_BP_OP_Ratio                 0.001687
p2_BPWon%                      0.001687
p1_BP_OP_Ratio                 0.001687
p1_BPWon%                      0.001687
p1_Pts2Gms_OP_Ratio            0.001218
p2_Pts2Gms_OP_Ratio            0.001218
p1_Pts2Sets_OP_Ratio           0.001187
p2_Pts2Sets_OP_Ratio           0.001187
p1_1stServeEffectiveness       0.001187
p1_2ndWon%                     0.001187
p1_1stWon%                     0.001156
p2_PtsDominanceRatio           0.001156
p1_1stServe%                   0.001156
p1_ServeWon%                   0.001156
p1_ReturnWon%                  0.001156
p1_PtsDominanceRatio           0.001156
p1_DF%                         0.001156
p1_Ace%                        0.001156
p2_1stReturnWon%               0.001156
p2_1stServeEffectiveness       0.001125
p2_2ndWon%                     0.001125
p2_Ace%                        0.001093
p1_2ndWon                      0.001093
p2_DF%                         0.001093
p2_1stServe%                   0.001093
p2_1stWon%                     0.001093
```

```
p1_Ret2ServPtsRatio          0.001093
p2_ReturnWon%                0.001093
p2_PtsWon%                   0.001093
p2_Ret2ServPtsRatio          0.001093
p2_bpSaved                   0.001093
p1_PtsWon%                   0.001093
p2_ServeWon%                 0.001093
p2_bpFaced                   0.001093
p1_1stWon                    0.001093
p2_SvGms                     0.001093
p1_1stReturnWon%             0.001093
dtype: float64
```

ستحتوي الأعمدة المقابلة للمجموعات 3 و4 و5 على الكثير من القيم المفقودة إذا لم أحولها إلى 0. وهذا منطقي، نظرًا لأن بعض البطولات "أفضل من 3 best of 3" و2–0 ستنهي المباراة بالفعل، دون الحاجة إلى لعب المجموعة 3). لست بحاجة إليها حقًا من الآن فصاعدًا، لذا أعتقد أنني سأتخلص منها.

بعض الارتفاعات heights مفقودة أيضًا. حوالي 21% من الألعاب تفتقر إلى هذه الميزة (سواء للاعبين الفائزين و/أو الخاسرين). يمكنني استخدام التضمين imputation ولكنني أفضل عدم القيام بذلك، فهو لا يبدو طبيعيًا بالنسبة لي.في الوقت الحالي، لن أزيله، لكنني سأقوم بذلك إذا ثبت أنه غير مرتبط أو منخفض الارتباط بفرص الفوز.

أيضًا، هناك 0.1016% من الألعاب التي تفتقر إلى ميزة BPRatio. وذلك لأنها تعتمد على ميزات النسبة المئوية التي يمكن أن تكون NaN بسبب القسمة 0 على 0. هذه تتطلب dropna() واضحة. الأمر نفسه ينطبق على الميزات المتقدمة الأخرى التي قمت بإنشائها.

سأقوم أيضًا بإسقاط تلك الصفوف التي تحتوي على قيم مفقودةفي الإحصائيات التقليدية (الآس ace، الأخطاء المزدوجة double faults...) والتي تكون جميعها مفقودةفي نفس الصفوف.

وبصرف النظر عن ذلك، يمكن حساب بقية القيم المفقودة (عمود الدقائق minutes column) باستخدام Sklearn.

```
pd.set_option('display.max_rows', 20) # Get it back to the default we
established

# 1. drop set columns
df.drop(columns = ["p1_1", "p2_1", "p1_2", "p2_2", "p1_3", "p2_3", "p1_4",
"p2_4", "p1_5", "p2_5"], inplace=True)

# 2. drop na
df.dropna(subset = ["p1_BPRatio", "p2_BPRatio", "p1_Gms2Sets_OP_Ratio",
"p2_Gms2Sets_OP_Ratio",
                    "p1_Pts2Gms_OP_Ratio", "p2_Pts2Gms_OP_Ratio",
"p1_GmsWon%", "p1_2ndWon%",
                    "p1_sets"], inplace=True)

# 3. Impute using sklearn
si = SimpleImputer(strategy = "median")
```

```
si.fit(df[["minutes"]])
df[["minutes"]] = si.transform(df[["minutes"]])

# I'll keep working with the same variable, df, all the time (aesthetics).
It's a good practice to save
# the current work in another variable, just in case I mess up later.
df_v2 = df.copy()

df.head()
```

	Court	Surface	Round	year	month	minutes	p1_name	p1_Rank	p1_Pts	p1_sets	p1_B365	p1_hand	p1_ht	p1_
0	Outdoor	Clay	1st Round	2005	7	69.0	Tommy Robredo	20.0	1425.0	2.0	1.10	R	180.0	23.1
2	Outdoor	Clay	1st Round	2005	7	61.0	Juan Carlos Ferrero	31.0	1150.0	2.0	1.07	R	183.0	25.3
3	Outdoor	Clay	1st Round	2005	7	67.0	Kevin Kim	71.0	541.0	0.0	4.00	R	180.0	26.9
4	Outdoor	Clay	1st Round	2005	7	66.0	Nicolas Almagro	88.0	469.0	2.0	2.75	R	183.0	19.8
5	Outdoor	Clay	1st Round	2005	7	85.0	Nicolas Lapentti	98.0	410.0	0.0	2.20	R	188.0	28.8

قد يعتقد البعض أننا قمنا بما يكفي. يبدو أن البيانات نظيفة وجاهزة للتحليل، وهذا صحيح. ولكن دعنا نعود إلى الأهداف أو الدوافع التي دفعتني إلى إنشاء هذا التحليل: التنبؤ بالفائزفي المباراة.

إذا حدث وحللتُ كل هذه الميزات وكيف ارتبطت بفوز اللاعب 1 بالمباراة، فسيكون ذلك خطأً وقد يؤدي إلى استنتاجات مضللة. لماذا؟ لأن كل صف يحتوي على البيانات من تلك المباراة، وهي معلومات من الواضح أننا لا نمتلكها قبل الحدث، عندما نريد إجراء التنبؤ.

يجب علينا، بطريقة ما، أن نفعل شيئًا لاستخدام المعلومات السابقة لوقت المباراة لتحليل تأثيراتها على نتيجة المباراة. كيف؟ المتوسطات المتحركة Rolling averages.

المتوسطات المتحركة

يتلخص إنشاء متوسط متحرك rolling average ببساطةفي إدخال متوسط الصفوف X السابقةفي الصف الحالي. نريد أن نجعل الأمر لا يأخذفي الاعتبار الصف الفعلي للمتوسط، وعدد الألعاب التي سأستخدمها هو 30.

وعلاوة على ذلك، سأضيف ميزة نسبة الفوز Win%.

```
# 1. Transform to long-format table (two rows per match, one per each
player)
p1 = [col for col in df.columns if "p1_" in col and col != "p1_win"]
p2 = [col for col in df.columns if "p2_" in col]
info = [col for col in df.columns if "p1_" not in col and "p2_" not in col]

new_cols = ["Win"] + info + [col[3:] for col in p1]
l = []
for i,row in df.iterrows():
    l.append([row["p1_win"]] + row[info + p1].tolist())
```

```
    l.append([abs(1-row["p1_win"])] + row[info + p2].tolist())

df = pd.DataFrame(l, columns = new_cols)
players = pd.unique(df["name"])
df["Win%"] = df["Win"]

# Columns to average
nums_avg = [
    "minutes", "sets", "ace", "df", "svpt", "1stIn", "1stWon", "2ndWon",
"SvGms",
    "bpSaved", "bpFaced", "1stWon%", "2ndWon%", "1stServeEffectiveness",
"Ret2ServPtsRatio", "ServeWon%",
    "ReturnWon%", "PtsDominanceRatio", "BPConverted%", "BPRatio",
"SetWon%", "PtsWon%", "Pts2Sets_OP_Ratio",
    "GmsWon%", "Pts2Gms_OP_Ratio", "Gms2Sets_OP_Ratio", "BPWon%",
"BP_OP_Ratio", "BPSaved%", "BPSaved_OP_Ratio",
    "BPConverted_OP_Ratio", "Ace%", "DF%", "1stServe%", "1stReturnWon%",
"UpsetScored", "UpsetAgainst", "Win%"
           ]
# Rolling averages
window = 30
for player in tqdm(players):
    for col in nums_avg:
        df.loc[df["name"] == player, col] = (
            df.loc[df["name"] == player, col].shift(1).rolling(window,
min_periods = 10).mean()
        )

df.rename(columns={"UpsetScored": "UpsetsScored%", "UpsetAgainst":
"UpsetsAgainst%"}, inplace=True)
df
```

	Win	Court	Surface	Round	year	month	minutes	name	Rank	Pts	sets	B365	hand
0	1	Outdoor	Clay	1st Round	2005	7	NaN	Tommy Robredo	20.0	1425.0	NaN	1.10	R
1	0	Outdoor	Clay	1st Round	2005	7	NaN	Michal Tabara	112.0	381.0	NaN	6.00	R
2	1	Outdoor	Clay	1st Round	2005	7	NaN	Juan Carlos Ferrero	31.0	1150.0	NaN	1.07	R
3	0	Outdoor	Clay	1st Round	2005	7	NaN	Lukas Dlouhy	136.0	315.0	NaN	7.00	R
4	0	Outdoor	Clay	1st Round	2005	7	NaN	Kevin Kim	71.0	541.0	NaN	4.00	R
...	...	...	...	...	...	...	...	...	...	...	...	...	...
57485	1	Outdoor	Clay	Quarterfinals	2021	4	117.629630	Aslan Karatsev	28.0	1946.0	1.518519	1.44	R
57486	1	Outdoor	Clay	Semifinals	2021	4	116.428571	Aslan Karatsev	28.0	1946.0	1.535714	6.00	R
57487	0	Outdoor	Clay	Semifinals	2021	4	117.433333	Novak Djokovic	1.0	11873.0	2.000000	1.12	R
57488	0	Outdoor	Clay	The Final	2021	4	119.482759	Aslan Karatsev	28.0	1946.0	1.551724	1.80	R
57489	1	Outdoor	Clay	The Final	2021	4	119.800000	Matteo Berrettini	10.0	3453.0	1.733333	2.00	R

بدلاً من إعادة تحويله إلى تنسيق أوسع، أعتقد أن الإبقاء عليه على هذا النحو قد يكون مفيداًفي وقت التحليل (لا يتعين علينا الاختلاف بين اللاعب 1 واللاعب 2، لذا لدينا صورة أوضح لكيفية ارتباط الإحصائيات بالميزات الأخرى).

تحليل البيانات وتصورها

الملخص

لنقم بتشغيل describe() سريعًا لنرى ملخصًا موجزًا لبياناتنا.

```
df.describe()
```

	Win	year	month	minutes	Rank	Pts	sets	B365
count	57490.000000	57490.000000	57490.000000	51422.000000	57490.000000	57490.000000	51422.000000	57490.000000
mean	0.500000	2012.608419	5.443938	110.992939	73.230057	1395.884710	1.358847	2.666485
std	0.500004	4.483682	2.984767	9.473094	91.038479	1767.390247	0.307591	2.606679
min	0.000000	2005.000000	1.000000	74.000000	1.000000	1.000000	0.100000	1.001000
25%	0.000000	2009.000000	3.000000	104.400000	25.000000	570.000000	1.133333	1.390000
50%	0.500000	2013.000000	5.000000	110.566667	53.000000	860.000000	1.333333	1.830000
75%	1.000000	2016.000000	8.000000	117.000000	90.000000	1430.000000	1.566667	2.870000
max	1.000000	2021.000000	12.000000	172.766667	1848.000000	16950.000000	2.533333	51.000000

البيانات الفئوية

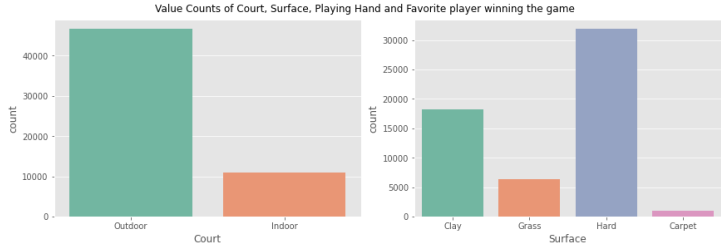
أريد أن أبدأفي تصور البيانات visualizing data. لنبدأ بشكل بسيط، فلنرى كيف يتم توزيع البيانات الفئوية categorical data من خلال رسم عدد القيم value-counts لكل نوع.

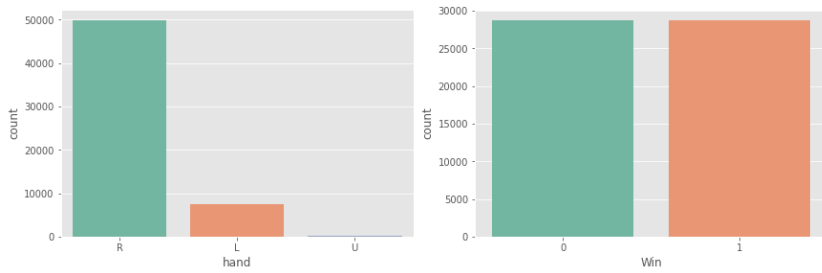
```
categorical = ["Court", "Surface", "hand", "win", "Round"]
numerical = [
    'Rank','Pts', 'sets', 'B365','ht', 'age', 'ace', 'df', 'svpt',
    '1stIn', '1stWon', '2ndWon', 'SvGms', 'bpSaved', 'bpFaced',
"1stServeEffectiveness",
    "Ret2ServPtsRatio", "PtsDominanceRatio", "BPRatio",
"Pts2Sets_OP_Ratio", "Pts2Gms_OP_Ratio",
    "Gms2Sets_OP_Ratio", "BP_OP_Ratio", "BPSaved_OP_Ratio",
"BPConverted_OP_Ratio", "Ace%",
    "DF%", "1stServe%", "1stReturnWon%"
]
other = ["year", "month", "name"]

# Useful lists
res = ["Win"]

fig, axs = plt.subplots(2,2, constrained_layout=True)
fig.suptitle("Value Counts of Court, Surface, Playing Hand and Favorite
player winning the game")
sns.countplot(x="Court", data=df, palette="Set2", ax = axs[0,0])
sns.countplot(x="Surface", data=df, palette="Set2", ax = axs[0,1])
sns.countplot(x="hand", data=df, palette="Set2", ax = axs[1,0])
sns.countplot(x="Win", data=df, palette="Set2", ax = axs[1,1])

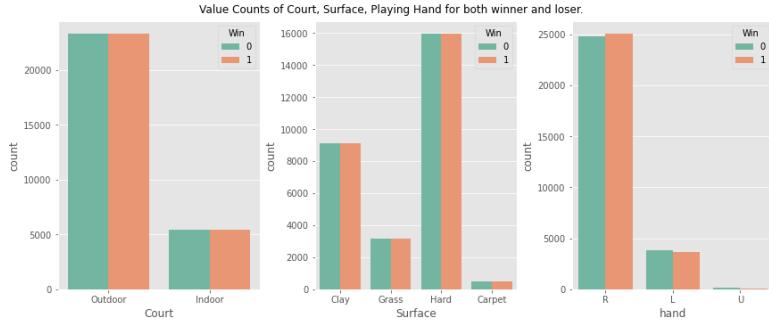
plt.show()
```






```
plt.rcParams["figure.figsize"] = [12,5]
fig, axs = plt.subplots(1, 3, constrained layout=True)
fig.suptitle("Value Counts of Court, Surface, Playing Hand for both winner
and loser.")
sns.countplot(x="Court", data=df, hue = "Win", palette="Set2", ax = axs[0])
sns.countplot(x="Surface", data=df, hue = "Win", palette="Set2", ax =
axs[1])
sns.countplot(x="hand", data=df, hue = "Win", palette="Set2", ax = axs[2])

plt.show()
```



ملاحظات واضحة:

- تُلعب أغلب البطولاتفي الهواء الطلق، ويحظى المرشحون بفرص أعلى للفوز عندما يكون الأمر كذلك.
- السطح الصلب هو السطح الأكثر شيوعًا، يليه الطين، والعشب، والسجاد (بالترتيب).
- معظم اللاعبين يستخدمون اليد اليمنى، تمامًا كما هو الحالفي العمل الحقيقي.
- لا يوجد شيء مفاجئ هنا، فنحن فقط نتعرف على البيانات بشكل أفضل!

البيانات الرقمية

هل حان الوقت لخريطة حرارية؟ دعنا نرى ما إذا كان هناك ارتباط بين المتغيرات الرقمية والميزة المستهدفة من خلال إلقاء نظرة على مصفوفة الارتباط correlation matrix.

```
# Pearson Correlation
corr_matrix = df[numerical + res].corr().sort_values(by=["Win"])
```

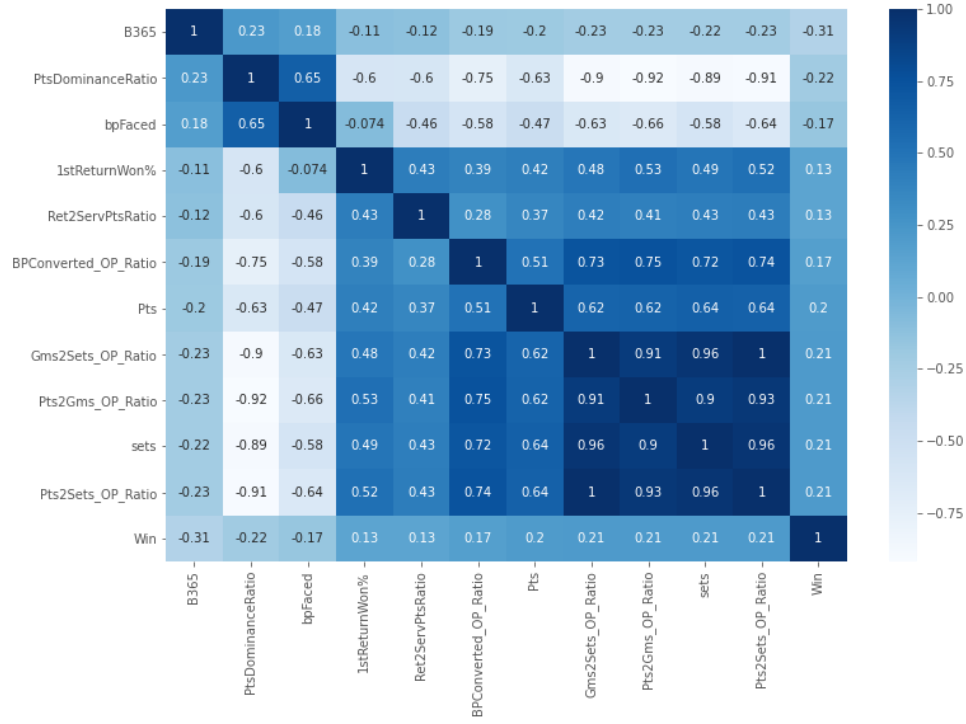
```
corr_matrix
```

	Rank	Pts	sets	B365	ht	age	ace	df	svpt	1stI
B365	0.247165	-0.195683	-0.224522	1.000000	-0.052292	-0.011029	-0.083458	0.051162	0.043445	0.0(
PtsDominanceRatio	0.482828	-0.632955	-0.891757	0.225303	-0.103308	0.026746	-0.200609	0.257048	0.208646	0.0
bpFaced	0.361284	-0.470100	-0.578029	0.181351	-0.301775	-0.004954	-0.478441	0.261552	0.363709	0.2(
Rank	1.000000	-0.371676	-0.484456	0.247165	-0.041210	-0.088472	-0.112685	0.158494	0.133843	0.0
bpSaved	0.268142	-0.365666	-0.380688	0.133334	-0.179305	-0.006546	-0.305634	0.254683	0.461053	0.2
...	...	...	...	...	...	...	...	...	...	...
Gms2Sets_OP_Ratio	-0.487970	0.615898	0.960937	-0.225366	0.148298	-0.034628	0.260549	-0.212169	-0.108612	0.0
Pts2Gms_OP_Ratio	-0.478236	0.622712	0.902950	-0.226286	0.127388	-0.036022	0.212188	-0.235519	-0.158429	-0.0
sets	-0.484456	0.642098	1.000000	-0.224522	0.133164	-0.027406	0.274380	-0.176368	0.019523	0.1
Pts2Sets_OP_Ratio	-0.495155	0.636452	0.958820	-0.227432	0.124339	-0.037993	0.220005	-0.234423	-0.167022	-0.0
Win	-0.162973	0.197851	0.210817	-0.314992	0.035434	-0.008636	0.053212	-0.064706	-0.059394	-0.0

إن إنشاء خريطة حرارية heatmap لجميع المتغيرات التي لدينا من شأنه أن يؤدي إلى إنشاء رسم بياني قبيح يحتوي على العديد من الأرقام المتداخلة مع بعضها البعض. وهذا ليس أمرًا لطيفًا... سأعرض بدلاً من ذلك الأرقام الأكثر ارتباطًا بفوز اللاعب 1 باللعبة.

```
plt.rcParams["figure.figsize"] = [12,8]
matrix = pd.concat([corr_matrix.iloc[:3], corr_matrix.iloc[-9:]])

sns.heatmap(matrix[matrix.index], cmap='Blues', annot=True)
plt.show()
```



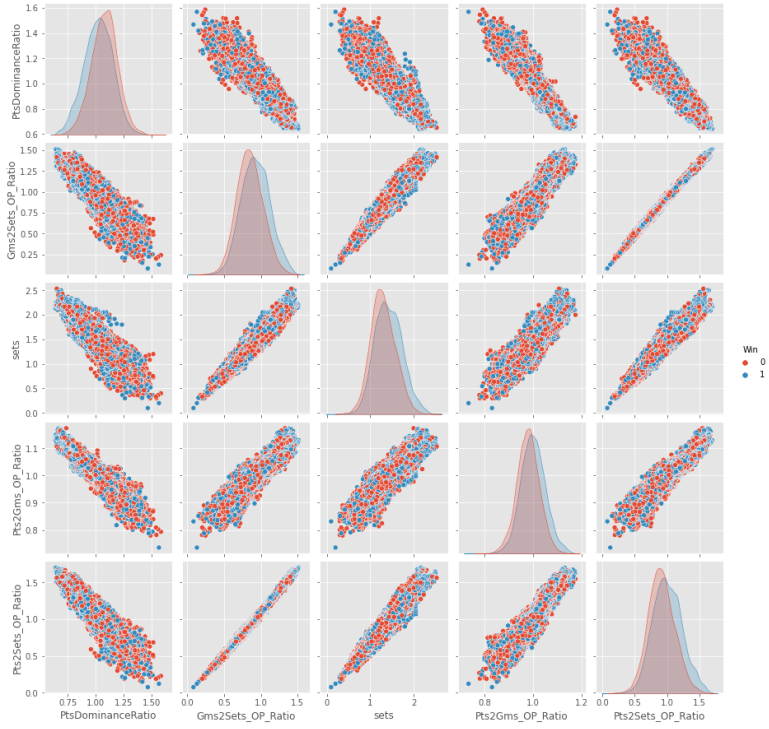
هذا مثير للاهتمام. لا يوجد أي ارتباط كبير، ولكن بعضها أعلى من عتبة 20%، مما يجعلها تستحق النظر!

يمكننا الحصول على رؤى جيدة من هنا، مثل حقيقة أن نسبة النقاط إلى الألعاب فوق الأداء Points to Games Over-Performance Ratio، ونسبة النقاط إلى المجموعات فوق الأداء Points to Sets Over-Performance Ratio، ونسبة الألعاب إلى المجموعات فوق الأداء Games to Sets Over-Performance Ratio ومتوسط عدد المجموعات التي تم الفوز بها average number of sets won مترابطة إلى حد ما بالإضافة إلى نسبة هيمنة النقاط، ومن الواضح، الاحتمالات. يمكننا أن نقول إن وكلاء المراهنات يقومون بعمل جيد إذا كانوا الميزة رقم واحد من حيث الارتباط.

ما هو مدهش، على الأقل صدمني، هو أن نسبة الفوز لا تبدو ذات صلة كبيرة، ولا أي ميزة متقدمة أخرى ليست نسبة. غريب.

معظم هذه الإحصائيات مترابطة بشكل كبير highly correlated فيما بينها. ومن المنطقي، أنها تشترك في علاقة خطية. هل يجب أن ندرس ذلك؟

```
sns.pairplot(data = df[["Win", "PtsDominanceRatio", "Gms2Sets_OP_Ratio",
"sets", "Pts2Gms_OP_Ratio", "Pts2Sets_OP_Ratio"]], diag_kind = 'kde', hue =
"Win")
plt.show()
```



حسنًا، هذا واضح جدًا، أليس كذلك؟ تُظهِر جميعها علاقة خطية قوية جدًا، وهي إشارة إلى التبعية dependence. وهذا أمر طبيعي، حيث تم إنشاؤها كمزيج من ميزات أساسية أخرى.

لقد أضفت أيضًا هذا اللون الذي يشير إلى ما إذا كانت نقطة معينة أدت إلى فوز أو خسارة ذلك اللاعب، فقط لمعرفة كيفية توزيع الألوان في جميع الرسوم البيانية. لأكون صادقًا، كنت أتوقع أن يكون أكثر تميزًا. إذا نظرنا إلى الأرقام وكذلك المخططات البيانية، فإن النمط موجود: بشكل عام، تميل النقاط البرتقالية إلى احتلال مساحة أكبر في أحد الطرفين والأزرق في الطرف الآخر. ولكن، مرة أخرى، الأمر ليس واضحًا للغاية.

أود الآن الانتقال إلى أهمية الميزة ولكن أود أولاً أن أبدأ في ترميز البيانات الفئوية encoding categorical data، حتى أتمكن من أخذها في الاعتبار أيضًا.

أهمية الميزة

```
#One hot encoding
surface=pd.get_dummies(df["Surface"], prefix='surface_')
df = pd.concat([df,surface],axis=1)
df.drop(columns='Surface', inplace=True)

hand=pd.get_dummies(df["hand"], prefix='hand_')
df = pd.concat([df,hand],axis=1)
df.drop(columns='hand', inplace=True)

playing_round=pd.get_dummies(df["Round"])
df = pd.concat([df,playing_round],axis=1)
df.drop(columns='Round', inplace=True)

df["Court"].replace(to_replace=['Outdoor','Indoor'],value=[1,0],
inplace=True)
```

الآن بعد أن أصبح لدينا جميع البيانات ذات الصلة بتنسيق رقمي، حان الوقت لاتخاذ قرار بشأن الميزات التي سنستخدمها للنموذج. هناك عدة طرق للقيام بذلك، لكنني سأستخدم إحدى أسهل الطرق وأبسطها لأن هذا قد يكون كافيًا على الأرجح.

سأستخدم نموذج الغابة العشوائية RandomForestClassifier من sklearn، والذي يمكن أن يعمل كنموذج تنبؤي في حد ذاته، ولكنه يصنف الميزات داخليًا أيضًا من حيث الأهمية. لهذا السبب اخترت هذه الطريقة. مرة أخرى، هناك طرق أكثر بكثير للقيام بذلك وربما تكون أفضل بكثير.

```
df.dropna(subset=["minutes", "ht"], inplace=True)
df.fillna(0, inplace=True)
df.replace([np.inf, -np.inf], 0, inplace=True)

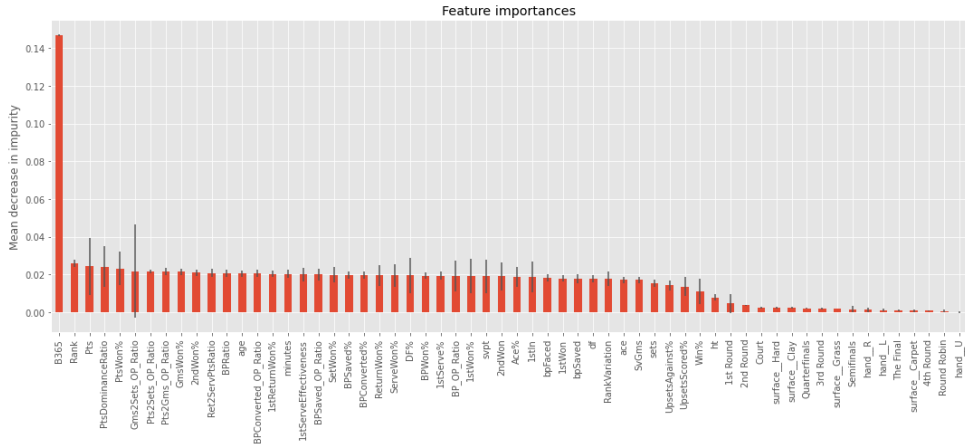
feature_names = [feature for feature in df.drop(columns=["Win", "name",
"year", "month"]).columns]
forest = RandomForestClassifier(random_state=0)
forest.fit(df.drop(columns=["Win", "name", "year", "month"]), df[["Win"]])

importances = forest.feature_importances_
std = np.std([
```

```
    tree.feature_importances_ for tree in forest.estimators_], axis=0)

forest_importances = pd.Series(importances,
index=feature_names).sort_values(ascending=False)

plt.rcParams["figure.figsize"] = [15,7]
fig, ax = plt.subplots()
forest_importances.plot.bar(yerr=std, ax=ax)
ax.set_title("Feature importances")
ax.set_ylabel("Mean decrease in impurity")
fig.tight_layout()
```



تم ترتيب هذا المخطط من حيث الأهمية. نرى أن الاحتمالات هي الأكثر أهمية على الإطلاق، ثم إن الغالبية العظمى لها نفس الأهمية تقريبًا. يسعدني أن 7 من أفضل 10 هي ميزات متقدمة ومبتكرة.

الميزات ذات الأهمية الأقل هي تلك التي تتوافق مع الميزات ذات الترميز واحد ساخن one-hot encoded features. وهذا منطقي ولا ينبغي أن يكون هذا سببًا لتركها خارجًا. ومع ذلك، أخطط لإسقاط الأعمدة المقابلة للجولة التي يتم لعبها، فأنا لا أحب هذه الميزة.

سأتخلى أيضًا عن الارتفاع.

بصرف النظر عن ذلك، أعتقد أن لدينا ما يكفي بالفعل! البيانات التي تمت ملاحظتهافي تصور الارتباط لدينا وما يوفره نموذج الغابة العشوائية هذا مفيدة وتساعدنافي تحديد الإحصائيات المهمة.

الاستنتاج

لقد أعطانا هذا المشروع بعض الرؤى المفيدة فيما يتعلق بالإحصائيات والميزات التي قد تكون الأكثر أهمية عندما يتعلق الأمر بالتنبؤ بالفائزفي مباراة التنس. ركزت معظم المهام على إعداد البيانات ومعالجتها، لكننا استخدمنا أيضًا بعض التصورات المرئية لفهم بياناتنا بشكل أفضل والحصول على النتائج التي كنا نبحث عنها.

احتمالات Odds هي الميزة الأكثر أهمية بلا شك. ومن المنطقي أن تكون نسبة احتمالات الفوزفي المراهنات صحيحةفي حوالي 70% من الوقت! كما فوجئت بأهمية ترتيب اللاعب ونقاطه. استنتجت من هذا أن نقاط ATP قوية ومهمة، ومحسوبة بشكل جيد.

وبصرف النظر عن ذلك، فقد رأينا بالفعل بعض النسب الأكثر تقدمًا وكيف ترتبط بفرص اللاعبفي الفوز، بالإضافة إلى بعض الإحصائيات الأكثر تقدمًا الأخرى.

يبدو أن العمر عامل مهم، أكثر من العديد من الميزات الأخرى التي قد يعتقد شخص ما أنها مهمة (مثل نسبة الآس، والأخطاء المزدوجة...).

أخيرًا، أود أن أسلط الضوء على دهشتي عندما وجدت أن نسبة الفوز خلال آخر 30 مباراة تبدو ضئيلة. كما أن الإحصائيات المتعلقة بالانزعاج ليست مهمة أيضًا.

المصدر:

https://www.kaggle.com/code/polmarinm/portfolio-project-data-analysis-on-tennis-stats#3.-THE-DATA

https://www.kaggle.com/code/chkrdhr/tennis-player-data-analysis

22) تحليلات دوري الهوكي الوطني NHL باستخدام بايثون
NHL Analytics With Python

كندا. الهوكي. قد يقول البعض إنهما نفس الشيء، ولكنني لم أكن من مشجعي الرياضة حقًا.في المناسبات النادرة التي يصل فيها فريق Maple Leafs إلى التصفيات النهائية، أرتدي بعض الملابس الزرقاء وأذهب مع أصدقائي لانتظار خيبة الأمل الحتمية، ولكن عندما يبدأونفي الحديث عن اللاعبين وكيف أدوا خلال الموسم، أبدأفي الشعور ببعض البعد. لذا،في محاولة للحصول على فهم أفضل للعبة، قررت ارتداء قبعة تحليل البيانات والغوص العميقفي بعض بيانات لاعبي دوري الهوكي الوطني NHL.

كان أول شيء قمت به هو التحقق لمعرفة نوع البيانات المتاحة، يا إلهي لم أشعر بخيبة أمل! يوجدفي الواقع واجهة برمجة تطبيقات بيانات NHL (NHL data API) غير موثقة تحتوي على كل شيء من جداول الفريق والقوائم إلى كل تسديدة تم التقاطها على الجليد بمافي ذلك المواقع واللاعبين المشاركين وفي أي وقت حدث ذلك. بالنسبة لأولئك الذين يشعرون بالفضول، قام شخص رائع بإنشاء صفحة GIT مع ملخص لاستدعاءات واجهة برمجة التطبيقات API calls(وثائق واجهة برمجة تطبيقات NHL). مع هذا، كان لدي كل ما أحتاجه لسحب البيانات والبدءفي إجراء بعض تحليلات اللاعبين.

سحب بيانات واجهة برمجة تطبيقات NHL

يتم إجراء جميع تحليلاتي باستخدام بايثون، ولتسهيل الأمر، أستخدم Kaggle كمنصة برمجة. إذا لم تكن قد قمت بفحص Kaggle، فيجب عليك فعل ذلك حقًا. لديهم مجموعة من مجموعات البيانات المفتوحة والمسابقات وطريقة لمشاركة وتخزين دفاتر الملاحظات الخاصة بك (Jupyter). كل التعليمات البرمجية التي سأعرضها هنا متاحة للجمهور من حسابي على Kaggle (انظر الأسفل للمراجع) لذا لا تترددفي استكشافها وتقسيمها. نظرًا لأن مجموعة البيانات عميقة للغاية، فقد قررت سحب بيانات أحداث اللعبة فقط من موسم 2019–2020 لهذا التحليل. يتيح لي هذا الوصول إلى جميع الألعاب التي لعبت حتى الآن، بمافي ذلك جميع أحداث الجليد (التسديدات shots والضربات hits والمواجهات المباشرة faceoffs وما إلى ذلك).

وكما هو الحال دائمًا، فإن أول شيء يجب القيام به هو تحميل الحزم وتهيئة المتغيرات. سترى هنا أنني أستخدم طلبات للحصول على البيانات من واجهة برمجة التطبيقات وpickle لحفظ البيانات لاستخدامها لاحقًا.

```
import requests
import pickle# Set up the API call variables
game_data = []
year = '2019'
```

```
season_type = '02'
max_game_ID = 1290
```

يتيح لنا تنسيق واجهة برمجة التطبيقات (API) إدخال سنة الموسم واختيار نوع اللعبة مثل ما قبل الموسم/المنتظم regular/التصفيات playoff (02 هو الموسم المنتظمفي هذا المثال) وأخيرًا max_game_ID. هذا هو الحد الأقصى لعدد أحداث اللعبة game eventsفي عام معين.

حان الوقت لسحب البيانات وتخزين البيانات بتنسيق jsonفي قائمة.

```
# Loop over the counter and format the API call
for i in range(0,max_game_ID):
    r = requests.get(url='http://statsapi.web.nhl.com/api/v1/game/'
        + year + season_type +str(i).zfill(4)+'/feed/live')    data =
r.json()
    game_data.append(data)
```

الآن بعد تخزين كل هذه البيانات الرائعةفي قائمة، يمكننا حفظها كملف pickle. والسبب وراء قيامي بهذا هو أنني أستطيع تحميلها على Kaggle وإتاحتها لأي شخص يرغبفي إجراء بعض التحليلات. كما تسمح ملفات pickle بتخزين الكائنات ولديها أوقات تحميل سريعة جدًا.

```
with open('./'+year+'FullDataset.pkl', 'wb') as f:
        pickle.dump(game_data, f, pickle.HIGHEST_PROTOCOL)
```

وبهذا أصبح لدينا ملف جديد يسمى FullDataset.pkl2019 والذي نحفظه كمجموعة بيانات على Kaggle. يمكنك التحقق من المرجعفي الأسفل للحصول على رابط لمجموعة البيانات الخاصة بي.

الآن حان وقت المرح! هناك الكثير من الأشياء التي يمكنك القيام بها بهذا، ولكن ما أردت القيام به هو إلقاء نظرة على كفاءة التصويب shooting efficiency لأي لاعبفي مواقع مختلفة من الجليد ومقارنتها بمعدل النجاح المتوسط للدوري بأكمله.

تحليل بيانات تسديد اللاعبين

باستخدام بيانات NHL، أردت أن أتعرف على الأماكن التي يسدد فيها اللاعبون أكثر تسديداتهم كفاءة (نسبة الأهداف Goal/إجمالي التسديدات Total Shot) مقارنة بمتوسطات الدوري. للبدء، دعنا نستورد جميع الحزم المطلوبة. لاحظ أنني أستخدم matplotlib لجميع مخططاتي. كما قمت بتثبيت pillow لاستيراد بيانات الصورة.

```
import numpy as np
import pandas as pd
import pickle
import matplotlib
import matplotlib.pyplot as plt
color_map = plt.cm.winter
from matplotlib.patches import RegularPolygon
import math
from PIL import Image# Needed for custom colour mapping!
from matplotlib.colors import ListedColormap,LinearSegmentedColormap
import matplotlib.colors as mcolors
```

بالنسبة لجميع مخططاتي، سأستخدم خريطة ألوان مخصصة باستخدام طريقة matplotlib ListedColormap. ولأنني أريد تلوين القيم الإيجابية بشكل مختلف عن السلبية، فإنني أقوم بعمل تعيينين للألوان.

```
c = mcolors.ColorConverter().to_rgb()
positive_cm = ListedColormap([c('#e1e5e5'),c('#d63b36')])
negative_cm = ListedColormap([c('#e1e5e5'),c('#28aee4')])
```

الآن بعد أن استقرينا على ذلك، فلنبدأفي تحميل ملف بيانات pickle للموسم العادي لعام 2019 والذي يحتوي على جميع بيانات الأحداث لكل مباراة. لاحظ أننا نقوم بتحميل ملف pickle الذي أنشأناه مسبقًا.

```
with open('../input/nhl-data/2019FullDataset.pkl', 'rb') as f:
        game_data = pickle.load(f)
```

حساب متوسط نسبة التسديد

مع تحميل البيانات، أريد أولاً حساب متوسط نسبة التسديدفي الدوري league average shooting percent عند كل نقطة على الجليد. تأتي البيانات باستخدام الأحداث وكائنات الإحداثياتفي القاموس.في تحليلنا، نريد فقط إلقاء نظرة على أحداث نوع "التصويب Shot" و"الهدف Goal ".

فيما يلي التقسيم خطوة بخطوة:

أولاً، ننشئ قاموسًا لاحتواء جميع إحداثيات بيانات التسديد والهدف للدوري بأكمله.

```
# Do some dictionary initialisation to hold our cleaned and condensed
league data
league_data = {};league_data['Shot'] = {};
league_data['Shot']['x'] = [];
league_data['Shot']['y'] = [];league_data['Goal'] = {};
league_data['Goal']['x'] = [];
league_data['Goal']['y'] = [];
```

نريد فقط الاحتفاظ بالأحداثفي البيانات الخاصة بالتسديد والأهداف.

```
event_types = ['Shot','Goal']
```

وأخيرًا، نراجع كل لعبة تم لعبها ونستخرج المعلومات ذات الصلة ونضعهافي قاموسنا.

```
# First loop over the elements of game_data. Each one of these is an NHL
game and contains all of the game event data.
for data in game_data:
# It is possible that the game data is not assigned to the data
   set, so to handle this we look for the key 'liveData' which
   contains all of the data we are looking for, otherwise we
   continue
   if 'liveData' not in data:
        continue
# Drilling down into the dataset to extract the play by play
   information for the game
   plays = data['liveData']['plays']['allPlays']

    for play in plays:
```

```
# For each play
        for event in event_types:  # For each event (Shot,Goal)
# If the play contains one of the events
            if play['result']['event'] in [event]:
# If the event contains coordinates
              if 'x' in play['coordinates']:
# Save the coordinates to the growing list
                  league_data[event]
                       ['x'].append(play['coordinates']['x'])
                    league_data[event]
                       ['y'].append(play['coordinates']['y'])
```

الآن بعد أن حصلنا على بيانات الدوري، يمكننا القيام بنفس الأشياء للاعب معين. الفرق الوحيد هو أننا سنقوم بالتصفية حسب نوع الحدث الفرعي Shooter عند إجراء استخراج البيانات.

بعد التشاور مع بعض مشجعي الهوكي، قيل لي أن Alex Ovechkin هو حالة اختبار رائعة. لديه مجال مميز للغاية ينشط فيه ويجب أن نكون قادرين على التحقق من صحة الأشياء بناءً على ذلك.

استخراج بيانات اللاعب مشابه جداً لاستخراج بيانات الدوري، لذا سأشير فقط إلى التغييرات هنا:

```
# Initialise the player dictionary
full_name = 'Alex Ovechkin'
player data = {};player data['Shot'] = {};
player_data['Shot']['x'] = [];
player_data['Shot']['y'] = [];player_data['Goal'] = {};
player_data['Goal']['x'] = [];
player_data['Goal']['y'] = [];
```

الآن نقوم بنفس العملية الأساسية كمافي السابق ولكن بالبحث عن بيانات اللاعب.

```
# Same code as before
...for play in plays:
   if 'players' in play:
      for player in play['players']:
         if player['player']['fullName'] in [full_name]
            and player['playerType'] in ["Shooter","Scorer"]:
          for event in event_types:

...
```

في القسم الغامق أعلاه، يمكنك أن ترى أننا نقوم الآن بتصفية اللاعب باعتباره الرامي Shooter أو المسجل Scorer. تتضمن كل لعبة من شارك فيها، لذا يمكننا ببساطة أن ننظر لمعرفة ما إذا كان Ovechkin مدرجًا باعتباره الرامي أو المسجل. وبهذا يمكننا تحليل بعض الأرقام وإجراء بعض التحليلات الرائعة.

حساب الإحصائيات الأساسية

قبل أن نرسم بيانات الموقع، أردت حساب الإحصائيات عالية المستوى للاعب مقارنة بمتوسطات الدوري.

```
# Get the total number of shots made by the player
player_total_shots = len(player_data['Shot']['x']) +
   len(player_data['Goal']['x'])
```

```
# Find the players goal score percentage
player_goal_pct = len(player_data['Goal']['x'])/player_total_shots
# Find the total number of shots taken in the league
league_total_shots = len(league_data['Shot']['x']) +
   len(league_data['Goal']['x'])
# Get the league percentage
league_goal_pct = len(league_data['Goal']['x'])/league_total_shots
# Calculate the spread of the SOG (Shots on Goal) %
PL_e_spread = player_goal_pct-league_goal_pct
```

```
Player Total Shots: 315
Player Total Goals: 49
Player SOG %: 0.15555555555555556
League Total Shots: 68499
League SOG %: 0.0970525117154995
Player Vs League SOG% Spread: 0.05850304384005606
```

وهنا لدينا النتيجة الأولى. ويمكننا مقارنة هذه النتيجة بالأرقام الرسمية، وهي متطابقة.

يمكننا أن نرى أن نسبة 5.85% التي حققها Ovechkin تعني أنه هداف كفوء.

ولكن هل هذا صحيحفي جميع النقاط على الجليد؟ هل لديه جانب مهيمن، أو أي نقطة ضعف؟ الآن يمكننا الانتقال إلى تحليل الموقع.

تحليل موقع التسديدات ورسمها

ما سنفعله أولاً هو إعداد شبكة تصنيف. أريد أن أفهم أين يتم التقاط التسديدات على الجليد، لكنني لا أريد رؤية كل التسديدات الفردية. من خلال أخذ المتوسط المكاني، يمكننا إنشاء تمثيل أكثر وضوحًا وبصريًا. تمتد بيانات الموضع من واجهة برمجة التطبيقات من:

- X: -100 to 100 (meters)
- Y: -42.5 to 42.5 (meters)

بالنسبة للتصنيف، نستخدم مخططات سداسية hex plots من matplotlib لاستخراج بيانات التصنيف الخام وسنستخدم المستطيلات المرسومة drawn rectangles (مرة أخرى matplotlib) للصور المرئية النهائية.

للبدء، نحدد أبعاد الشكل وحجم الشبكة:

```
# To keep the aspect ration correct we use a square figure size
xbnds = np.array([-100.,100.0])
ybnds = np.array([-100,100])
extent = [xbnds[0],xbnds[1],ybnds[0],ybnds[1]]
# We are going to bin in 30 unit increments.  It is fun to play with this!
gridsize= 30;mincnt=0
```

بعد ذلك، سنجد كفاءة الدوريفي كل موقع على الجليد. للقيام بذلك، نستدعي طريقة hexbin ونستخرج رأس الموقع وبيانات العد.

شيء واحد يجب ملاحظته هو أنه نظرًا لأن المسجل لا يسجل أبدًا على شباكه الخاصة، فيجب التأكد من قلب المواقع السلبية لتمثل دائمًا الجانب المهاجم.

كل هذا لأن كل فترة يقوم اللاعب بتبديل الجانبين ونظام الإحداثيات ثابت.

```
# First concatenate the arrays for x and y league data
league_x_all_shots = league_data['Shot']['x']
   + league_data['Goal']['x'];
league_y_all_shots = league_data['Shot']['y']
   + league_data['Goal']['y']
# Perform the coordinate flipping!
league_x_all_shots_normalized = [];
league_y_all_shots_normalized = []
# Enumerate the list so we can use the index for y also
for i,s in enumerate(league_x_all_shots):
    if league_x_all_shots[i] <0:
        league_x_all_shots_normalized.append(-league_x_all_shots[i])
        league_y_all_shots_normalized.append(-league_y_all_shots[i])
    else:
        league_x_all_shots_normalized.append(league_x_all_shots[i])
        league_y_all_shots_normalized.append(league_y_all_shots[i])

# Do the same treatment for the goals
league_x_goal_normalized = [];
league_y_goal_normalized=[]
for i,s in enumerate(league_data['Goal']['x']):
    if league_data['Goal']['x'][i] <0:
       league_x_goal_normalized.append(-league_data['Goal']['x'][i])
       league_y_goal_normalized.append(-league_data['Goal']['y'][i])
    else:
       league_x_goal_normalized.append(league_data['Goal']['x'][i])
       league_y_goal_normalized.append(league_data['Goal']['y'][i])
```

رائع. الآن حان وقت المال! استدعاء مخطط hexbin واستخراج الأعداد والمواقع. هذا الجزء طويل بعض الشيء، ولكن ما عليك سوى متابعة التعليقات وستصبح الأمور واضحة تمامًا.

```
# First we will used the hexbin function to simply bucket our shot data
into basically a 2D histogram
league_hex_data = plt.hexbin(league_x_all_shots_normalized,
   league_y_all_shots_normalized,gridsize=gridsize,
   extent=extent,mincnt=mincnt,alpha=0.0)
# Now we extract the bin coordinates and counts
league_verts = league_hex_data.get_offsets();
league_shot_frequency = league_hex_data.get_array();
# Do the same thing for the goal data
league goal hex data =  plt.hexbin(league x goal normalized,
   league_y_goal_normalized,gridsize=gridsize,
   extent=extent,mincnt=mincnt,alpha=0.0)
# Since the grid is the same we can use a shared bin coordinate set from
the above. So here we just get the counts
league_goal_frequency = league_goal_hex_data.get_array();
```

الآن بعد أن حصلنا على مواقع وعدد بيانات التسديدات، سنحاول عرضها بطريقة مفيدة. أولاً، سنحمل صورة نموذجية مصغرة لنصف حلبة NHL. ثم سنتأكد من تغيير حجم إحداثياتنا لتتناسب مع حجم الصورة وبالتالي الحصول على تمثيل دقيق لمكان إطلاق التسديدات على الجليد.

```
# Using matplotlib we create a new figure for plotting
fig=plt.figure(figsize=(10,10))
ax = fig.add_subplot(111)
# Clean up the figure to be completely blank
ax.set_facecolor("white")
fig.patch.set_facecolor("white")
fig.patch.set_alpha(0.0)
# Remove the labelling of axes
ax.set_xticklabels(labels = [''], fontsize = 18,
   alpha = .7,minor=False)
ax.set_yticklabels(labels = [''], fontsize = 18,
   alpha = .7,minor=False)
# Using pillow to get the rink image and extract the image size
I = Image.open('../input/nhl-images/half.png')
ax.imshow(I);width, height = I.size
```

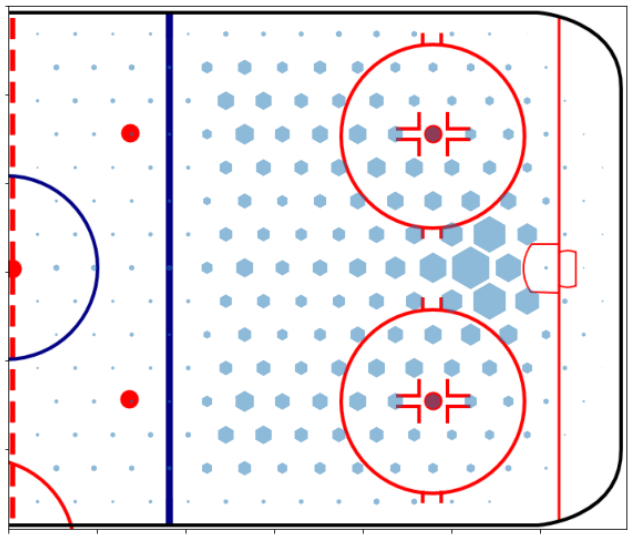
في هذه المرحلة، يجب أن يكون لدينا صورة لحلبة هوكي يتم رسمها كصورة matplotlib. بعد ذلك، نريد تحديد بعض عوامل القياس والإزاحات لمحاذاة نظام إحداثيات الصورة والبيانات.

```
# Calculate the scaling factor and offset (trial and error)
scalingx=width/100-0.6;
scalingy=height/100+0.5;
x_trans=33;
y_trans=height/2
# We will want to scale the size of our hex bins with the image so we
calculate a "radius" scaling factor here
S = 3.8*scalingx;
```

أخيرًا أصبحنا جاهزين للرسم. يسمح برنامج Matplotlib بإضافة عناصر التصحيح patch elementsفي شكل مضلع polygon shape معين. لذا سنستخدم عناصر سداسية لإضافة تصحيحات فوق صورة الحلبة.

```
# Loop over the locations and draw the hex
for i,v in enumerate(league_verts):   # Ignore empty locations
   if league_shot_frequency[i] < 1:continue

   # Normalize the shot frequency data between 0-1
   scaled_league_shot_frequency =
      league_shot_frequency[i]/max(league_shot_frequency)   # Scale the
hexagon size based on shot frequency
   radius = S*math.sqrt(scaled_league_shot_frequency)   # Finally we will
plot the hexagon including the scaling and
   translations we found earlier
   hex = RegularPolygon((x_trans+v[0]*scalingx,
      y_trans-v[1]*scalingy),numVertices=6, radius=radius,
      orientation=np.radians(0),alpha=0.5, edgecolor=None)
ax.add_patch(hex)
```

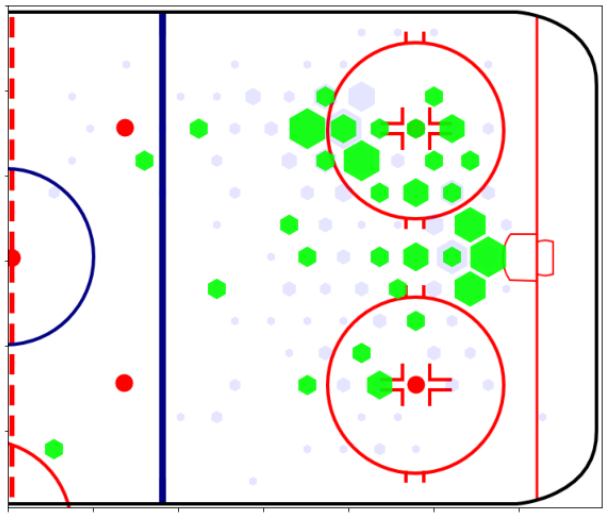


يمكنك أن ترى أن التسديداتفي المتوسط متماثلة إلى حد كبير وتحدث معظمها أمام الشبكة. يمكنك أيضًا أن ترى انخفاضًا كبيرًا عند الخط الأزرق وهو أمر منطقي أيضًا.

الآن دعنا نفعل نفس الأشياء لبيانات اللاعب. الفرق الوحيد هو أن جميع المتغيرات لديها بادئة "الدوري league" بدلاً من "اللاعب player ". على سبيل المثال:

```
player_x_all_shots = player_data['Shot']['x']
  + player_data['Goal']['x'];
player_y_all_shots = player_data['Shot']['y']
  + player_data['Goal']['y']
```

الآن بدلاً من مجرد رسم التسديدات، سنعرض أيضًا الأهداف باللون الأخضر.



رائع جدًا! يمكننا أن نرى أنه نشط جدًا على الجانب الأيسر فوق الدائرة. بعد التشاور مع أصدقائيفي لعبة الهوكي، اتضح أن هذا هو مكانه حقًا! ولكن الآن يجب أن ننظر إلى كفاءته على الجليد. هل هذا مكانه لأنه يسدد كثيرًا هناك؟

للقيام بذلك، سأجعل حجم السداسي يمثل تردد التسديد واللون يمثل كفاءته.

```
# Get some lists initialised
league_efficiency = []
player_efficiency = []
relative_efficiency = []
# Looping over the league shots (which are the same in length as player)
for i in range(0,len(league_shot_frequency)):
# We will only look at positions on the ice where the player or
    league had more than two shots during the season
    if league_shot_frequency[i]<2 or player_shot_frequency[i]<2:
        continue
# Calculate the efficiencies
league_efficiency.append(
   league_goal_frequency[i]/league_shot_frequency[i])

player_efficiency.append(
   player_goal_frequency[i]/player_shot_frequency[i])

# And the relative efficiency
relative_efficiency.append(
   (player_goal_frequency[i]/player_shot_frequency[i]-
   (league_goal_frequency[i]/league_shot_frequency[i]))
# Keep track of the max so we can scale the colour and radius of the hex
plot after
max_league_efficiency = max(league_efficiency)
max_player_efficiency = max(player_efficiency)
max_relative_efficiency = max(relative_efficiency)
min_relative_efficiency = min(relative_efficiency)
```

وأخيرًا، أصبحنا مستعدين للمرحلة النهائية المتمثلةفي رسم الكفاءة النسبية للتسديداتفي موقع معين.

```
# Loop over the locations and draw the hex
for i,v in enumerate(player_verts):
# Here we will only include locations where the player made at
    least on shot.  We will adjust this later for plotting.
    if player_shot_frequency[i] < 1:continue

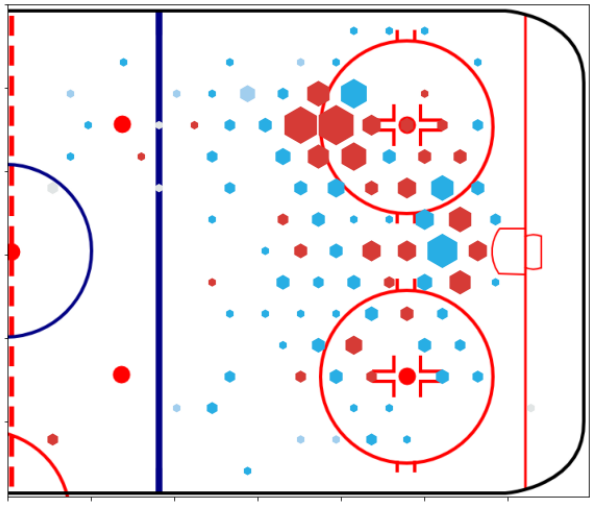
    # Scaling the frequencies
    scaled_player_shot_frequency =
        player_shot_frequency[i]/max(player_shot_frequency)
# Calculate a radius of the hex
    radius = S*math.sqrt(scaled_player_shot_frequency)

    # Find the player efficiency and relative at this point on the
    ice.
    player_efficiency =
        player_goal_frequency[i]/player_shot_frequency[i]
    league_efficiency =
        league_goal_frequency[i]/league_shot_frequency[i]
# This is what we were after the whole time!
    relative_efficiency = player_efficiency - league_efficiency

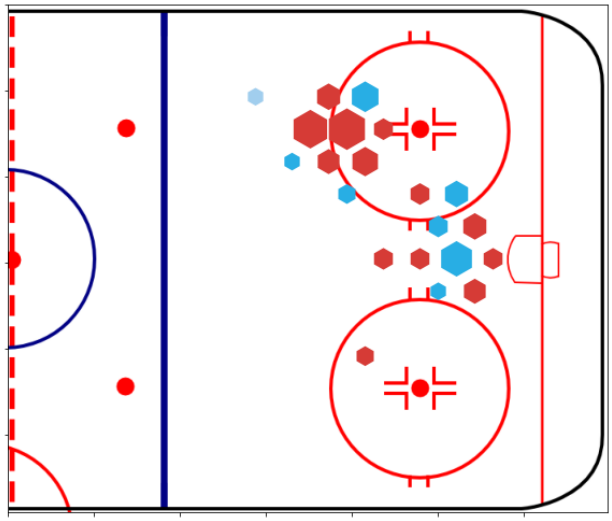
   # Since there can be positive and negative efficiencies
   (relative) we colour the more efficient locations red and the
```

```
    less blue.
    if relative_efficiency>0:
         colour = positive_cm(math.pow(relative_efficiency,0.1))
    else:
         colour = negative_cm(math.pow(-relative_efficiency,0.1))

    # And finally we plot!
    hex = RegularPolygon((x_trans+v[0]*scalingx,
        y_trans-v[1]*scalingy),numVertices=6, radius=radius,
        orientation=np.radians(0),facecolor=colour,alpha=1,
        edgecolor=None)
    ax.add_patch(hex)
```



كخطوة أخيرة، يمكننا زيادة عتبة نقاط البيانات التي يجب النظر إليها. دعنا ننظر فقط إلى المواقع التي سدد فيها أكثر من 4 تسديداتفي الموسم.



يا له من أمر رائع! يبدو جيدًا الآن! يمكننا أن نرى بوضوح أن Ovechkin لاعب فعال للغاية على الجانب الأيسر، وهو أيضًا المكان الذي يسدد فيه معظم تسديداته. وبالمقارنة بمتوسط الدوري، فهو مذهل.

أحد الأشياء التي نجدها هي أنه أقل من المتوسط عندما يتعلق الأمر بتسديد التسديدات أمام المرمى مباشرةً (وهو أمر شائع جدًا بناءً على حجم السداسي). قد يكون هذا شيئًا يساعده على التحسن أو استخدامه ضده!

المراجع والاكواد:

- استخراج البيانات.
- مجموعة البيانات.
- تحليل البيانات.
- NHL API.

المصدر:

https://towardsdatascience.com/nhl-analytics-with-python-6390c5d3206d

Sports Analytics using Artificial Intelligence

Sports Data Analyzed and Explained using Artificial Intelligence

Dr. Alaa Taima