HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG
KHOA CÔNG NGHỆ THÔNG TIN 1

-----□□□□□-----

BÀI TẬP LỚN MÔN
LẬP TRÌNH PYTHON

Giảng viên hướng dẫn: Kim Ngọc Bách

Lớp : D23CQCE04-B

Sinh viên thực hiện: An Việt Quyền

Mã sinh viên : D23DCVT365

Mục Lục

I. ĐỀ BÀI

Problem 1:Write a Python program to collect footballer player statistical data with the following requirements:

• Collect statistical data [*] for all players who have played more than 90 minutes in the 2024-2025 English Premier League season. • Data source: https://fbref.com/en/

• Save the result to a file named 'results.csv', where the result table has the following structure:

- Each column corresponds to a statistic.
- Players are sorted alphabetically by their first name.
- Any statistic that is unavailable or inapplicable should be marked as "N/a".

• [*] The required statistics are:

- Nation
- Team
- Position
- Age
- Playing Time: matches played, starts, minutes
- Performance: goals, assists, yellow cards, red cards
- Expected: expected goals (xG), expedited Assist Goals (xAG)
- Progression: PrgC, PrgP, PrgR
- Per 90 minutes: Gls, Ast, xG, xGA
- Goalkeeping:
    + Performance: goals against per 90 mins (GA90), Save%, CS%
    + Penalty Kicks: penalty kicks Save%
- Shooting:
    + Standard: shoots on target percentage (SoT%), Shoot on

Target per 90min (SoT/90), goals/shot (G/sh), average shoot distance (Dist)

- Passing:
  + Total: passes completed (Cmp),Pass completion (Cmp%), progressive passing distance (TotDist)
  + Short: Pass completion (Cmp%),
  + Medium: Pass completion (Cmp%),
  + Long: Pass completion (Cmp%),
  + Expected: key passes (KP), pass into final third (1/3), pass into penalty area (PPA), CrsPA, PrgP
- Goal and Shot Creation:
  + SCA: SCA, SCA90
  + GCA: GCA, GCA90
- Defensive Actions:
  + Tackles: Tkl, TklW
  + Challenges: Att, Lost
  + Blocks: Blocks, Sh, Pass, Int
- Possession:
  + Touches: Touches, Def Pen, Def 3rd, Mid 3rd, Att 3rd, Att Pen
  + Take-Ons: Att, Succ%, Tkld%
  + Carries: Carries, ProDist, ProgC, 1/3, CPA, Mis, Dis
  + Receiving: Rec, PrgR o Miscellaneous Stats:
  + Performance: Fls, Fld, Off, Crs, Recov
  + Aerial Duels: Won, Lost, Won%
- Reference: https://fbref.com/en/squads/822bd0ba/Liverpool-Stats

Problem 2:

- Identify the top 3 players with the highest and lowest scores for each statistic. Save result to a file name ‘top_3.txt’
- Find the median for each statistic. Calculate the mean and standard deviation for each statistic across all players and for each team. Save the results to a file named 'results2.csv' with the following format:

		Median of Attribute 1	Mean of Attribute 1	Sid of Attribute 1	. . .	. . .
0	all					
1	Team 1					
…						
n	Team n					

- Plot a histogram showing the distribution of each statistic for all players in the league and each team.
- Identify the team with the highest scores for each statistic. Based on your analysis, which team do you think is performing the best in the 2024-2025 Premier League season?
- Histogram Plot: https://matplotlib.org/stable/api/_as_gen/matplotlib.pyplot.hist.html

Problem 3:

- Use the K-means algorithm to classify players into groups based on their statistics.
- How many groups should the players be classified into? Why? Provide your comments on the results.
- Use PCA to reduce the data dimensions to 2, then plot a 2D cluster of the data points

Problem 4:

- Collect player transfer values for the 2024-2025 season from https://www.footballtransfers.com.
- Note that only collect for the players whose playing time is greater than 900 minutes
- Propose a method for estimating player values. How do you select feature and model?

II . XỬ LÝ CÂU HỎI

Problem 1:

A. Phân tích đề bài

Viết chương trình Python thu thập dữ liệu thống kê cầu thủ Premier League 2024–2025 từ FBref.com, với yêu cầu:

- Yêu cầu chính:
  + Chỉ lấy cầu thủ đá > 90 phút.
  + Nguồn dữ liệu: FBref (các bảng thống kê từng đội bóng).
  + Lưu kết quả vào: results.csv.
- Yêu cầu về định dạng file:
  + Mỗi cột là một chỉ số thống kê.
  + Cầu thủ được sắp xếp theo tên đầu (first name).
  + Nếu thiếu dữ liệu → ghi "N/a".
- Cần thu thập từ nhiều bảng thống kê khác nhau, sau đó gộp, chuẩn hóa và xuất ra CSV.

B. Phân tích bài làm

1. Thư viện sử dụng

- BeautifulSoup (bs): Dùng để phân tích HTML sau khi tải bằng Selenium.
- Selenium: Tự động điều khiển trình duyệt Chrome để tải dữ liệu web.
- WebDriverWait + EC: Đợi trang load xong mới xử lý.
- Pandas (pd): Lưu trữ và xuất dữ liệu dưới dạng bảng (Dataframe)
- Regex (re): Xử lý chuỗi (ví dụ: trích quốc tịch).

- time: Thêm delay nếu cần.

2. Cấu hình ChromeDriver

- Thiết lập đường dẫn đến ChromeDriver và khởi tạo trình điều khiển driver.
- Đảm bảo đã cài đúng version chromedriver khớp với Chrome.

```
    # Cấu hình ChromeDriver
    service = Service("C:/ChromeDriver/chromedriver.exe")  # Thay đổi đường dẫn
    driver = webdriver.Chrome(service=service)
```

3. Danh sách URL các bảng thống kê.

- **Mỗi URL tương ứng một loại thống kê: tiêu chuẩn (standard), dứt điểm (shooting), chuyền bóng (passing), kiến tạo (gca), phòng ngự (defense), kiểm soát bóng (possession), chỉ số phụ (misc), và thủ môn (keepers).**

```
# Danh sách các URL
STATS_URLS = {
    'standard': "https://fbref.com/en/comps/9/stats/Premier-League-Stats",
    'shooting': "https://fbref.com/en/comps/9/shooting/Premier-League-Stats",
    'passing': "https://fbref.com/en/comps/9/passing/Premier-League-Stats",
    'gca': "https://fbref.com/en/comps/9/gca/Premier-League-Stats",
    'defense': "https://fbref.com/en/comps/9/defense/Premier-League-Stats",
    'possession': "https://fbref.com/en/comps/9/possession/Premier-League-Stats",
    'misc': "https://fbref.com/en/comps/9/misc/Premier-League-Stats",
    'keepers': "https://fbref.com/en/comps/9/keepers/Premier-League-Stats"
}
```

4. Bảng ánh xạ

Key là data-stat trên trang FBref.

Value gồm:

+ Tên cột mới ('Standard_Gls', 'Passing_Cmp', ...)
+ Loại dữ liệu (int, float, str).

Dùng để ánh xạ chính xác và chuẩn hóa dữ liệu.

```python
# Mapping đầy đủ data-stat
COLUMN_MAP = {
    'player': ('Player', str),
    'nationality': ('Nation', str),
    'position': ('Pos', str),
    'team': ('Team', str),
    'age': ('Age', int),

    # Standard Stats
    'minutes': ('Standard_Min', int),
    'goals': ('Standard_Gls', int),
    'assists': ('Standard_Ast', int),
    'cards_yellow': ('Standard_CrdY', int),
    'cards_red': ('Standard_CrdR', int),
    'xg': ('Standard_xG', float),
    'xg_assist': ('Standard_xAG', float),
    'progressive_carries': ('Standard_PrgC', int),
    'progressive_passes': ('Standard_PrgP', int),
    'progressive_passes_received': ('Standard_PrgR', int),

    # Per 90 Stats
    'goals_per90': ('Standard_Gls/90', float),
    'assists_per90': ('Standard_Ast/90', float),
    'xg_per90': ('Standard_xG/90', float),
    'xg_assist_per90': ('Standard_xAG/90', float),

    # Shooting
    'shots_on_target_percent': ('Shooting_SoT%', float),
    'shots_on_target_per90': ('Shooting_SoT/90', float),
    'goals_per_shot': ('Shooting_G/Sh', float),
    'average_shot_distance': ('Shooting_Dist', float),

    # Passing
    'passes_completed': ('Passing_Cmp', int),
    'passes_pct': ('Passing_Total_Cmp%', float),
    'passes_total_distance': ('Passing_TotDist', int),
    'passes_short_pct': ('Passing_Short_Cmp%', float),
    'passes_medium_pct': ('Passing_Medium_Cmp%', float),
    'passes_long_pct': ('Passing_Long_Cmp%', float),
    'assisted_shots': ('Passing_KP', int),
    'passes_into_final_third': ('Passing_1/3', int),
    'passes_into_penalty_area': ('Passing_PPA', int),
    'crosses_into_penalty_area': ('Passing_CrsPA', int),
```

```
    # Goalkeeping
    'goals_against_per90': ('Goalkeeping_GA90', float),
    'save_percent': ('Goalkeeping_Save%', float),
    'clean_sheets_pct': ('Goalkeeping_CS%', float),
    'penalty_save_percent': ('Goalkeeping_Penalty_Save%', float),

    # Defense
    'tackles': ('Defense_Tkl', int),
    'tackles_won': ('Defense_TklW', int),
    'dribblers_tackled': ('Defense_Att', int),
    'challenges_lost': ('Defense_Lost', int),
    'blocks': ('Defense_Blocks', int),
    'blocked_shots': ('Defense_Sh', int),
    'blocked_passes': ('Defense_Pass', int),
    'interceptions': ('Defense_Int', int),

    # Possession
    'touches': ('Possession_Touches', int),
    'touches_def_pen_area': ('Possession_Def Pen', int),
    'touches_def_3rd': ('Possession_Def 3rd', int),
    'touches_mid_3rd': ('Possession_Mid 3rd', int),
    'touches_att_3rd': ('Possession_Att 3rd', int),
    'touches_att_pen_area': ('Possession_Att Pen', int),
    'dribble_success_pct': ('Possession_Succ%', float),
    'dribbled_past_pct': ('Possession_Tkld%', float),
    'carries': ('Possession_Carries', int),
    'carry_progressive_distance': ('Possession_PrgDist', int),
    'carries_into_final_third': ('Possession_1/3', int),
    'carries_into_penalty_area': ('Possession_CPA', int),
    'miscontrols': ('Possession_Mis', int),
    'dispossessed': ('Possession_Dis', int),
    'passes_received': ('Possession_Rec', int),

    # Misc
    'fouls': ('Misc_Fls', int),
    'fouls_drawn': ('Misc_Fld', int),
    'offsides': ('Misc_Off', int),
    'crosses': ('Misc_Crs', int),
    'ball_recoveries': ('Misc_Recov', int),
    'aerials_won': ('Misc_Won', int),
    'aerials_lost': ('Misc_Lost', int),
    'aerials_won_pct': ('Misc_Won%', float)
}
```

5. Hàm get_html(url)

- Truy cập url, đợi đến khi bảng dữ liệu xuất hiện.
- time.sleep(2) để đảm bảo toàn bộ nội dung được tải (tránh lỗi thiếu dữ liệu).
- Trả về mã HTML của trang.

```
def get_html(url):
    driver.get(url)
    WebDriverWait(driver, 20).until(EC.presence_of_element_located((By.CSS_SELECTOR, 'table.stats_table')))
    time.sleep(2)
    return driver.page_source
```

6. Hàm parse_value(raw_value, dtype)

- Làm sạch và chuyển đổi raw_value (giá trị thô từ HTML) sang kiểu dữ liệu mong muốn.
- Nếu không có dữ liệu → 'N/A'.
- Với kiểu float, chuyển % về dạng số thập phân.
- Với kiểu int, xử lý số dạng "1,234" hoặc "5.0".

```
121  def parse_value(raw_value, dtype):
122      try:
123          if not raw_value or raw_value.strip() == '':
124              return 'N/A'
125
126          cleaned_value = raw_value.replace(',', '').replace('%', '').replace('m', '').strip()
127
128          # Xử lý giá trị đặc biệt
129          if dtype == int:
130              return int(float(cleaned_value)) if cleaned_value else 0
131          elif dtype == float:
132              if '%' in raw_value:
133                  return round(float(cleaned_value)/100, 3)
134              return float(cleaned_value) if cleaned_value else 0.0
135          else:
136              return raw_value.strip()
137      except Exception as e:
138          return 'N/A'
```

7. Khởi tạo kho dữ liệu

- Dạng dict lưu dữ liệu theo tên cầu thủ.
- Mỗi cầu thủ là 1 dict con chứa các cột thống kê đã ánh xạ từ ’COLUMN_MAP’.

```
140  # Khởi tạo dictionary lưu dữ liệu
141  players_data = {}
142
```

8. Crawl từng URL trong STATS_URLS

- Lặp qua tất cả các loại thống kê.
- Gọi get_html(url) → phân tích bằng BeautifulSoup.
- Tìm bảng theo id=f'stats_{stat_type}'.
- Lặp qua các dòng (<tr>) trong <tbody>, lấy tên cầu thủ và từng chỉ số.

```python
# Crawl dữ liệu từ tất cả các bảng
for stat_type, url in STATS_URLS.items():
    print(f"Đang crawl: {stat_type}...")
    try:
        html = get_html(url)
        soup = bs(html, 'html.parser')
        table = soup.find('table', id=f'stats_{stat_type}')

        if not table:
            continue

        for row in table.find('tbody').find_all('tr'):
            # Lấy thông tin cầu thủ
            player_cell = row.find('th', {'data-stat': 'player'}) or row.find('td', {'data-stat': 'player'})
            if not player_cell:
                continue

            player_name = player_cell.text.strip()
            if player_name not in players_data:
                players_data[player_name] = {col: 'N/A' for col, _ in COLUMN_MAP.values()}
                players_data[player_name]['Player'] = player_name

            # Cập nhật dữ liệu
            for cell in row.find_all(['th', 'td']):
                stat = cell.get('data-stat')
                if stat in COLUMN_MAP:
                    col_name, dtype = COLUMN_MAP[stat]
                    raw_value = cell.text.strip()

                    # Xử lý đặc biệt
                    if stat == 'age' and '-' in raw_value:
                        raw_value = raw_value.split('-')[0]
                    if stat == 'nationality':
                        match = re.search(r'(.+?)\s+\(([A-Z]{3})\)', raw_value)
                        raw_value = f"{match.group(1)} ({match.group(2)})" if match else raw_value

                    # Gán giá trị đã parse
                    parsed_value = parse_value(raw_value, dtype)
                    players_data[player_name][col_name] = parsed_value
    except Exception as e:
        print(f"Lỗi khi crawl {stat_type}: {str(e)}")
        continue
```

- Chi tiết bên trong:
    + Lấy tên cầu thủ. Nếu cầu thủ chưa có trong players_data → khởi tạo.

```python
            # Cập nhật dữ liệu
            for cell in row.find_all(['th', 'td']):
                stat = cell.get('data-stat')
                if stat in COLUMN_MAP:
                    col_name, dtype = COLUMN_MAP[stat]
                    raw_value = cell.text.strip()

                    # Xử lý đặc biệt
                    if stat == 'age' and '-' in raw_value:
                        raw_value = raw_value.split('-')[0]
                    if stat == 'nationality':
                        match = re.search(r'(.+?)\s+\(([A-Z]{3})\)', raw_value)
                        raw_value = f"{match.group(1)} ({match.group(2)})" if match else raw_value

                    # Gán giá trị đã parse
                    parsed_value = parse_value(raw_value, dtype)
                    players_data[player_name][col_name] = parsed_value
```

    + Lặp qua từng ô dữ liệu trong dòng đó.

+ Nếu data-stat có trong ‘ COLUMN_MAP ‘:

  Lấy tên cột và kiểu dữ liệu mong muốn.

  Xử lý dữ liệu đặc biệt: tuổi dạng 24-150, quốc tịch dạng England (ENG), …

  Gọi parse_value() để chuyển dữ liệu đúng kiểu.

  Lưu vào: players_data [player_name] [column_name].

9. Lọc dữ liệu hợp lệ

- Chỉ giữ cầu thủ có số phút thi đấu > 90 hoặc bị thiếu ('N/A').
- Đảm bảo đúng thứ tự các cột (theo COLUMN_MAP.values()).

```
# Lọc và chuẩn bị dữ liệu cuối cùng
filtered_data = []
for player in players_data.values():
    try:
        if player.get('Standard_Min', 0) > 90 or player.get('Standard_Min') == 'N/A':
            # Thay thế None/NaN bằng 'N/A' và đảm bảo thứ tự cột
            ordered_row = [
                player.get(col, 'N/A') if player.get(col) not in [None, ''] else 'N/A'
                for col, _ in COLUMN_MAP.values()
            ]
            filtered_data.append(ordered_row)
    except:
        continue

```

10. Tạo DataFrame, xuất file csv và đóng trình duyệt

- Tạo DataFrame từ dữ liệu đã lọc.
- Ghi file CSV tên results.csv.
- Giải phóng tài nguyên trình duyệt Chrome.

```
# Tạo DataFrame
columns = [col for col, _ in COLUMN_MAP.values()]
df = pd.DataFrame(filtered_data, columns=columns)
df.fillna('N/A', inplace=True)

# Xuất CSV
df.to_csv("results.csv", index=False)
print(f"{len(df)}")
driver.quit()
```

Problem 2:

A.Phân tích đề bài

1. Top 3 cầu thủ cao nhất/thấp nhất mỗi chỉ số
 → Ghi vào top_3.txt
2. Tính median, mean, std cho mỗi chỉ số
 → Cả giải và từng đội
 → Ghi vào results2.csv
3.Vẽ histogram phân phối mỗi chỉ số
 → Toàn giải và từng đội
4.Tìm đội có điểm cao nhất mỗi chỉ số
 → Phân tích đội mạnh nhất mùa 2024–2025

B. Phân tích bài làm

I. Identify the top 3 players with the highest and lowest scores for each statistic.Save result to a file name ‘top_3.txt’

1. Thư viện sử dụng

- pandas (pd) : thư viện dùng để xử lý dữ liệu dạng bảng (DataFrame).
- numpy (np) : dùng cho xử lý mảng số và toán học, ví dụ: xác định kiểu dữ liệu số (np.number).

```
1  import pandas as pd
2  import numpy as np
3
```

2. Đọc dữ liệu từ file

- Đọc dữ liệu từ file CSV tên "results.csv" vào DataFrame df.

```
4    # Đọc dữ liệu từ file CSV
5    df = pd.read_csv("results.csv")
```

3. Lọc ra các cột chứa số liệu thống kê

- Lấy ra các cột kiểu số (int, float) các chỉ số thống kê như bàn thắng, kiến tạo, ...

```
7  # Lọc các cột chứa số liệu thống kê
8  numeric_df = df.select_dtypes(include=[np.number])
9
```

4. Loại trừ các cột không dùng để xếp hạng

- exclude_cols: là danh sách các cột số nhưng không dùng để so sánh thành tích (tuổi, số phút thi đấu).
- stat_cols: là danh sách các cột thống kê thực sự để xếp hạng, bỏ qua những cột bị loại trừ.

```
# Loại bỏ các cột không cần thống kê xếp hạng
exclude_cols = ['Age', 'Standard_Min']
stat_cols = [col for col in numeric_df.columns if col not in exclude_cols]

```

5. Lưu thông tin cầu thủ và lưu kết quả

- Lưu lại thông tin cầu thủ và đội bóng để kết hợp khi hiển thị kết quả.
- Tạo danh sách rỗng để lưu kết quả top 3 và bottom 3 của từng chỉ số.

```
# Chuẩn bị thông tin cầu thủ để kết hợp
player_info = df[['Player', 'Team']]
top_3_results = []

```

6. Duyệt qua từng chỉ số và tìm top 3

- Với mỗi cột thống kê (col):
    + nlargest(3, col): chọn 3 cầu thủ có giá trị lớn nhất.
    + nsmallest(3, col): chọn 3 cầu thủ có giá trị nhỏ nhất.
    + Chỉ lấy các cột: Player, Team, và chỉ số đó để hiển thị.

```
# Duyệt qua từng cột thống kê
for col in stat_cols:
    top3 = df.nlargest(3, col)[['Player', 'Team', col]]
    bottom3 = df.nsmallest(3, col)[['Player', 'Team', col]]

```

7. Ghi từng phần vào danh sách kết quả

- Ghi tiêu đề của chỉ số (Statistic: Goals, Statistic: xG,...).
- Ghi top 3, bottom 3, phân cách bằng dòng
- Dùng to_string(index=False) để xuất DataFrame ra dạng văn bản, bỏ chỉ số dòng.

```
    # Ghi vào danh sách kết quả
    top_3_results.append(f"Statistic: {col}\nTop 3:")
    top_3_results.extend(top3.to_string(index=False).split('\n'))
    top_3_results.append("Bottom 3:")
    top_3_results.extend(bottom3.to_string(index=False).split('\n'))


```

8. Ghi kết quả vào file văn bản

- Tạo file top_3.txt với mã hóa UTF-8.
- Ghi toàn bộ danh sách kết quả vào file, mỗi phần nằm trên dòng riêng biệt.

```
# Ghi kết quả ra file
with open("top_3.txt", "w", encoding="utf-8") as f:
    f.write("\n".join(top_3_results))

```

II. Find the median for each statistic. Calculate the mean and standard deviation for each statistic across all players and for each team

1. Thư viện sử dụng

- pandas (pd): để xử lý dữ liệu dạng bảng (DataFrame).
- numpy (np): dùng để xác định kiểu số và tính toán thống kê.

```
import pandas as pd
import numpy as np

```

2. Đọc dữ liệu

- Đọc dữ liệu từ file results.csv vào DataFrame df.

```
# Đọc dữ liệu
df = pd.read_csv("results.csv")

```

3. Xác định các cột số (numeric) để thống kê

- select_dtypes(include=[np.number]): chọn tất cả cột có kiểu số.
- Loại bỏ 2 cột Age và Standard_Min vì không muốn tính thống kê trên đó.
- Kết quả là danh sách stat_cols chứa các cột thống kê bàn thắng, kiến

tạo, cứu thua,...

```
# Xác định các cột số
exclude_cols = ['Age', 'Standard_Min']
stat_cols = [col for col in df.select_dtypes(include=[np.number]).columns if col not in exclude_cols]

```

4. Tính trung vị (median), trung bình (mean) và độ lệch chuẩn (std) toàn bộ cầu thủ

- Tạo dictionary overall chứa thống kê cho toàn giải đấu (Team = 'all')
- Với mỗi cột:
    + Tính Median, Mean, Standard Deviation nếu cột không bị rỗng hoàn toàn (dropna().empty).
    + Nếu cột trống, gán = 0.

```
# Tính Median, Mean, Std
overall = {"Team": "all"}
for col in stat_cols:
    overall[f"Median of {col}"] = df[col].median(skipna=True) if not df[col].dropna().empty else 0
    overall[f"Mean of {col}"] = df[col].mean(skipna=True) if not df[col].dropna().empty else 0
    overall[f"Std of {col}"] = df[col].std(skipna=True) if not df[col].dropna().empty else 0

```

5. Tính thống kê tương tự theo từng đội bóng

- df.groupby("Team"): chia dữ liệu theo từng đội.
- Với mỗi đội:
    + Tạo một dòng row chứa các thống kê tương tự như phần toàn giải.
    + Lưu kết quả mỗi đội vào teams_stats (danh sách các dictionary).

```
# Tính theo từng đội
teams_stats = []
for team, group in df.groupby("Team"):
    row = {"Team": team}
    for col in stat_cols:
        row[f"Median of {col}"] = group[col].median(skipna=True) if not group[col].dropna().empty else 0
        row[f"Mean of {col}"] = group[col].mean(skipna=True) if not group[col].dropna().empty else 0
        row[f"Std of {col}"] = group[col].std(skipna=True) if not group[col].dropna().empty else 0
    teams_stats.append(row)

```

6. Gộp lại thành DataFrame

- Tạo bảng kết quả result_df gồm:
    + Dòng đầu tiên: thống kê toàn giải (overall).

+ Các dòng sau: thống kê từng đội trong teams_stats.

```
# Gộp tất cả kết quả vào DataFrame
result_df = pd.DataFrame([overall] + teams_stats)

```

7. Xử lý thiếu dữ liệu

- Thay toàn bộ giá trị NaN còn lại (nếu có) bằng 0.

```
# Thay thế toàn bộ NaN còn sót lại bằng 0
result_df.fillna(0, inplace=True)
```

8. Ghi kết quả ra file

- Xuất bảng kết quả result_df ra file CSV mới tên là results2.csv.
- index=False: không ghi chỉ số dòng vào file.

```
# Ghi ra file
result_df.to_csv("results2.csv", index=False)

```

III. Plot a histogram showing the distribution of each statistic for all players in the league and each team.

1. Thư viện sử dụng

- pandas (pd) : dùng để xử lý dữ liệu dạng bảng (DataFrame).
- numpy (np) : xử lý dữ liệu số (ở đây chỉ dùng gián tiếp).
- matplotlib.pyplot: dùng để vẽ biểu đồ.
- os: thao tác với thư mục (tạo thư mục lưu ảnh).
- re: xử lý chuỗi bằng biểu thức chính quy (lọc tên file hợp lệ)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
import re

```

2. Hàm làm sạch tên file

- Thay thế tất cả ký tự không phải chữ cái, số hoặc dấu gạch dưới trong chuỗi name bằng dấu _.
- Dùng để tạo tên file an toàn, tránh lỗi khi lưu file có dấu hoặc ký tự

đặc biệt.

```
# Hàm làm sạch tên file
def sanitize_filename(name):
    return re.sub(r'[^a-zA-Z0-9_]', '_', name)
```

3. Đọc dữ liệu và tạo thư mục lưu ảnh

- Đọc file results.csv xuất từ bài 1 chứa dữ liệu cầu thủ.
- Tạo thư mục 'histograms' để lưu ảnh biểu đồ, nếu chưa tồn tại.

```
df = pd.read_csv("results.csv")
os.makedirs("histograms", exist_ok=True)
```

4. Chọn ba chỉ số đại diện của tấn công và phòng thủ

- Tấn công: chọn 3 chỉ số từ bảng Shooting.
- Phòng ngự: chọn 3 chỉ số từ bảng Defense.
- Gộp lại tất cả vào selected_stats để duyệt chung.

```
# Chỉ lấy các cột cụ thể (3 tấn công và 3 phòng thủ)
attack_stats = {
    'shots_on_target_percent': ('Shooting_SoT%', float),
    'shots_on_target_per90': ('Shooting_SoT/90', float),
    'goals_per_shot': ('Shooting_G/Sh', float)
}

defense_stats = {
    'tackles': ('Defense_Tkl', int),
    'tackles_won': ('Defense_TklW', int),
    'dribblers_tackled': ('Defense_Att', int)
}
```

5. Vẽ biểu đồ cho tất cả cầu thủ

- Kết hợp tất cả các chỉ số cần vẽ.Lặp qua từng chỉ số đã chọn.
- df[col_name].dropna(): lấy dữ liệu cột và bỏ NaN.
- Dùng plt.hist() để vẽ biểu đồ tần suất (histogram).
- Đặt tiêu đề, nhãn trục, lưới, căn lề...
- Dùng sanitize_filename() để tạo tên file an toàn.

- Lưu ảnh vào thư mục histograms/.

```
# Từng đội
for team in df["Team"].dropna().unique():
    team_df = df[df["Team"] == team]
    for stat_name, (col_name, col_type) in selected_stats.items():
        plt.figure(figsize=(8, 5))
        plt.hist(team_df[col_name].dropna(), bins=20, color='lightcoral', edgecolor='black')
        plt.title(f"Histogram of {stat_name} ({col_name}) - {team}")
        plt.xlabel(stat_name)
        plt.ylabel("Frequency")
        plt.grid(True, linestyle='--', alpha=0.5)
        plt.tight_layout(pad=2.0)
        safe_col = sanitize_filename(col_name)
        safe_team = sanitize_filename(team)
        plt.savefig(f"histograms/{safe_col}_{safe_team}.png")
        plt.show()
        plt.close()
```

```
# Toàn giải
for stat_name, (col_name, col_type) in selected_stats.items():
    plt.figure(figsize=(8, 5))
    plt.hist(df[col_name].dropna(), bins=20, color='skyblue', edgecolor='black')
    plt.title(f"Histogram of {stat_name} ({col_name}) - All Players")
    plt.xlabel(stat_name)
    plt.ylabel("Frequency")
    plt.grid(True, linestyle='--', alpha=0.5)
    plt.tight_layout(pad=2.0)
    safe_col = sanitize_filename(col_name)
    plt.savefig(f"histograms/{safe_col}_all.png")
    plt.show()
    plt.close()

```

6. Vẽ biểu đồ cho từng đội

- Lặp qua từng đội bóng trong cột "Team".
- Lọc các cầu thủ của đội đó (team_df).
- Vẽ biểu đồ tương tự như trên, nhưng chỉ cho dữ liệu của đội đó. Lưu từng biểu đồ với tên dạng: histograms/{stat}_{team}.png.

IV. Identify the team with the highest scores for each statistic. Based on your analysis, which team do you think is performing the best in the 2024-2025 Premier League season?

1. Thư viện sử dụng

- pandas: Thư viện xử lý dữ liệu dạng bảng

```
import pandas as pd

```

2. Đọc dữ liệu

- Dùng pandas để đọc file CSV tên là results2.csv và lưu vào biến df (dataframe).

- df lúc này là bảng dữ liệu gồm các chỉ số của các đội bóng.

```
# Đọc dữ liệu từ file CSV
df = pd.read_csv('results2.csv')

```

3. Chỉ số được lựa chọn để đánh giá

- Đây là danh sách các cột tên chỉ số trong df (goals, xG, xAG, …).
- Mỗi chỉ số có hai đại lượng thống kê: Mean (trung bình) và Median (trung vị).

```
# Danh sách các chỉ số để đánh giá
chi_so_quan_trong = [
    'Median of Standard_Gls', 'Mean of Standard_Gls',
    'Median of Standard_Ast', 'Mean of Standard_Ast',
    'Median of Standard_xG', 'Mean of Standard_xG',
    'Median of Standard_xAG', 'Mean of Standard_xAG',
    'Median of Passing_Cmp', 'Mean of Passing_Cmp',
    'Median of Possession_Touches', 'Mean of Possession_Touches',
    'Median of Defense_Int', 'Mean of Defense_Int',
    'Median of Goalkeeping_Save%', 'Mean of Goalkeeping_Save%'
]

```

4. Tìm đội dẫn đầu cho từng chỉ số

- Vòng lặp đi qua từng chỉ số trong danh sách.
- df[chi_so].idxmax() tìm vị trí (index) của đội có giá trị cao nhất cho chỉ số đó.
- df.loc[... , 'Team']: lấy tên đội bóng ở dòng đó.
- Ghi lại đội đứng đầu cho từng chỉ số vào dictionary doi_dan_dau.

```
# Tìm đội dẫn đầu cho từng chỉ số
doi_dan_dau = {}
for chi_so in chi_so_quan_trong:
    doi_tot_nhat = df.loc[df[chi_so].idxmax(), 'Team']
    doi_dan_dau[chi_so] = doi_tot_nhat

```

5. Đếm số lần mỗi đội dẫn đầu

- Duyệt qua danh sách các đội đang dẫn đầu mỗi chỉ số.

- Ghi nhận mỗi đội xuất hiện bao nhiêu lần (tức là dẫn đầu bao nhiêu chỉ số).

```
# Đếm số lần mỗi đội dẫn đầu
diem_so_doi = {}
for doi in doi_dan_dau.values():
    if doi in diem_so_doi:
        diem_so_doi[doi] += 1
    else:
        diem_so_doi[doi] = 1

```

6. Xếp hạng đội theo số lần dẫn đầu

- Sắp xếp các đội theo điểm số (số lần dẫn đầu), giảm dần.

```
# Sắp xếp các đội theo điểm số
doi_xep_hang = sorted(diem_so_doi.items(), key=lambda x: x[1], reverse=True)

```

7. Hiển thị kết quả

- In ra đội đứng đầu theo từng chỉ số.
- In ra bảng xếp hạng theo số lần dẫn đầu.
- In ra đội có thành tích tổng quát tốt nhất (đội dẫn đầu nhiều chỉ số nhất).

```
# Hiển thị kết quả

for chi_so, doi in doi_dan_dau.items():
    print(f"{chi_so}: {doi}")


for doi, diem in doi_xep_hang:
    print(f"{doi}: {diem}")

```

Problem 3 :

A.Phân tích đề bài

- Sử dụng thuật toán K-means để phân loại cầu thủ dựa trên các chỉ số thống kê thi đấu.
- Tìm ra số nhóm tối ưu (k) để phân cụm.

- Đưa ra nhận xét về từng cụm sau khi phân nhóm
- Dùng PCA (Principal Component Analysis) để giảm số chiều (features) xuống 2.
- Vẽ biểu đồ scatter 2D, tô màu theo nhóm (Cluster) để trực quan hóa kết quả phân cụm.

B. Phân tích bài làm

1. Thư viện sử dụng

- pandas (pd) : Đọc và xử lý dữ liệu dạng bảng
- numpy (np) : Hỗ trợ toán học
- matplotlib (plt) , seaborn (sns) : vẽ biểu đồ
- sklearn.cluster (KMeans) : thuật toán phân cụm
- sklearn.preprocessing (StandardScaler): Chuẩn hóa dữ liệu
- sklearn.decomposition (PCA) : Giảm chiều dữ liệu
- sklearn.metrics (silhouette_score) : Đánh giá chất lượng phân cụm
- datetime , time , sys : Ghi log, tạo timestamp, redirect stdout

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import time
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.metrics import silhouette_score
from datetime import datetime
import sys
```

2. Đọc và xử lý dữ liệu

- Lấy toàn bộ dữ liệu số để đưa vào phân cụm. Xử lý thiếu dữ liệu.

```
# Đọc dữ liệu từ file
df = pd.read_csv("results.csv")
df_numeric = df.select_dtypes(include=[np.number])
df_numeric.dropna(axis=1, how='all', inplace=True)
df_numeric.fillna(0, inplace=True)
```

3. Chuẩn hóa dữ liệu

- Đưa các chỉ số về cùng thang đo (mean = 0, std = 1), cần thiết với KMeans.

```
# Chuẩn hoá dữ liệu
scaler = StandardScaler()
X_scaled = scaler.fit_transform(df_numeric)

```

4. Chọn số cụm tối ưu bằng Elbow và Silhouette

- Chạy KMeans với k = 2 đến 10:

+ inertia: dùng trong Elbow method

+ silhouette_score: dùng để đo độ chặt chẽ và tách biệt giữa các cụm

```
inertia = []
silhouette_scores = []
K_range = range(2, 11)

for k in K_range:
    kmeans = KMeans(n_clusters=k, random_state=42, n_init='auto')
    kmeans.fit(X_scaled)
    inertia.append(kmeans.inertia_)
    silhouette_scores.append(silhouette_score(X_scaled, kmeans.labels_))

```

5. Vẽ biểu đồ Elbow & Silhouette

- Chọn số cụm phù hợp nhất bằng hình ảnh.File lưu: "Elbow silhouette <timestamp>.png"

```
# Vẽ và lưu biểu đồ Elbow và Silhouette
plt.figure(figsize=(14, 5))
plt.subplot(1, 2, 1)
plt.plot(K_range, inertia, marker='o')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Inertia')
plt.title('Elbow Method')
plt.grid(True)

plt.subplot(1, 2, 2)
plt.plot(K_range, silhouette_scores, marker='o', color='orange')
plt.xlabel('Number of clusters (k)')
plt.ylabel('Silhouette Score')
plt.title('Silhouette Score')
plt.grid(True)

plt.tight_layout()
elbow_silhouette_file = f"Elbow silhouette {timestamp}.png"
plt.savefig(elbow_silhouette_file, dpi=300, bbox_inches='tight')
plt.close()
# Chọn số lượng nhóm
optimal_k = 4

```

6. Chọn k = 4 và phân cụm

- Gán cụm cho từng cầu thủ. df giờ có thêm cột "Cluster".

```
# Phân cụm với k=4
kmeans = KMeans(n_clusters=optimal_k, random_state=42, n_init='auto')
clusters = kmeans.fit_predict(X_scaled)
df['Cluster'] = clusters

```

7. PCA để giảm chiều và trực quan hóa

- Biến dữ liệu nhiều chiều về 2 chiều để vẽ biểu đồ cụm cầu thủ.

```
# Giảm chiều và trực quan hóa
pca = PCA(n_components=2)
X_pca = pca.fit_transform(X_scaled)

plt.figure(figsize=(12, 8))
scatter = sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['Cluster'], palette='Set2', s=80)

```

8. Vẽ biểu đồ phân cụm PCA

- Gán màu theo cụm
- Hiển thị tên các cầu thủ ghi bàn nhiều nhất (Top 5 theo Standard_Gls)
- File ảnh: "PCA Clusters <timestamp>.png"

```
plt.figure(figsize=(12, 8))
scatter = sns.scatterplot(x=X_pca[:, 0], y=X_pca[:, 1], hue=df['Cluster'], palette='Set2', s=80)

# Định nghĩa các nhóm
cluster_names = {
    0: "Hậu vệ/Phòng ngự",
    1: "Tiền vệ trung tâm",
    2: "Tiền đạo/Tấn công",
    3: "Tiền vệ cánh/Công"
}

```

9. Xuất file txt

- Ghi toàn bộ phần in (print) vào file thay vì ra màn hình.

```
log_file = f"Bài 3.txt"
original_stdout = sys.stdout
with open(log_file, 'w', encoding='utf-8') as f:
    sys.stdout = f
```

- Phân tích số lượng cầu thủ trong mỗi cụm
- Phân bố vị trí
- Nhận xét đặc điểm nổi bật thủ công
- Liệt kê 3 cầu thủ nổi bật nhất cụm (theo bàn thắng)

```
for cluster_num in range(optimal_k):
    cluster_data = df[df['Cluster'] == cluster_num]
    size = len(cluster_data)
    pos_dist = cluster_data['Pos'].value_counts(normalize=True).head(3)

    print(f"\nNHÓM {cluster_num}: {cluster_names[cluster_num]} ({size} cầu thủ)")
    print("Phân bố vị trí chính:")
    for pos, percent in pos_dist.items():
        print(f"   - {pos}: {percent:.1%}")
    print("\nĐặc điểm nổi bật:")
    if cluster_num == 0:
        print("   - Chỉ số phòng ngự cao (tackles, interceptions)")
        print("   - Ít tham gia tấn công (xG, assists thấp)")
    elif cluster_num == 1:
        print("   - Cân bằng giữa tấn công và phòng ngự")
        print("   - Chỉ số chuyền và kiểm soát bóng tốt")
    elif cluster_num == 2:
        print("   - Chỉ số tấn công vượt trội (xG, shots, goals)")
        print("   - Ít tham gia phòng ngự")
    else:
        print("   - Khả năng tạt bóng, dribble tốt")
        print("   - Tham gia cả tấn công lẫn phòng thủ")
    print("\nCầu thủ tiêu biểu:")
    top_in_cluster = cluster_data.sort_values('Standard_Gls', ascending=False).head(3)['Player']
    for i, player in enumerate(top_in_cluster, 1):
        print(f"   {i}. {player}")
sys.stdout = original_stdout
```

Problem 4:

A. Phân tích đề bài

- Thu thập giá trị chuyển nhượng cầu thủ mùa 2024–2025 từ footballtransfers.com
- Chỉ lấy cầu thủ chơi > 900 phút

B. Phân tích bài làm

I. Collect player transfer values for the 2024-2025 season from https://www.footballtransfers.com.

1. Thư viện sử dụng

- Sử dụng Pandas để xử lý dữ liệu dạng bảng (DataFrame).

```python
import pandas as pd

```

2. Đọc file dữ liệu

- Đọc dữ liệu từ file CSV chứa thông tin cầu thủ mùa giải.

```python
# Đọc dữ liệu từ file CSV
df = pd.read_csv('results.csv')

```

3. Xử lý dữ liệu thiếu

- Với cột số (như bàn thắng, kiến tạo): điền 0.
- Với cột chuỗi hoặc phần trăm: điền 'N/A'.

```python
df.fillna({

    'Standard_Gls': 0,
    'Standard_Ast': 0,
    'Standard_xG': 0,
    'Standard_xAG': 0,
    'Passing_Cmp': 0,
    'Defense_Tkl': 0,
    'Possession_Touches': 0,
    'Standard_Min': 0,
    'Standard_MP': 0,
    'Standard_Starts': 0,
    'Standard_90s': 0,


    'Shooting_SoT%': 'N/A',
    'Shooting_SoT': 'N/A',
    'Passing_Att': 'N/A',
    'Passing_Cmp%': 'N/A',
    'Defense_Int': 'N/A',
    'Defense_Blocks': 'N/A',
    'Possession_Prog': 'N/A',
    'Playing_Time_90s': 'N/A'
}, inplace=True)

```

- Điền 'N/A' cho tất cả cột còn lại có giá trị thiếu mà chưa được xử lý.

```
for column in df.columns:
    if df[column].isna().any():
        if column not in ['Standard_Gls', 'Standard_Ast', 'Standard_xG', 'Standard_xAG',
                          'Passing_Cmp', 'Defense_Tkl', 'Possession_Touches',
                          'Standard_Min', 'Standard_MP', 'Standard_Starts', 'Standard_90s']:
            df[column].fillna('N/A', inplace=True)
```

4. Hàm ước tính giá trị chuyển nhượng

Yếu tố	Ý nghĩa
base_value	Giá trị cơ bản theo vị trí: FW > MF > DF > GK
age_factor	Càng trẻ giá trị càng cao (tuổi > 20 sẽ giảm dần)
performance_factor	Dựa trên: bàn thắng, kiến tạo, xG, xAG (mỗi chỉ số có trọng số riêng)
minutes_factor	Dựa trên thời gian thi đấu (giới hạn tối đa là 2000 phút để cân bằng)
transfer_value	Tổng hợp tất cả nhân tố lại → làm tròn + không thấp hơn 0.5 triệu euro

```python
# Hàm ước tính giá trị chuyển nhượng
def estimate_transfer_value(row):
    base_value = 0

    # Giá trị cơ bản theo vị trí
    if 'FW' in row['Pos']:
        base_value = 15
    elif 'MF' in row['Pos']:
        base_value = 10
    elif 'DF' in row['Pos']:
        base_value = 8
    elif 'GK' in row['Pos']:
        base_value = 5
    else:
        base_value = 5


    age_factor = max(0.5, 1.5 - (row['Age'] - 20) * 0.03)


    performance_factor = 1 + (
        row['Standard_Gls'] * 0.2 +
        row['Standard_Ast'] * 0.15 +
        row['Standard_xG'] * 0.1 +
        row['Standard_xAG'] * 0.1
    ) / 10

```

5. Bổ sung thành phần

- Tính toán và gán giá trị chuyển nhượng cho từng cầu thủ.

```python

df['Transfer_Value'] = df.apply(estimate_transfer_value, axis=1)

```

- Thêm cột định dạng hiển thị.

```python
# Định dạng giá trị chuyển nhượng
df['Transfer_Value_Str'] = df['Transfer_Value'].apply(lambda x: f"€{x}M")

```

6. Xuất file dữ liệu

- Lưu kết quả vào file results4.csv.

```python
print(df[['Player', 'Team', 'Pos', 'Age', 'Standard_Gls', 'Standard_Ast', 'Transfer_Value']]
      .sort_values('Transfer_Value', ascending=False)
      .head(20))

# Lưu file
df.to_csv('results4.csv', index=False)
```

II. Note that only collect for the players whose playing time is greater than 900 minutes Propose a method for estimating player values. How do you select feature and model?

1. Thư viện sử dụng

- Pandas (pd) : dùng để xử lý dữ liệu bảng
- Path (pathlib) : giúp kiểm tra sự tồn tại của file

```
1  import pandas as pd
2  from pathlib import Path
3
```

2. Hàm đầu tiên

- Đọc file CSV tránh lỗi :
    + File không tồn tại
    + File bị rỗng
- Chi tiết trong hàm :
    + Nếu file không tồn tại, ném lỗi để xử lý trong except.
    + Đọc file CSV
    + Nếu file tồn tại nhưng rỗng, trả về default_data nếu có, hoặc một DataFrame() trống.
    + Nếu có bất kỳ lỗi nào xảy ra, in lỗi và trả về default_data (nếu có) hoặc DataFrame trống.

```python
def safe_read_csv(file_path, default_data=None):

    try:
        if not Path(file_path).exists():
            raise FileNotFoundError(f"{file_path}")

        data = pd.read_csv(file_path)

        if data.empty:
            return default_data if default_data is not None else pd.DataFrame()

        return data

    except Exception as e:
        print(f"{str(e)}")
        return default_data if default_data is not None else pd.DataFrame()

```

3. Hàm thứ 2

- Đọc file results4.csv. Nếu:
    + File không có
    + File bị lỗi
- In data

```python
data = safe_read_csv(
    'results4.csv',
    default_data=pd.DataFrame({'player': ['Demo'], 'goals': [0]})
)

print(data)
```