HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

KHOA CÔNG NGHỆ THÔNG TIN 1

BỘ MÔN LẬP TRÌNH PYTHON

BÀI BÁO CÁO BÀI TẬP LỚN MÔN LẬP TRÌNH PYTHON

Giảng viên hướng dẫn : **Kim Ngọc Bách**
Họ và tên sinh viên : **Nguyễn Vĩnh Hưng**
Mã sinh viên : **B23DCCE042**
Lớp : **D23DCCE06 – B**

Hà Nội 2024 – 2025

MỤC LỤC

LỜI MỞ ĐẦU

Báo cáo này trình bày chi tiết quy trình thu thập, xử lý và phân tích dữ liệu thống kê của các cầu thủ bóng đá tham gia giải Ngoại Hạng Anh mùa giải 2024 – 2025. Dữ liệu được thu thập từ nguồn uy tín FBREF, bao gồm tập hợp đầy đủ các chỉ số hiệu suất của từng cầu thủ.

Trong báo cáo, em mô tả rõ ràng phương pháp thu thập dữ liệu tự động sử dụng ngôn ngữ lập trình Python. Tiếp theo, quá trình tiền xử lý dữ liệu được trình bày, bao gồm các kỹ thuật xử lý dữ liệu bị khuyết thiếu và các giá trị không hợp lệ nhằm đảm bảo chất lượng và độ tin cậy của tập dữ liệu cho các bước phân tích tiếp theo.

Phần phân tích dữ liệu bắt đầu bằng việc xác định top ba cầu thủ có chỉ số cao nhất và thấp nhất đối với từng chỉ số thống kê riêng biệt. Đồng thời, báo cáo cũng xây dựng và áp dụng phương pháp tính toán nhằm đánh giá định lượng hiệu suất tổng thể của cầu thủ dựa trên các chỉ số thu thập được.

Quá trình trực quan hóa dữ liệu được thực hiện sau đó để biểu diễn các chỉ số và kết quả phân tích dưới dạng đồ thị, cung cấp cái nhìn trực quan và dễ hiểu về đặc điểm của cầu thủ.

Trong bước phân tích nâng cao, thuật toán phân cụm K-Means được áp dụng để nhóm các cầu thủ thành các cụm dựa trên sự tương đồng về chỉ số thống kê, giúp phát hiện các mẫu (pattern) tiềm ẩn trong dữ liệu. Phương pháp Phân tích (PCA) cũng được sử dụng để giảm chiều dữ liệu, hỗ trợ việc phân tích và trực quan hóa trong không gian chiều thấp hơn.

Cuối cùng, báo cáo tích hợp thêm dữ liệu về giá trị chuyển nhượng của cầu thủ được thu thập từ nguồn Transfermarkt. Tập dữ liệu tổng hợp này sau đó được sử dụng để xây dựng các mô hình dự đoán từ thư viện Scikit-learn (Sklearn) nhằm ước tính giá trị chuyển nhượng tiềm năng của cầu thủ dựa trên các chỉ số hiệu suất. Hiệu quả và khả năng dự đoán của các mô hình này được đánh giá một cách khách quan thông qua việc sử dụng các chỉ số đánh giá mô hình tiêu chuẩn.

Kết quả được trình bày trong báo cáo có thể không hoàn toàn trùng khớp với kết quả thu được khi trực tiếp thực thi đoạn mã nguồn, do có thể tồn tại sự khác biệt về môi trường thực thi, phiên bản phần mềm, dữ liệu đầu vào, hoặc các yếu tố ngẫu nhiên trong quá trình chạy chương trình.

I, Bài 1

1. Bước lấy dữ liệu cầu thủ trực tiếp từ FBREF

1.1.Import các thư viện cần thiết

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup, Comment
import pandas as pd
import time
```

- **SELENIUM: Điều khiển trình duyệt web tự động (ở đây là Chrome).**
- **BEAUTIFULSOUP: Dùng để phân tích cú pháp HTML, đặc biệt để trích xuất dữ liệu từ phần HTML bị comment.**
- **COMMENT: Dùng để tìm các phần nội dung bị ẩn trong HTML dưới dạng <!-- comment -->.**
- **PANDAS: Thư viện xử lý bảng dữ liệu (dataframe).**
- **TIME: Dùng để tạm dừng (sleep) vài giây cho trang web tải xong.**

1.2. Hàm khởi tạo trình duyệt

```
def init_driver():
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--disable-gpu")
    chrome_options.add_argument("--no-sandbox")
    driver = webdriver.Chrome(options=chrome_options)
    return driver
```

- **Sử dụng Chrome ở chế độ headless để tiết kiệm tài nguyên (không hiển thị cửa sổ trình duyệt).**

- **Trả về một đối tượng driver dùng để truy cập trang web.**

1.3. Danh sách các URL và bảng cần thu thập

```python
urls = {
    "standard": ("https://fbref.com/en/comps/9/stats/Premier-League-Stats", "stats_standard"),
    "keepers": ("https://fbref.com/en/comps/9/keepers/Premier-League-Stats", "stats_keeper"),
    "shooting": ("https://fbref.com/en/comps/9/shooting/Premier-League-Stats", "stats_shooting"),
    "passing": ("https://fbref.com/en/comps/9/passing/Premier-League-Stats", "stats_passing"),
    "gca": ("https://fbref.com/en/comps/9/gca/Premier-League-Stats", "stats_gca"),
    "defense": ("https://fbref.com/en/comps/9/defense/Premier-League-Stats", "stats_defense"),
    "possession": ("https://fbref.com/en/comps/9/possession/Premier-League-Stats", "stats_possession"),
    "misc": ("https://fbref.com/en/comps/9/misc/Premier-League-Stats", "stats_misc")
}
```

Mỗi mục trong url chứa:

- **key: tên loại thống kê (standard, keepers, shooting, ...)**
- **url: địa chỉ trang web chứa bảng dữ liệu.**
- **table_id: id của bảng HTML cần trích xuất (để tìm đúng bảng trong nhiều bảng).**

1.4. Hàm lấy bảng thống kê từ HTML bị comment

```python
def get_stats_table_selenium(driver, url, table_id):
    driver.get(url)
    time.sleep(3)
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    comments = soup.find_all(string=lambda text: isinstance(text, Comment))

    for comment in comments:
        if table_id in comment:
            comment_soup = BeautifulSoup(comment, 'html.parser')
            table = comment_soup.find("table", {"id": table_id})
            if table:
                df = pd.read_html(str(table))[0]
                df = df[df[df.columns[0]] != df.columns[0]]
                return df
    print(f"❌ Không tìm thấy bảng với id='{table_id}' trong {url}")
    return None
```

- **def get_stats_table_selenium(driver, url, table_id) : Hàm sẽ nhận vào driver (trình điều khiển Selenium), url (địa chỉ trang web cần thu thập dữ liệu), table_id (ID của bảng HTML cần tìm).**
- **driver.get(url) : Mở trình duyệt Chrome (headless) đến URL.**
- **time.sleep(3) : Dừng 3 giây để đảm bảo trang web tải xong.**
- **soup = BeautifulSoup(driver.page_source, 'html.parser') : Lấy nội dung HTML của trang vừa mở (driver.page_source), dùng BeautifulSoup để phân tích HTML đó.**

- **comments = soup.find_all(string=lambda text: isinstance(text, Comment)) : Do đó phải tìm tất cả phần nội dung dạng <!-- ... --> trong trang, kết quả là một danh sách các đoạn comment.**

```
for comment in comments:
    if table_id in comment:
```

- **Với mỗi comment, kiểm tra xem nó có chứa chuỗi table_id (ví dụ "stats_standard") không.**

- **Nếu có, thì rất có thể đây là bảng cần tìm.**

```
comment_soup = BeautifulSoup(comment, 'html.parser')
table = comment_soup.find("table", {"id": table_id})
```

- **Dùng BeautifulSoup lần nữa để "giải mã" nội dung trong comment (coi như HTML thật).**

- **Tìm bảng có đúng id mong muốn.**

```
if table:
    df = pd.read_html(str(table))[0]
```

- **Chuyển bảng HTML thành bảng dữ liệu pandas DataFrame.**

- **read_html trả về một danh sách, nên lấy phần tử đầu tiên [0].**

- **df = df[df[df.columns[0]] != df.columns[0]] : Lọc loại bỏ dòng tiêu đề bị lặp(nếu có) bằng cách so sánh giá trị dòng với tiêu đề cột nếu dòng giống với tiêu đề loại bỏ.**

- **return df : cuối cùng trả về bảng dữ liệu**

```
print(f"❌ Không tìm thấy bảng với id='{table_id}' trong {url}")
return None
```

- **Thông báo lỗi nếu không tìm thấy bảng cần thiết trong toàn bộ comment HTML.**

1.5. Vòng lặp thu thập toàn bộ dữ liệu

```
driver = init_driver()
all_stats = {}

for key, (url, table_id) in urls.items():
    print(f"🔍 Đang thu thập {key}...")
    df = get_stats_table_selenium(driver, url, table_id)
    if df is not None:
        all_stats[key] = df
        print(f"✅ Thu thập xong {key}, {len(df)} dòng.")
    else:
        print(f"⚠️ Không thể thu thập dữ liệu {key}.")

driver.quit()
```

- **Duyệt qua từng mục trong urls.**
- **Gọi hàm get_stats_table_selenium để lấy bảng.**
- **Nếu có dữ liệu, lưu vào all_stats[key].**
- **driver.quit() : Sau khi hoàn tất, đóng trình duyệt để giải phóng tài nguyên.**

1.6. Mô phỏng kết quả thu được

```
{
  "standard": DataFrame,
  "keepers": DataFrame,
  ...
}
```

- **Ta có được 'key' là tên của bảng dữ liệu và 'DataFrame' là các bảng dữ liệu.**
- **Do có nhiều bảng nên em biểu diễn kết quả ở dạng mô phỏng như trên.**

2. Xử lý dữ liệu

2.1. Xử lý tên của các cột dữ liệu

```
for key in all_stats.keys():
    all_stats[key].columns = [' | '.join(col) if isinstance(col, tuple) else col
                              for col in all_stats[key].columns]
    print(all_stats[key].columns)
```

- **Chuyển tên cột dạng nhiều cấp (multi-index) thành tên cột dạng chuỗi đơn giản, để dễ xử lý hơn**

- **Chúng ta có được kết quả sau khi thực thi câu lệnh print trên như sau:**

```
Index(['Unnamed: 0_level_0 | Rk', 'Unnamed: 1_level_0 | Player',
       'Unnamed: 2_level_0 | Nation', 'Unnamed: 3_level_0 | Pos',
       'Unnamed: 4_level_0 | Squad', 'Unnamed: 5_level_0 | Age',
       'Unnamed: 6_level_0 | Born', 'Playing Time | MP',
       'Playing Time | Starts', 'Playing Time | Min', 'Playing Time | 90s',
       'Performance | Gls', 'Performance | Ast', 'Performance | G+A',
       'Performance | G-PK', 'Performance | PK', 'Performance | PKatt',
       'Performance | CrdY', 'Performance | CrdR', 'Expected | xG',
       'Expected | npxG', 'Expected | xAG', 'Expected | npxG+xAG',
       'Progression | PrgC', 'Progression | PrgP', 'Progression | PrgR',
       'Per 90 Minutes | Gls', 'Per 90 Minutes | Ast', 'Per 90 Minutes | G+A',
       'Per 90 Minutes | G-PK', 'Per 90 Minutes | G+A-PK',
       'Per 90 Minutes | xG', 'Per 90 Minutes | xAG',
       'Per 90 Minutes | xG+xAG', 'Per 90 Minutes | npxG',
       'Per 90 Minutes | npxG+xAG', 'Unnamed: 36_level_0 | Matches'],
      dtype='object')
```

2.2. Làm phẳng cột, chuẩn hóa tên, và gộp các bảng dữ liệu

```python
from functools import reduce

processed_dfs = []

for idx, (key, df) in enumerate(all_stats.items()):

    df.columns = ['_'.join(col) if isinstance(col, tuple) else col for col in df.columns]

    player_col = next((col for col in df.columns if "Player" in col), None)
    if player_col is None:
        print(f"⚠️ Bảng '{key}' không có cột Player.")
        continue

    df = df.rename(columns={player_col: "Player"})

    df = df.rename(columns={col: f"{key}_{col}" for col in df.columns if col != "Player"})

    df = df.drop_duplicates(subset="Player")

    processed_dfs.append(df)

if processed_dfs:
    merged_df = reduce(lambda left, right: pd.merge(left, right, on="Player", how="outer"), processed_dfs)
else:
    print("❌ Không có DataFrame nào hợp lệ để gộp.")
```

- **processed_dfs = [] : Tạo một danh sách rỗng để lưu các dataFrame sau khi đã được chuẩn hóa và xử lý**
- **for idx, (key, df) in enumerate(all_stats.items()) : lặp qua từng bảng trong all_stats.**
- **df.columns = ['_'.join(col) if isinstance(col, tuple) else col for col in df.columns] : làm phẳng cột nếu là MultiIndex**
- **player_col = next((col for col in df.columns if "Player" in col), None) : một số bảng có cột tên là 'Player', nhưng cũng có thể là**

'player_Player', 'standard_Player', ..., dòng này sẽ tìm cột đầu tiên có chữ "Player" trong tên và gán vào player_col

```
if player_col is None:
    print(f"⚠️ Bảng '{key}' không có cột Player.")
    continue
```

- **Bỏ qua bảng này vì không thể gộp được nếu không có tên cầu thủ.**
- **df = df.rename(columns={player_col: "Player"}) : đổi tên cột tìm được thành "Player" cho đồng nhất giữa các bảng.**
- **df = df.rename(columns={col: f"{key}_{col}" for col in df.columns if col != "Player"}) : Với các cột không phải "Player", thêm tiền tố là tên bảng, nhờ vậy khi gộp các bảng lại sẽ tránh trùng tên cột.**
- **df = df.drop_duplicates(subset="Player") : Xóa các dòng trùng nhau theo "Player"**
- **processed_dfs.append(df) : Thêm df sau xử lý vào danh sách processed_dfs.**
- **if processed_dfs: merged_df = reduce(lambda left, right: pd.merge(left, right, on="Player", how="outer"), processed_dfs) : Dùng reduce() để gộp tất cả bảng trong processed_dfs lại theo cột "Player", how='outer': kiểu gộp ngoài — giữ lại tất cả cầu thủ, ngay cả khi họ không có dữ liệu trong một vài bảng.**

2.3. Xử lý tuổi

```
merged_df['standard_Unnamed: 5_level_0 | Age'] = merged_df['standard_Unnamed: 5_level_0 | Age'].apply(
    lambda x: int(x.split('-')[0].strip()) if isinstance(x, str) and '-' in x else x)
```

- **Dòng code trên chuyển các giá trị dạng chuỗi có dấu `-` (như `"18-362"`) trong cột `'standard_Unnamed: 5_level_0 | Age'` thành số nguyên đại diện cho tuổi bắt đầu (ví dụ: `"18-362"` → `18`).**

2.4. Lựa chọn các cột dữ liệu phù hợp

```
feat = ['Player',
        'standard_Unnamed: 2_level_0 | Nation',
        'standard_Unnamed: 3_level_0 | Pos',
        'standard_Unnamed: 4_level_0 | Squad',
        'standard_Unnamed: 5_level_0 | Age',
        'standard_Playing Time | MP',
        'standard_Playing Time | Starts',
        'standard_Playing Time | Min',
        'standard_Performance | Gls',
        'standard_Performance | Ast',
        'standard_Performance | CrdY',
```

```
        'standard_Performance | CrdR',
        'standard_Expected | xG',
        'standard_Expected | xAG',
        'standard_Progression | PrgC',
        'standard_Progression | PrgP',
        'standard_Progression | PrgR',
        'standard_Per 90 Minutes | Gls',
        'standard_Per 90 Minutes | Ast',
        'standard_Per 90 Minutes | xG',
        'standard_Per 90 Minutes | xAG',
        'keepers_Performance | GA90',
        'keepers_Performance | Save%',
        'keepers_Performance | CS%',
        'keepers_Penalty Kicks | Save%',
        'shooting_Standard | SoT%',
        'shooting_Standard | SoT/90',
        'shooting_Standard | G/Sh',
        'shooting_Standard | Dist',
        'passing_Total | Cmp',
        'passing_Total | Cmp%',
        'passing_Total | TotDist',
        'passing_Short | Cmp%',
        'passing_Medium | Cmp%',
        'passing_Long | Cmp%',
        'passing_Unnamed: 26_level_0 | KP',
        'passing_Unnamed: 27_level_0 | 1/3',
        'passing_Unnamed: 28_level_0 | PPA',
        'passing_Unnamed: 29_level_0 | CrsPA',
        'passing_Unnamed: 30_level_0 | PrgP',
        'gca_SCA | SCA',
        'gca_SCA | SCA90',
        'gca_GCA | GCA',
        'gca_GCA | GCA90',
        'defense_Tackles | Tkl',
        'defense_Tackles | TklW',
        'defense_Challenges | Att',
        'defense_Challenges | Lost',
        'defense_Blocks | Blocks',
        'defense_Blocks | Sh',
        'defense_Blocks | Pass',
        'defense_Unnamed: 20_level_0 | Int',
        'possession_Touches | Touches',
        'possession_Touches | Def Pen',
        'possession_Touches | Def 3rd',
        'possession_Touches | Mid 3rd',
        'possession_Touches | Att 3rd',
        'possession_Touches | Att Pen',
        'possession_Take-Ons | Att',
        'possession_Take-Ons | Succ%',
        'possession_Take-Ons | Tkld%',
```

```
        'possession_Carries | Carries',
        'possession_Carries | PrgDist',
        'possession_Carries | PrgC',
        'possession_Carries | 1/3',
        'possession_Carries | CPA',
        'possession_Carries | Mis',
        'possession_Carries | Dis',
        'possession_Receiving | Rec',
        'possession_Receiving | PrgR',
        'misc_Performance | Fls',
        'misc_Performance | Fld',
        'misc_Performance | Off',
        'misc_Performance | Crs',
        'misc_Performance | Recov',
        'misc_Aerial Duels | Won',
        'misc_Aerial Duels | Lost',
        'misc_Aerial Duels | Won%']
len(feat)
```

- **feat = ['Player', 'standard_Unnamed: 2_level_0 | Nation', 'standard_Unnamed: 3_level_0 | Pos', ..., 'misc_Aerial Duels | Lost', 'misc_Aerial Duels | Won%'] : danh sách feat sẽ bao gồm đầy đủ cột dữ liệu theo yêu cầu của bài 1.**
- **len(feat) : trả về số lượng cột dữ liệu có trong mảng**

2.5. Chuẩn hóa kiểu dữ liệu

```
def convert_to_number(value):
    if isinstance(value, (int, float)):
        return value
    if isinstance(value, str):
        try:
            if '.' in value:
                return float(value)
            else:
                return int(value)
        except ValueError:
            return None
    return None


for col in feat[4:]:
  merged_df[col] = merged_df[col].apply(convert_to_number)
  print(f"{col} __ {merged_df[col].dtypes}")
```

- **Do kiểu dữ liệu ban đầu là kiểu 'Object' nên em sử dụng hàm convert_to_number để chuyển các giá trị trong các cột `feat[4:]`**

của `merged_df` sang kiểu số (`int` hoặc `float`). Nếu không chuyển được (ví dụ: chuỗi không phải số), sẽ gán `None`. Sau mỗi cột, in ra kiểu dữ liệu (`dtype`) của cột đó.

2.6. Lọc và xử lý các dữ liệu bị thừa và thiếu

```
merged_df = merged_df[merged_df['standard_Playing Time | Min'] > 90]
merged_df.sort_values(by='Player',ascending=True,inplace=True)
```

- **Chỉ giữ lại các cầu thủ chơi hơn 90 phút.**
- **Sắp xếp tên cầu thủ theo thứ tự từ điển.**

```
import re

clean_feat = [re.sub(r'_Unnamed: \d+_level_0', '', col) for col in feat]

merged_df.columns = [re.sub(r'_Unnamed: \d+_level_0', '', col) for col in merged_df.columns]
```

- **Xóa phần dư thừa như _Unnamed: X_level_0 trong tên cột.**

```
kep = ['keepers_Performance | GA90',
       'keepers_Performance | Save%',
       'keepers_Performance | CS%',
       'keepers_Penalty Kicks | Save%']

merged_df[kep] = merged_df[kep].fillna(0)
```

- **Đối với các cầu thủ không phải là thủ môn sẽ được điền '0' đối với các chỉ số trong list kep**

```
shoot = ['shooting_Standard | SoT%',
         'shooting_Standard | SoT/90',
         'shooting_Standard | G/Sh',
         'shooting_Standard | Dist']

mask = merged_df['standard | Pos'] == 'GK'
merged_df.loc[mask, shoot] = merged_df.loc[mask, shoot].fillna(0)
```

- **Đối với vị trí thủ môn sẽ không có dữ liệu của các cột trong list shoot nên sẽ được điền '0'.**

```
miss_val_col = []

for col in clean_feat:
  if merged_df[col].isna().sum() > 0:
    print(f"{col} __ {merged_df[col].isna().sum()}")
    miss_val_col.append(col)

for col in miss_val_col:
  merged_df[col] = merged_df[col].fillna(merged_df[col].mean())

merged_df[clean_feat].to_csv('results1.csv', index=False)
```

- **Tìm cột có giá trị bị thiếu và lưu vào list miss_val_col, sau đó điền các cột có giá trị bị thiếu bằng giá trị trung bình của cột đó**
- **merged_df[clean_feat].to_csv('results1.csv', index=False) : Xuất dữ liệu đã xử lý ra file 'results1.csv'**

II, Bài 2

1. Tạo file top_3.txt với 3 cầu thủ cao/ thấp nhất cho từng chỉ số

```
with open("top_3.txt", "w", encoding="utf-8") as file:
    for col in clean_feat[4:]:
        if col == 'Player':
            continue
        sorted_idx = merged_df[col].sort_values(ascending=False).index
        g = pd.DataFrame(merged_df.loc[sorted_idx[:3], ['Player', col]]).reset_index(drop=True)
        f = pd.DataFrame(merged_df.loc[sorted_idx[-3:], ['Player', col]]).reset_index(drop=True)
        g.index += 1
        f.index += 1

        file.write('---------------------------------------\n')
        file.write(f"Top 3 players with highest {col}:\n")
        file.write(g.to_string(index=True, header=False))
        file.write('\n')
        file.write(f"Top 3 players with lowest {col}:\n")
        file.write(f.to_string(index=True, header=False))
        file.write('\n')
        file.write('---------------------------------------\n\n')
```

1.1. Ghi vào file

```
with open("top_3.txt", "w", encoding="utf-8") as file:
    for col in clean_feat[4:]:
        if col == 'Player':
            continue
```

- **Mở file top_3.txt để ghi.**
- **Lặp qua các cột đặc trưng (trừ 4 cột đầu).**
- **Bỏ qua nếu cột là 'Player'.**
- **sorted_idx=merged_df[col].sort_values(ascending=False).index: lấy chỉ số sắp xếp giảm dần theo giá trị trong cột col.**

```
g = pd.DataFrame(merged_df.loc[sorted_idx[:3], ['Player', col]]).reset_index(drop=True)
f = pd.DataFrame(merged_df.loc[sorted_idx[-3:], ['Player', col]]).reset_index(drop=True)
g.index += 1
f.index += 1
```

- **Lấy top 3 cao nhất (g) và thấp nhất (f), kèm tên cầu thủ.**
- **Reset index và đánh lại chỉ số từ 1.**

```
file.write('----------------------------------------\n')
file.write(f"Top 3 players with highest {col}:\n")
file.write(g.to_string(index=True, header=False))
file.write('\n')
file.write(f"Top 3 players with lowest {col}:\n")
file.write(f.to_string(index=True, header=False))
file.write('\n')
file.write('----------------------------------------\n\n')
```

- **Ghi kết quả top 3 cao nhất và thấp nhất vào file theo định dạng dễ đọc**

2. Tính toán và thống kê cho từng đội

2.1. Lọc ra các đội cần thống kê

```
teams = merged_df['standard | Squad'].unique()
teams.sort()
teams = ['all'] + list(teams)
```

- **Lấy danh sách tất cả các đội trong cột 'standard | Squad'.**
- **Thêm 'all' để đại diện cho toàn bộ cầu thủ.**

2.2. Tính median, mean, std cho từng chỉ số và từng đội

```
rows = []

for team in teams:
    row = {'Team': team}

    for feat in clean_feat[4:]:
        if team == 'all':
            subset = merged_df[feat]
        else:
            subset = merged_df.loc[merged_df['standard | Squad'] == team, feat]

        row[f"Median of {feat}"] = round(subset.median(), 2)
        row[f"Mean of {feat}"] = round(subset.mean(), 2)
        row[f"Std of {feat}"] = round(subset.std(), 2)

    rows.append(row)

metrics_df = pd.DataFrame(rows)

metrics_df.to_csv('result2.csv', index=False)
```

- **Khởi tạo danh sách rows lưu các dòng kết quả thống kê theo đội.**
- **Với mỗi đội (hoặc 'all'), tạo 1 dòng mới.**
- **Lấy dữ liệu theo đội (subset), hoặc toàn bộ nếu 'all'.**
- **Tính trung vị, trung bình và độ lệch chuẩn, làm tròn 2 chữ số.**

- Thêm dòng dữ liệu vào danh sách kết quả.
- Chuyển danh sách kết quả sang DataFrame.
- Lưu thành file CSV.

3. Vẽ biểu đồ Histogram

3.1. Chọn các đặc trưng tấn công và phòng thủ

```
att_feat = ['shooting_Standard | SoT%', 'shooting_Standard | SoT/90', 'shooting_Standard | Dist']
def_feat = ['defense_Tackles | Tkl', 'defense_Tackles | TklW', 'defense_Challenges | Att']
```

3.2. Plot histogram bằng matplotlib

```
import matplotlib.pyplot as plt

for team in teams:
    fig, (ax_row1, ax_row2) = plt.subplots(2, 3, figsize=(12, 6))
    fig.suptitle(f'{team} - Attack vs Defense Histograms', fontsize=14)

    for idx in range(3):
        if team == 'all':
            data_att = merged_df[att_feat[idx]]
            data_def = merged_df[def_feat[idx]]
        else:
            data_att = merged_df.loc[merged_df['standard | Squad'] == team, att_feat[idx]]
            data_def = merged_df.loc[merged_df['standard | Squad'] == team, def_feat[idx]]

        ax_row1[idx].hist(data_att, bins=20, color='skyblue', alpha=0.7)
        ax_row1[idx].set_title(f'{att_feat[idx]}', pad=12)

        ax_row2[idx].hist(data_def, bins=20, color='lightcoral', alpha=0.7)
        ax_row2[idx].set_title(f'{def_feat[idx]}', pad=12)

    plt.tight_layout(rect=[0, 0, 1, 0.95])
    plt.show()
```

- Import thư viện vẽ biểu đồ.
- Vẽ biểu đồ gồm 2 hàng (mỗi hàng 3 biểu đồ).
- Gán tiêu đề theo đội.
- Lấy dữ liệu tấn công và phòng ngự tương ứng theo đội hoặc toàn bộ.
- Vẽ histogram (biểu đồ tần suất) cho mỗi chỉ số.
- Hàng 1 (ax_row1): tấn công, hàng 2 (ax_row2): phòng ngự.
- Điều chỉnh khoảng cách các biểu đồ.
- Hiển thị biểu đồ bằng câu lệnh plt.show()

4. Đánh giá đội có màn trình diễn tốt nhất

4.1. Chọn ra các tiêu chí đánh giá

```
team_only_df = metrics_df[metrics_df['Team'] != 'all']
stat_cols = [col for col in team_only_df.columns if col.startswith('Mean of')]
```

- **metrics_df chứa các thống kê trung bình, trung vị và độ lệch chuẩn của tất cả các đội + 1 dòng tổng hợp cho "all".**
- **Dòng này lọc ra chỉ các đội cụ thể để phân tích riêng từng đội, và lấy danh sách các cột liên quan đến trung bình (mean) của từng chỉ số**

4.2. In ra đội có giá trị trung bình cao nhất cho từng chỉ số

```
for col in stat_cols:
    max_row = team_only_df.loc[team_only_df[col].idxmax()]
    print(f" 🔹 {col}: {max_row['Team']} với {max_row[col]}")
```

4.3. Đánh giá cuối cùng

```
from sklearn.preprocessing import MinMaxScaler

scaler = MinMaxScaler()
team_scaled_df = team_only_df.copy()
team_scaled_df[stat_cols] = scaler.fit_transform(team_only_df[stat_cols])


team_scaled_df['Composite Score'] = team_scaled_df[stat_cols].mean(axis=1)

best_team_row = team_scaled_df.loc[team_scaled_df['Composite Score'].idxmax()]

print("\n🏆 Đội có performance tổng thể tốt nhất (dựa trên tất cả các chỉ số Mean):")
print(f" 🔹 Team: {best_team_row['Team']}")
print(f" 🔹 Composite Score: {best_team_row['Composite Score']:.3f}")
```

- **Sử dụng MinMaxScaler biến đổi giá trị mỗi cột về khoảng [0, 1], điều này giúp đảm bảo rằng không có chỉ số nào chiếm ưu thế vì có đơn vị hoặc phạm vi lớn hơn.**
- **Tính điểm trung bình của tất cả các chỉ số đã chuẩn hóa cho mỗi đội, đây là thước đo tổng quát để đánh giá performance tổng thể (cả tấn công lẫn phòng ngự).**
- **Tìm đội có Composite Score cao nhất → xem là đội có performance tốt nhất toàn giải đấu.**
- **Hiển thị tên đội và điểm tổng hợp của đội có performance tốt nhất.**

5. Kết quả

- **Đội có performance tổng thể tốt nhất (dựa trên tất cả các chỉ số Mean):**
- **Team: Liverpool**
- **Composite Score: 0.777**

III, Bài 3

1. Sử dụng thuật toán K – means để phân cụm cầu thủ

1.1. Chuẩn bị và chuẩn hóa dữ liệu

```
from sklearn.preprocessing import StandardScaler
from sklearn.cluster import KMeans

selected_columns = ['Player'] + clean_feat[4:]
df = merged_df.loc[:, selected_columns]

scaler = StandardScaler()
scaled_df = scaler.fit_transform(df.drop(columns=['Player']))
```

- **Chọn các cột gồm tên cầu thủ (Player) và các đặc trưng thống kê (clean_feat[4:]).**
- **Tạo DataFrame df chỉ chứa các cột này.**
- **Sử dụng StandardScaler để chuẩn hóa dữ liệu về cùng một thang đo.**
- **Loại bỏ cột 'Player' vì nó không phải là số.**

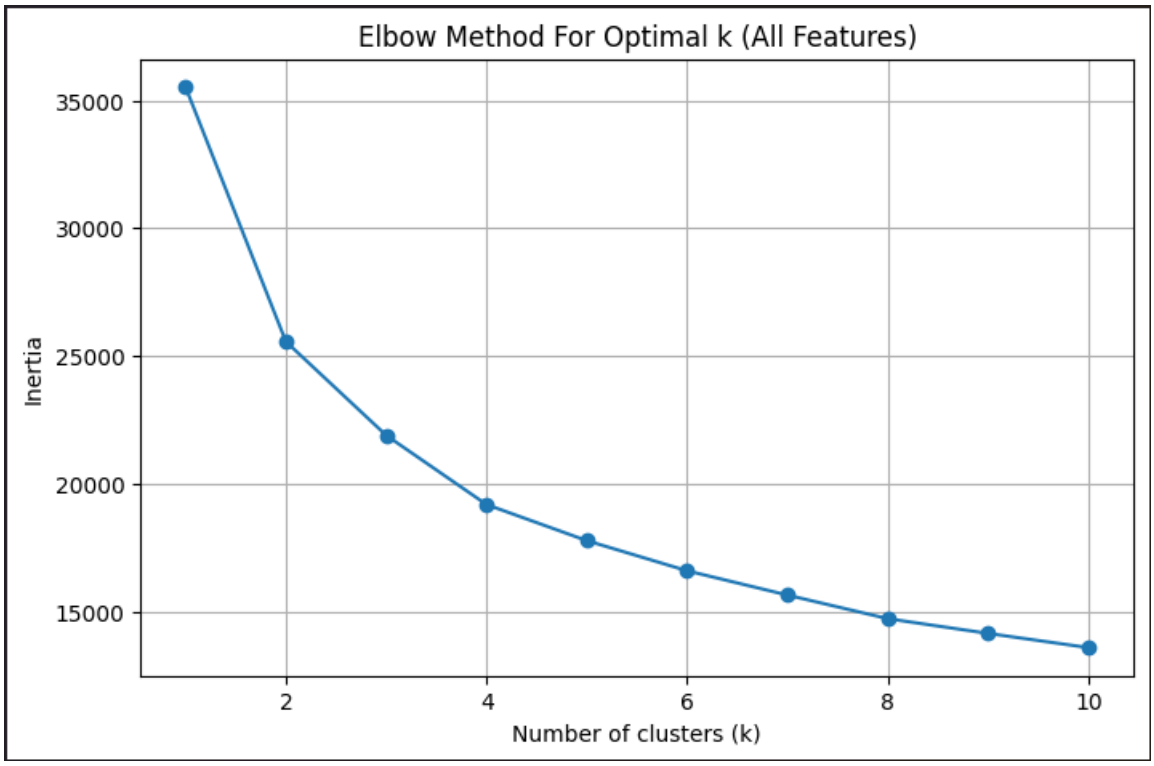
1.2. Dùng phương pháp Elbow và vẽ biểu đồ Elbow

```
inertia_all = []
K_range = range(1, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)
    kmeans.fit(scaled_df)
    inertia_all.append(kmeans.inertia_)


plt.figure(figsize=(8,5))
plt.plot(K_range, inertia_all, marker='o')
plt.title('Elbow Method For Optimal k (All Features)')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.grid(True)
plt.show()
```

- **Duyệt qua các giá trị k từ 1 đến 10.**
- **Lưu inertia (tổng khoảng cách từ các điểm đến tâm cụm) để tìm "elbow" — điểm mà việc tăng k không còn giúp giảm inertia đáng kể.**
- **Sử dụng plotting để giúp xác định số cụm tối ưu bằng mắt (elbow point).**
- **Ta có được kết quả sau:**



1.3. Phân cụm chính thức với Kmeans

```
kmeans_final = KMeans(n_clusters=4, random_state=42, n_init=10)
kmeans_final.fit(scaled_df)

df['Cluster'] = kmeans_final.labels_

print(df[['Player', 'Cluster']].head(20))
```

- **Áp dụng KMeans với 4 cụm, có thể dễ thấy từ k = 4 sự giảm của đồ thị nhỏ hơn so với từ 2 đến 3 và 3 đến 4.**
- **Gán nhãn cụm (Cluster) cho từng cầu thủ trong df.**
- **Hiển thị 20 cầu thủ đầu tiên cùng nhãn cụm tương ứng.**
- **Sau khi thực hiện lệnh print ta có được kết quả:**

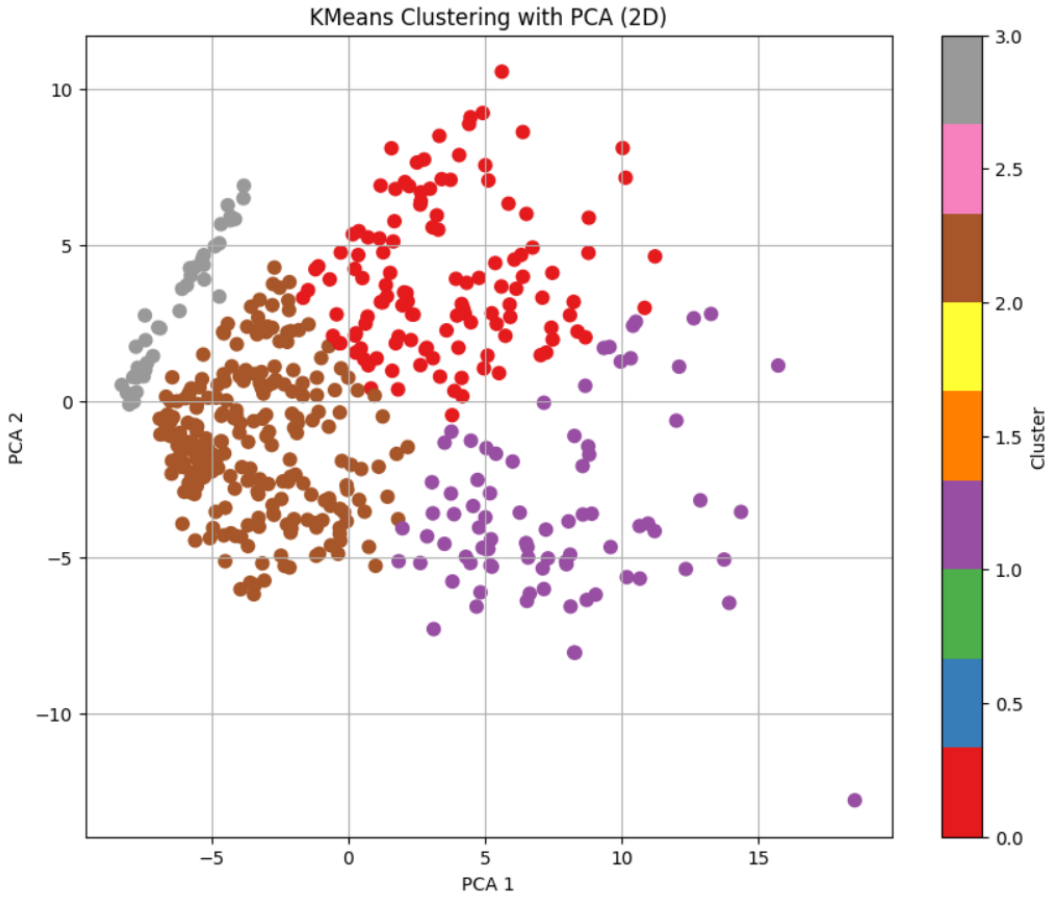
	Player	Cluster
0	Aaron Cresswell	2
1	Aaron Ramsdale	3
2	Aaron Wan-Bissaka	0
3	Abdoulaye Doucouré	0
4	Abdukodir Khusanov	2
5	Abdul Fatawu Issahaku	2
6	Adam Armstrong	2
7	Adam Lallana	2
8	Adam Smith	2
9	Adam Webster	2
10	Adam Wharton	2
11	Adama Traoré	1
12	Albert Grønbaek	2
13	Alejandro Garnacho	1
14	Alex Iwobi	1
15	Alex McCarthy	3
16	Alex Palmer	3
17	Alex Scott	2
18	Alexander Isak	1
19	Alexis Mac Allister	0

1.4. Giảm chiều dữ liệu với PCA và trực quan hóa sơ đồ PCA

```
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
pca_df = pca.fit_transform(scaled_df)


plt.figure(figsize=(10,8))
scatter = plt.scatter(pca_df[:, 0], pca_df[:, 1], c=df['Cluster'], cmap='Set1', s=50)
plt.title('KMeans Clustering with PCA (2D)')
plt.xlabel('PCA 1')
plt.ylabel('PCA 2')
plt.colorbar(scatter, label='Cluster')
plt.grid(True)
plt.show()
```

- **PCA giảm tất cả đặc trưng xuống còn 2 thành phần chính để vẽ được trên đồ thị 2D.**
- **Hiển thị các cầu thủ trên mặt phẳng PCA 2D, tô màu theo cụm.**
- **Ta thu được kết quả sau plotting:**



1.5. Đáng giá đặc trưng của từng cụm

```
stats_cols = clean_feat[4:]
cluster_means = df.groupby('Cluster')[stats_cols].mean()

cluster_means.to_excel('cluster_means.xlsx')
```

- **Nhóm các cầu thủ theo cụm mà họ đã được gán nhãn bởi Kmeans, sau đó tính giá trị trung bình của các chỉ số thống kê (trong cluster_cols) theo từng cụm để hiểu rõ đặc điểm nổi bật của mỗi nhóm cầu thủ.**
- **cluster_means.to_excel('cluster_means.xlsx') : xuất các đặc trưng trung bình của từng cụm thành file excel**
- **Ta thu được kết quả dựa vào một số đặc trưng:**
- **Cluster 0 – Cầu thủ thi đấu ổn định, thể chất tốt**
- **Tuổi: ~26.5**
- **Ra sân nhiều: ~28 trận, ~2172 phút**
- **Đóng góp ghi bàn vừa phải: ~1.33 bàn, ~1.37 kiến tạo**
- **Thu hồi bóng cao, nhận bóng nhiều, phạm lỗi nhiều**
- **Không chiến tốt: ~53%**
- **Vai trò: Tiền vệ box-to-box, hậu vệ biên hoặc trung vệ hiện đại**

- **Cluster 1 – Cầu thủ tấn công chủ lực**
- **Tuổi: ~25.6 (trẻ)**
- **Ra sân nhiều nhất: ~30 trận, ~2171 phút**
- **Ghi bàn cao nhất: ~7.28 bàn, ~4.94 kiến tạo, xG rất cao**
- **Chuyền, rê bóng và nhận bóng nhiều**
- **Không chiến kém: ~35%**
- **Vai trò: Tiền đạo, tiền vệ tấn công, cầu thủ sáng tạo chủ lực**

- **Cluster 2 – Cầu thủ dự bị hoặc ít đóng góp**
- **Tuổi: ~26.1**
- **Ra sân ít nhất: ~14 trận, ~653 phút**
- **Đóng góp tấn công và phòng ngự đều thấp**
- **Nhận bóng ít, không chiến trung bình**
- **Vai trò: Cầu thủ dự bị, tân binh, cầu thủ trẻ hoặc ít được sử dụng**

- **Cluster 3 – Cầu thủ phòng ngự kỳ cựu, ít hoạt động**
- **Tuổi: ~29.9 (già nhất)**
- **Ra sân trung bình: ~17 trận, ~1557 phút**
- **Không ghi bàn, ít kiến tạo, gần như không tấn công**
- **Không chiến thắng cực cao: ~88.4%**
- **Vai trò: Trung vệ thuần túy, thủ môn cao tuổi hoặc hậu vệ phòng ngự truyền thống**

IV, Bài 4

1. Thu thập dữ liệu chuyển nhượng cầu thủ từ trang web

1.1. Chuẩn bị dữ liệu

```
from selenium import webdriver
from selenium.webdriver.chrome.options import Options
from bs4 import BeautifulSoup
import pandas as pd
import time
```

- **Import các thư viện: Selenium (tự động hóa trình duyệt), BeautifulSoup (phân tích HTML), pandas (xử lý dữ liệu), và time (tạm dừng xử lý).**

```
def init_driver():
    chrome_options = Options()
    chrome_options.add_argument("--headless")
    chrome_options.add_argument("--disable-gpu")
    chrome_options.add_argument("--no-sandbox")
    return webdriver.Chrome(options=chrome_options)
```

- **khởi tạo một trình duyệt Chrome không hiển thị giao diện (headless) bằng Selenium**

```
def get_players_table_selenium(driver, url):
    driver.get(url)
    time.sleep(5)
    soup = BeautifulSoup(driver.page_source, 'html.parser')
    table = soup.find("table")
    if table:
        df = pd.read_html(str(table))[0]
        return df
    else:
        print(f"❌ Không tìm thấy bảng tại {url}")
        return None
```

- **Truy cập trang, chờ tải xong, tìm bảng dữ liệu rồi dùng pandas.read_html() để chuyển sang DataFrame.**

```
urls = [
    "https://www.footballtransfers.com/en/players/uk-premier-league",
    "https://www.footballtransfers.com/en/players/uk-premier-league/2",
    "https://www.footballtransfers.com/en/players/uk-premier-league/3",
    "https://www.footballtransfers.com/en/players/uk-premier-league/4",
    "https://www.footballtransfers.com/en/players/uk-premier-league/5",
    "https://www.footballtransfers.com/en/players/uk-premier-league/6",
    "https://www.footballtransfers.com/en/players/uk-premier-league/7",
    "https://www.footballtransfers.com/en/players/uk-premier-league/8",
    "https://www.footballtransfers.com/en/players/uk-premier-league/9",
    "https://www.footballtransfers.com/en/players/uk-premier-league/10",
    "https://www.footballtransfers.com/en/players/uk-premier-league/11",
    "https://www.footballtransfers.com/en/players/uk-premier-league/12",
    "https://www.footballtransfers.com/en/players/uk-premier-league/13",
    "https://www.footballtransfers.com/en/players/uk-premier-league/14",
    "https://www.footballtransfers.com/en/players/uk-premier-league/15",
    "https://www.footballtransfers.com/en/players/uk-premier-league/16",
    "https://www.footballtransfers.com/en/players/uk-premier-league/17",
    "https://www.footballtransfers.com/en/players/uk-premier-league/18",
    "https://www.footballtransfers.com/en/players/uk-premier-league/19",
    "https://www.footballtransfers.com/en/players/uk-premier-league/20",
    "https://www.footballtransfers.com/en/players/uk-premier-league/21",
    "https://www.footballtransfers.com/en/players/uk-premier-league/22"
]
```

- **Danh sách chứa các đường link của từng trang dữ liệu chuyển nhượng.**

```
driver = init_driver()
all_players_data = []

for idx, url in enumerate(urls, 1):
    print(f"🔍 Đang thu thập dữ liệu từ trang {idx}...")
    df = get_players_table_selenium(driver, url)
    if df is not None:
        all_players_data.append(df)
        print(f"✅ Đã thu thập {len(df)} dòng từ trang {idx}.")
    else:
        print(f"⚠️ Không thể thu thập dữ liệu từ trang {idx}.")

driver.quit()
```

- **Duyệt qua từng URL trong danh sách urls, dùng Selenium để mở trang, trích xuất bảng cầu thủ bằng hàm get_players_table_selenium(), lưu vào danh sách all_players_data.**

```
if all_players_data:
    all_data = pd.concat(all_players_data, ignore_index=True)
    all_data.to_csv("football_transfers_players_all_pages.csv", index=False)
    print(f"📁 Đã lưu {len(all_data)} dòng dữ liệu vào 'football_transfers_players_all_pages.csv'")
else:
    print("❌ Không có dữ liệu để lưu.")
```

- **Gộp tất cả dữ liệu lại và lưu vào file CSV**

2. Tiền xử lý dữ liệu

2.1. Xử lý tên cột và tên cầu thủ, dữ liệu dư thừa

```
value_df = all_data.copy()
value_df.columns = value_df.columns.get_level_values(0)
```

- **Tạo một bản sao của all_data có tên là value_df, value_df để lưu giá trị chuyển nhượng và tên cầu thủ**

```
def remove_duplicate_prefix(text):
    words = text.split()
    for i in range(1, len(words)//2 + 1):
        if words[:i] == words[i:2*i]:
            return ' '.join(words[:i])
    return text

value_df['Player'] = value_df['Player'].apply(remove_duplicate_prefix)
```

- **Loại bỏ phần tên bị lặp kiểu và dữ liệu bị thừa ví dụ : " Erling Haaland Erling Haaland Man City • F (C) F (C) " → "Erling Haaland"**

```
player_df = merged_df[clean_feat].copy()

value_df.drop(columns=['Skill / pot', 'Age', 'Team'], inplace=True)
value_df.head()
```

- **Tạo một bản sao mới của bảng dữ liệu merged_df cùng các đặc trừng trong danh sách clean_feat, thay vì sử dụng trực tiếp merged_df[clean_feat] em sử dụng một bản sao để không ảnh hưởng trực tiếp đến bảng dữ liệu ban đầu khi thực hiện các thay đổi đối với player_df**
- **In ra 5 hàng đầu của bảng dữ liệu value_df với câu lệnh value_df.head()**

```
def normalize_name(name):
    return ' '.join(sorted(name.lower().split()))

player_df["Normalized_Name"] = player_df["Player"].apply(normalize_name)
value_df["Normalized_Name"] = value_df["Player"].apply(normalize_name)
```

- **Do dữ liệu được lấy từ hai trang web khác nhau là FBREF và FOOTBALLTRANSFERS nên cách sắp xếp tên cầu thủ khác nhau, ví dụ 'Son Heung-Min' và 'Heung-Min Son', dó đó sắp**

xếp lại tên theo bảng chữ cái, viết thường để phục vụ việc gộp dữ liệu (merge).

2.2. Merge dữ liệu và thực hiện các xử lý thêm

```
final_df = pd.merge(player_df, value_df, on="Normalized_Name", how="inner", suffixes=('_df1', '_df2'))
final_df.rename(columns={"Player_df1": "Player"}, inplace=True)
final_df.drop(columns=['Normalized_Name', 'Player_df2'], inplace=True)


for index, row in final_df.iterrows():
    for col in final_df.columns:
        if pd.isnull(row[col]):
            print(f"❌ Missing '{col}' for player: {row['Player_df1']}")


final_df = final_df[final_df['standard_Playing Time | Min'] > 900]
```

- **Gộp hai bảng player_df và value_df theo tên đã chuẩn hóa (Normalized_Name), how="inner": Chỉ giữ lại các cầu thủ xuất hiện trong cả hai bảng.**
- **Player_df1 là tên gốc từ player_df, đổi lại thành "Player" để dễ hiểu.**
- **Player_df2 là tên từ value_df, bị loại bỏ sau khi ghép vì đã chọn giữ cột ở player_df.**
- **Duyệt từng hàng và từng cột trong final_df.**
- **Nếu ô nào thiếu dữ liệu (NaN), in ra cảnh báo kèm tên cầu thủ.**
- **Giữ lại các cầu thủ có thời gian ra sân trên 900 phút.**

```
def map_position_type(pos_str):
    if 'GK' in pos_str:
        return 'Goalkeeper'
    elif 'FW' in pos_str and 'MF' not in pos_str and 'DF' not in pos_str:
        return 'Attacker'
    elif 'FW' in pos_str and 'MF' in pos_str:
        return 'Attacking Midfielder'
    elif 'FW' in pos_str and 'DF' in pos_str:
        return 'Attacking Defender'
    elif 'MF' in pos_str and 'DF' in pos_str:
        return 'Defensive Midfielder'
    elif 'MF' in pos_str:
        return 'Midfielder'
    elif 'DF' in pos_str:
        return 'Defender'
    else:
        return 'Unknown'

final_df['PositionType'] = final_df['standard | Pos'].apply(map_position_type)


final_df['standard | Nation'] = final_df['standard | Nation'].apply(lambda x: x.split()[1])
```

- **Hàm map_position_type để phân loại vị trí thi đấu, sau đó sử dụng apply để áp dụng đối với final_df['standard | Pos'] để tạo một đặc trưng mới là 'PositionType'**
- **Giả sử giá trị ban đầu như: "br BRA", "fr FRA", "pt POR" sử dụng x.split()[1] để lấy phần cuối là 'BRA', 'FRA', 'POR'.**

```
def convert_market_value(value_str):
    value_str = value_str.lower().replace('€', '').strip()
    if value_str.endswith('k'):
        return int(float(value_str[:-1]) * 1_000)
    elif value_str.endswith('m'):
        return int(float(value_str[:-1]) * 1_000_000)
    else:
        return int(float(value_str))  # nếu không có hậu tố

final_df['ETV'] = final_df['ETV'].apply(convert_market_value)

final_df.rename(columns={'ETV' : 'Market Value(EUR)'}, inplace=True)
```

- **Chuyển đổi giá trị chuyển nhượng sang số, dữ liệu như "€5.8M" hoặc "€350k" sẽ được chuyển thành 5800000 hoặc 350000, xóa ký hiệu euro (€) và xử lý hậu tố k, m. Sau đó áp dụng đối với final_df['ETV']**
- **Đổi tên 'ETV' sang 'Market Value(EUR)' để dễ hiểu hơn**

3. Chuẩn bị và chuẩn hóa dữ liệu

3.1. Import các thư viện cần thiết

```
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import mean_absolute_error, mean_squared_error, r2_score
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
import numpy as np
```

- **Các thư viện này sẽ được sử dụng trong quá trình huấn luyện và thử nghiệm đối với mô hình**

3.2. Xác định các đặc trừng và biến mục tiêu

```
feature_cols = [
    # Thông tin nền tảng
    'standard | Age',
    'standard | Nation',
    'standard | Squad',
    'PositionType',

    # Thời gian thi đấu
```

```
    'standard_Playing Time | MP',
    'standard_Playing Time | Starts',
    'standard_Playing Time | Min',

    # Hiệu suất tổng quát
    'standard_Performance | Gls',
    'standard_Performance | Ast',
    'standard_Expected | xG',
    'standard_Expected | xAG',
    'standard_Per 90 Minutes | Gls',
    'standard_Per 90 Minutes | Ast',
    'standard_Per 90 Minutes | xG',
    'standard_Per 90 Minutes | xAG',


    # Dứt điểm
    'shooting_Standard | SoT%',
    'shooting_Standard | G/Sh',
    'shooting_Standard | Dist',

    # Tạo cơ hội
    'passing | KP',
    'passing | 1/3',
    'passing | PPA',
    'passing | CrsPA',
    'gca_SCA | SCA',
    'gca_SCA | SCA90',
    'gca_GCA | GCA',
    'gca_GCA | GCA90',

    # Tiến triển bóng
    'standard_Progression | PrgC',
    'standard_Progression | PrgP',
    'standard_Progression | PrgR',

    # Phòng ngự
    'defense_Tackles | Tkl',
    'defense_Tackles | TklW',
    'defense_Blocks | Blocks',
    'defense | Int',
    'misc_Performance | Recov',
    'misc_Aerial Duels | Won%',

    # Thủ môn
    'keepers_Performance | GA90',
    'keepers_Performance | Save%',
    'keepers_Performance | CS%',
    'keepers_Penalty Kicks | Save%',

    # Phân phối và kiểm soát bóng
```

```
    'passing_Total | Cmp%',
    'passing_Total | TotDist',
    'passing_Short | Cmp%',
    'passing_Medium | Cmp%',
    'passing_Long | Cmp%',
    'possession_Carries | Carries',
    'possession_Carries | PrgDist',
    'possession_Receiving | PrgR',
]
```

Danh sách các đặc trưng được sử dụng cho mô hình

```
X = final_df[feature_cols]
y = final_df['Market Value(EUR)']
```

- **X là đầu vào của mô hình, y là giá trị mục tiêu**

```
for col in categorical_cols:
    le = LabelEncoder()
    X[col] = le.fit_transform(X[col])
```

- **Mã hóa các cột dạng chuỗi (quốc tịch, đội bóng, vị trí) thành số để mô hình xử lý được.**

```
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.8, test_size=0.2, random_state=42)
```

- **Tách tập huấn luyện và kiểm tra, 80% để huấn luyện (X_train), 20% để kiểm tra (X_test).**

```
scl = StandardScaler()
X_train = scl.fit_transform(X_train)
X_test = scl.transform(X_test)
```

- **Dữ liệu được chuẩn hóa về trung bình = 0, độ lệch chuẩn = 1 để đảm bảo các đặc trưng có cùng thang đo.**

4. Huấn luyện và đánh giá mô hình

4.1. Sử dụng RandomForestRegressor để huấn luyện mô hình

```
rf_model = RandomForestRegressor(n_estimators=100, random_state=42)
rf_model.fit(X_train, y_train)
```

- **Sử dụng 100 cây quyết định (n_estimators=100) để tạo mô hình dự đoán, sau đó fit dữ liệu vào mô hình**

4.2. Dự đoán kết quả

```
y_pred = rf_model.predict(X_test)
pred_df = pd.DataFrame({'Actual': y_test, 'Predicted': y_pred})
pred_df['Predicted'] = pred_df['Predicted'].astype(int)
pred_df['Player'] = final_df.loc[y_test.index, 'Player']
pred_df = pred_df[['Player', 'Actual', 'Predicted']]
pred_df.index = range(1, len(pred_df) + 1)
pred_df.to_csv('prediction.csv')
```

- **Dự đoán đầu ra (y_pred) từ tập dữ liệu kiểm tra (X_test) bằng mô hình Random Forest (rf_model).**
- **Tạo một DataFrame mới gồm hai cột: giá trị thực tế (Actual) và giá trị mô hình dự đoán (Predicted).**
- **Ép kiểu integer đối với 'Predicted' để hiển thị ra theo dạng 33898000 thay vì 3.389800e+07**
- **Thêm cột 'Player' vào pred_df, trích xuất từ final_df, tương ứng với các chỉ số của y_test. Điều này giúp gắn tên cầu thủ tương ứng với từng dòng dữ liệu.**
- **Sắp xếp lại thứ tự các cột để có: tên cầu thủ, giá trị thực tế, và giá trị dự đoán.**
- **Đặt lại chỉ số của pred_df bắt đầu từ 1 thay vì 0.**
- **Lưu kết quả thực tế và dự đoán vào file prediction.csv.**

4.3. Sử dụng các thước đo để đánh giá mô hình

```
r2 = r2_score(y_test, y_pred)
mae = mean_absolute_error(y_test, y_pred)
mse = mean_squared_error(y_test, y_pred)
rmse = np.sqrt(mse)

print(f"R2 Score: {r2}")
print(f"MAE: {mae}")
print(f"MSE: {mse}")
print(f"RMSE: {rmse}")
```

- **Tính R^2 Score (hệ số xác định): độ đo độ chính xác tổng thể, gần 1 là tốt.**
- **MAE (Mean Absolute Error): sai số tuyệt đối trung bình, đơn vị giống đầu ra, càng thấp càng tốt.**
- **MSE (Mean Squared Error): sai số bình phương trung bình, nhấn mạnh sai lệch lớn hơn.**

- **RMSE (Root Mean Squared Error): căn bậc hai của MSE, giúp đưa sai số về cùng đơn vị với biến đầu ra.**
- **In ra các chỉ số đánh giá để bạn dễ kiểm tra hiệu quả của mô hình.**

4.4. Đánh giá mức độ quan trọng của các đặc trưng

```
feature_importances = rf_model.feature_importances_

importances_rdf = pd.DataFrame({'Feature':X.columns, 'Importance':feature_importances})

importances_rdf = importances_rdf.sort_values(by='Importance', ascending=False)

importances_rdf.plot(kind = 'bar', x = "Feature" , y = "Importance", color = 'lightblue')
```

- **Giúp biết đặc trưng nào ảnh hưởng nhiều nhất đến giá trị cầu thủ (ví dụ: số bàn thắng, số phút thi đấu...).**
- **Kết quả sau khi đánh giá mức độ quan trọng của đặc trưng:**