HỌC VIỆN CÔNG NGHỆ BƯU CHÍNH VIỄN THÔNG

BÀI TẬP LỚN
MÔN: LẬP TRÌNH PYTHON

HỌ TÊN: Đỗ Tiến Thành

MÃ SV: B23DCDT239

LỚP: D23CQCE04-B

Mục lục

1 YÊU CẦU ĐỀ BÀI

0.1 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**: expected goals (xG), expedted Assist Goals (xAG)
  - **Progression**: PrgC, PrgP, PrgR
  - **Per 90 minutes**: Gls, Ast, xG, xGA
  - **Goalkeeping**:
    - Performance: goals against per 90mins (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
  - **Miscellaneous Stats**:
    - Performance: Fls, Fld, Off, Crs, Recov
    - Aerial Duels: Won, Lost, Won

0.2 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	Std 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?

0.3 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.

0.4 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?

1 PHÂN TÍCH CÂU 1

* **Truy cập link github để có thể xem code tốt nhất:** https://github.com/DoTienThanh325/Big_exercise/tree/main/Problem%201

1.1 Ý tưởng làm bài

Truy cập từng url có số liệu của từng đội bóng một lấy các số liệu theo yêu cầu của đề bài. Mỗi url lấy số liệu ở 8 bảng:

- **Standard Stats**
- **Goalkeeping**
- **Shooting**
- **Passing**
- **Goal and Shot Creation**
- **Defensive Actions**
- **Possession**
- **Miscellaneous Stats**

Sau đó gộp các bảng của một đội bóng lại thành một chuyển các dữ liệu kiểu null -> 'N/a'. Làm tương tự với tất cả các đội bóng rồi ghép các bảng dữ liệu lại làm một.

1.2 Phân tích code câu 1

1.2.1 Hàm đặt lại tên cột

```python
import pandas as pd
from collections import Counter
import time
def deduplicate_columns(columns):
    counts = Counter()
    new_col = []
    for col in columns:
        if counts[col]:
            new_col.append(f"{col}_{counts[col]}")
        else:
            new_col.append(col)
        counts[col] += 1
    return new_col
```

- Sử dụng thư viện collections để đếm số lần xuất hiện của cột. - Duyệt qua tất cả các cột có trong dataframe kiểm tra số lần xuất hiện thông qua. counts[col]

- Nếu cột có counts[col] == 0 tức là lần đầu được duyệt -> đẩy vào list new_col.
- Nếu cột có counts[col] != 0 -> đưa vào new_col với tên mới là **tên côt + số lần xuất hiện**.

1.2.2 Hàm cài đặt Dataframe

- Tạo hàm crawl_data_table chuyền vào ba tham số là df(dữ liệu dataframe), list(các cột dữ liệu đề bài yêu cầu), club(tên câu lạc bộ).

```
def crawl_data_table(df,list,club):
    df.columns = df.columns.droplevel() if isinstance(df.columns, pd.MultiIndex) else df.columns
    df.columns = deduplicate_columns(df.columns)
    df['Team'] = club
    df = df[list]
    return df
```

- ***df.columns = df.columns.droplevel() if isinstance(df.columns, pd.MultiIndex) else df.columns***: câu lệnh xóa bỏ kiểu multiindex của col.

- Kiểm tra xem hàng các cột có phải kiểu multiindex không: ***isinstance(df.columns, pd.MultiIndex)***.
- Nếu df.columns là kiểu multiindex thì xóa bỏ ***df.columns.droplevel()*** không thì giữ nguyên.

- ***df.columns = deduplicate_columns(df.columns)***: Gọi hàm deduplicate_columns để đạt lại tên các cột do có các cột bị trùng tên.
- ***df['Team'] = club***: Thêm cột Team với giá trị là club.
- ***df = df[list]***: Cài đặt lại dataframe lấy các giá trị theo đề bài.

1.2.3 Hàm tạo Dataframe

- Cài đặt hàm trả về một dataframe

```
def create_team_table(url, club):
    table = pd.read_html(url)
    df_playerstats = crawl_data_table(table[0], ['Player', 'Nation', 'Team', 'Pos', 'Age', 'MP', 'Starts',
        'Min', 'Gls', 'Ast', 'CrdY', 'CrdR', 'xG', 'xAG', 'PrgC',
        'PrgP', 'PrgR', 'Gls_1', 'Ast_1', 'xG_1', 'xAG_1'], club)
    df_gk = crawl_data_table(table[2], ['Player', 'GA90', 'Save%', 'CS%', 'Save%_1'], club)
    df_shooting = crawl_data_table(table[4], ['Player', 'SoT%', 'SoT/90', 'G/Sh', 'Dist'], club)
    df_passing = crawl_data_table(table[5], ['Player', 'Cmp', 'Cmp%', 'TotDist', 'Cmp%_1', 'Cmp%_2', 'Cmp%_3', 'KP', '1/3', 'PPA', 'CrsPA', 'PrgP'], club)
    df_goalshot = crawl_data_table(table[7], ['Player', 'SCA', 'SCA90', 'GCA', 'GCA90'], club)
    df_defensive = crawl_data_table(table[8], ['Player', 'Tkl', 'TklW', 'Att', 'Lost', 'Blocks', 'Sh', 'Pass', 'Int'], club)
    df_possesion = crawl_data_table(table[9], ['Player', 'Touches', 'Def Pen', 'Def 3rd', 'Mid 3rd', 'Att 3rd', 'Att Pen', 'Att', 'Succ%', 'Tkld%', 'Carries', 'PrgDist', 'PrgC', '1/3
    df_miscellaneus = crawl_data_table(table[11], ['Player', 'Fls', 'Fld', 'Off', 'Crs', 'Recov', 'Won', 'Lost', 'Won%'], club)
    df = df_playerstats.merge(df_gk, on = 'Player', how = 'outer').merge(df_shooting,on = 'Player',how = 'outer').merge(df_passing, on = 'Player', how = 'outer').merge(df_goalshot,on
    df = df[df['Min'] > 90]
    df.fillna("N/a", inplace=True)
    df['first_name'] = df['Player'].apply(lambda x: x.split()[0])
```

- Sử dụng hàm **read_html()** trong thư viện pandas để đọc các bảng dữ liệu có trong url.
- Khởi tạo các dataframe ***df_playerstats, df_gk, df_shooting, df_passing, df_goalshot, df_defensive,***

df_possesion, df_miscellaneus bàng hàm ***crawl_data_table*** với vị trí bảng tương ứng trong url và các giá trị đề bài yêu cầu cùng tên club.

- Khởi tạo dataframe **df** là dataframe tổng của các dataframe trên bằng cách **merge** tất cả các dataframe trên lại dựa vào thuộc tính **on = 'Player'** (merge các dataframe dựa vào tên cầu thủ) và cách thức **how = 'outer'** (thêm các giá trị mà dataframe trước không có).
- **df = df[df['Min] > 90]**: Chỉ lấy các cầu thủ có thời gian trên 90 phút.
- **df.fillna("N/a", inplace=True)**: Thay thế tất cả các giá trị null bằng 'N/a'.

```
df['first_name'] = df['Player'].apply(lambda x: x.split()[0])
df = df.rename(columns={
    'MP': 'Playing_time: MP',
    'Starts': 'Playing_time: Starts',
    'Min': 'Playing_time: Min',
    'Gls': 'Performance: Gls',
    'Ast': 'Performance: Ast',
    'CrdY': 'Performance: CrdY',
    'CrdR': 'Performance: CrdR',
    'xG': 'Expected: xG',
    'xAG': 'Expected: xAG',
    'PrgC_x': 'Progression: PrgC',
    'PrgP_x': 'Progression: PrgP',
    'PrgR_x': 'Progression: PrgR',
    'Gls_1': 'Per_90_minutes: Gls',
    'Ast_1': 'Per_90_minutes: Ast',
    'xG_1': 'Per_90_minutes: xG',
    'xAG_1': 'Per_90_minutes: xAG',
    'GA90': 'Goalkeeping_Performance: GA90',
    'Save%': 'Goalkeeping_Performance: Save%',
    'CS%': 'Goalkeeping_Performance: CS%',
    'Save%_1': 'Goalkeeping_Penalty_Kicks: Save%',
    'Cmp': 'Passing_Total: Cmp',
    'Cmp%': 'Passing_Total: Cmp%',
    'TotDist': 'Passing_Total: TotDist',
    'Cmp%_1': 'Passing_Short: Cmp%',
    'Cmp%_2': 'Passing_Medium: Cmp%',
    'Cmp%_3': 'Passing_Long: Cmp%',
    'KP': 'Passing_Expected: KP',
    '1/3_x': 'Passing_Expected: 1/3',
    'PPA': 'Passing_Expected: PPA',
    'CrsPA': 'Passing_Expected: CrsPA',
    'PrgP_y': 'Passing_Expected: PrgP',
    'SCA': 'Goal_shot_creation: SCA',
    'SCA90': 'Goal_shot_creation: SCA90',
```

- Sử dụng hàm **rename** có sẵn trong thư viện pandas để cài đặt lại tên các cột.
- Tên các cột bằng **Tên các nhóm chỉ số + tên chỉ số**.

VD: chỉ số **Gls** trong nhóm chỉ số **Performance** -> **Performance: Gls**

- Cuối cùng hàm trả về dataframe hoàn chỉnh.

1.2.4 Viết chương trình trong hàm main

```
if __name__ == '__main__':
    df_liv = create_team_table('https://fbref.com/en/squads/822bd0ba/Liverpool-Stats', 'Liverpool')
    time.sleep(3)
    df_mu = create_team_table('https://fbref.com/en/squads/19538871/Manchester-United-Stats', 'Manchester United')
    time.sleep(3)
    df_ars = create_team_table('https://fbref.com/en/squads/18bb7c10/Arsenal-Stats', 'Arsenal')
    time.sleep(3)
    df_villa = create_team_table('https://fbref.com/en/squads/8602292d/Aston-Villa-Stats', 'Aston Villa')
    time.sleep(3)
    df_bou = create_team_table('https://fbref.com/en/squads/4ba7cbea/Bournemouth-Stats', 'Bournemouth')
    df_epl = pd.concat([df_liv, df_mu, df_ars, df_villa, df_bou], ignore_index=True)

    time.sleep(3)
    df_bren = create_team_table('https://fbref.com/en/squads/cd051869/Brentford-Stats', 'Brentford')
    time.sleep(3)
    df_brighton = create_team_table('https://fbref.com/en/squads/d07537b9/Brighton-and-Hove-Albion-Stats', 'Brighton & Hove Albion')
    time.sleep(3)
    df_chel = create_team_table('https://fbref.com/en/squads/cff3d9bb/Chelsea-Stats', 'Chelsea')
    time.sleep(3)
    df_crystal = create_team_table('https://fbref.com/en/squads/47c64c55/Crystal-Palace-Stats', 'Crystal Palace')
    time.sleep(3)
    df_ever = create_team_table('https://fbref.com/en/squads/d3fd31cc/Everton-Stats', 'Everton')
    df_epl = pd.concat([df_epl, df_bren, df_brighton, df_chel, df_crystal, df_ever], ignore_index=True)
```

```
time.sleep(3)
df_fulham = create_team_table('https://fbref.com/en/squads/fd962109/Fulham-Stats', 'Fulham')
time.sleep(3)
df_ips = create_team_table('https://fbref.com/en/squads/b74092de/Ipswich-Town-Stats', 'Ipswich Town')
time.sleep(3)
df_lei = create_team_table('https://fbref.com/en/squads/a2d435b3/Leicester-City-Stats', 'Leicester City')
time.sleep(3)
df_mc = create_team_table('https://fbref.com/en/squads/b8fd03ef/Manchester-City-Stats', 'Manchester City')
time.sleep(3)
df_new = create_team_table('https://fbref.com/en/squads/b2b47a98/Newcastle-United-Stats', 'Newcastle United')
df_epl = pd.concat([df_epl, df_fulham, df_ips, df_lei, df_mc, df_new], ignore_index=True)

time.sleep(3)
df_not = create_team_table('https://fbref.com/en/squads/e4a775cb/Nottingham-Forest-Stats', 'Nottingham Forest')
time.sleep(3)
df_sou = create_team_table('https://fbref.com/en/squads/33c895d4/Southampton-Stats', 'Southampton')
time.sleep(3)
df_tot = create_team_table('https://fbref.com/en/squads/361ca564/Tottenham-Hotspur-Stats', 'Tottenham Hotspur')
time.sleep(3)
df_wes = create_team_table('https://fbref.com/en/squads/7c21e445/West-Ham-United-Stats', 'West Ham United')
time.sleep(3)
df_wolves = create_team_table('https://fbref.com/en/squads/8cec06e1/Wolverhampton-Wanderers-Stats', 'Wolverhampton Wanderers')

df_epl = pd.concat([df_epl, df_not, df_sou, df_tot, df_wes, df_wolves], ignore_index=True)
df_epl = df_epl.sort_values(by = 'first_name').reset_index(drop = True)
df_epl = df_epl.drop(columns='first_name', errors='ignore')
df_epl = df_epl[df_epl['Player'] != 'Squad Total']
df_epl = df_epl[df_epl['Player'] != 'Opponent Total']
df_epl.to_csv('D:/file python/bài tập lớn/Problem 1/results.csv', index = True)
```

- Viết chương trình trong hàm main.
- Sử dụng hàm **create_team_table** để tạo dataframe cho từng đội bóng.
- Giữa mỗi lần lấy dữ liệu của đội bóng thêm **time.sleep(3)** để tránh bị web chặn.
- Khởi tạo ***df_epl*** bằng gộp các dataframe đội bóng bằng hàm **concat** trong thư viện pandas với **ignore_index=True** (khởi tạo lại index cho dataframe sau khi gộp).
- Chia nhỏ lần gửi request (5 đội bóng liên tiếp xong gộp vào với **df_epl** luôn để tránh bị web chặn request).
- Sau khi có dữ liệu của tất cả các cầu thủ trong dataframe **df_epl** ta thực hiện sắp xếp theo **first_name** có reset index sau khi sắp xếp: **df_epl = df_epl.sort_values(by = 'first_name') .reset_index(drop = True)**.
- Loại bỏ cột **first_name** và các hàng có 'Player' = 'Squad Total' và 'Opponent Total':

  - **df_epl = df_epl.drop(columns='first_name', errors='ignore')**

- **df_epl = df_epl[df_epl['Player'] != 'Squad Total']**
- **df_epl = df_epl[df_epl['Player'] != 'Opponent Total']**

- Cuối cùng lưu kết quả dataframe hoàn chỉnh vào file 'results.csv' bằng câu lệnh:
df_epl.to_csv('D:/file python/bài tập lớn/Problem 1/results.csv', index = True)

2 PHÂN TÍCH CÂU 2

* **Truy cập link github để có thể xem code tốt nhất:** https://github.com/DoTienThanh325/Big_exercise/tree/main/Problem%202

2.1 Câu 2 ý 1

2.1.1 Ý tưởng làm bài

Lấy dữ liệu từ bài 1 đưa vào một dataframe lọc ra các cột chỉ số dữ liệu dưới dạng number. Duyệt qua các cột mỗi cột sắp xếp lại theo thứ tự giảm dần của chỉ số lấy ra ba cầu thủ đầu lưu vào file theo yêu cầu đề bài sau đó lại sắp xếp theo thứ tự tăng dần của các chỉ số của cột đó và lấy ba cầu thủ đầu để đưa vào file

2.1.2 Phân tích code câu 2 ý 1

```python
import pandas as pd

df = pd.read_csv('Problem 1/results.csv', na_values='N/a')
df.drop(columns='Unnamed: 0', inplace=True)
numeric_col = df.select_dtypes(include='number').columns
index = 1
with open('Problem 2/top_3.txt', 'w', encoding='utf-8') as f:
    for col in numeric_col:
        f.write(f'{index}. {col}\n')
        f.write('Top 3: \n')
        top3 = df[['Player', col]].sort_values(by=col, ascending=False).head(3)
        f.write(top3.to_string(index=False) + '\n')

        f.write('Bottom 3: \n')
        bottom3 = df[['Player', col]].sort_values(by = col, ascending=True).head(3)
        f.write(bottom3.to_string(index=False) + '\n\n')
        index += 1
```

- Lấy dữ liệu câu 1 thu được đưa vào dataframe **df** chuyển tất cả các chuỗi 'N/a' về dạng na bằng câu lệnh: **df = pd.read_csv('Problem 1/results.csv', na_values='N/a')**.
- Loại bỏ cột 'Unnamed: 0' (Là cột index lưu từ câu 1): **df.drop(columns='Unnamed: 0', inplace=True)**.
- Lấy ra list các cột có dữ liệu dạng số lưu vào **numeric_col**:
numeric_col = df.select_dtypes(include='number').columns - Khởi tạo biến index bằng 1 để đánh dấu index các cột chỉ số.
- Mở file để lưu dữ liệu phân tích vào:
with open('Problem 2/top_3.txt', 'w', encoding='utf-8') as f:.
- Dùng vòng **for** duyệt qua từng cột một, tại mỗi cột:

- Tìm ra ba cầu thủ có chỉ số cao nhất bằng cách sắp xếp dựa vào chỉ số của cột theo chiều

giảm dần và đưa vào file:

```
f.write(f'{index}. {col}\n')
f.write('Top 3: \n')
top3 = df[['Player', col]].sort_values(by=col, ascending=False).head(3)
f.write(top3.to_string(index=False) + '\n')
```

- Tìm ra ba cầu thủ có chỉ số thấp nhất bằng cách sắp xếp dựa vào chỉ số của cột theo chiều tăng dần và đưa vào file:

```
f.write('Bottom 3: \n')
bottom3 = df[['Player', col]].sort_values(by = col, ascending=True).head(3)
f.write(bottom3.to_string(index=False) + '\n\n')
```

- Sau cột ta lại tăng biến index lên một đơn vị.

2.2 Câu 2 ý 2

2.2.1 Ý tưởng làm bài

Lấy dữ liệu từ bài 1 đưa vào một dataframe lọc ra các cột chỉ số dữ liệu dưới dạng number. Tạo dictionary all_team để lưu trữ dữ liệu của toàn bộ các đội bóng và có 'Team': 'all' duyệt qua các cột dữ liệu bên trên tính median, mean, std của từng cột và đưa vào dictionary với keys phù hợp sau đó đẩy dictionary đó vào một list tổng. Với từng câu lạc bộ ta lấy tất cả cầu thủ thuộc đội bóng đưa vào một dataframe và làm tương tự như toàn giải đấu sau đó cũng đưa vào list tổng. Cuối cùng chuyển list tổng sang dạng dataframe và đẩy vào file csv.

2.2.2 Phân tích code câu 2 ý 2

```
import pandas as pd

df = pd.read_csv('Problem 1/results.csv', na_values='N/a')
df.drop(columns='Unnamed: 0', inplace=True)
numeric_col = df.select_dtypes(include='number').columns

epl_clb = []
all_team = {'Team': 'all'}
for col in numeric_col:
    all_team[f'Median of {col}'] = df[col].median()
    all_team[f'Mean of {col}'] = df[col].mean()
    all_team[f'Std of {col}'] = df[col].std()
epl_clb.append(all_team)
```

- Lấy dữ liệu của file csv và chỉnh sửa giống ý 1.
- Khởi tạo một list rỗng: **epl_clb = []** - Tạo một dictionary có 1 keys là **Team** và values là **all**: **all_team = {'Team': 'all'}**.
- Duyệt qua từng cột của **numeric_col**, mỗi cột tạo key và value tương ứng cho dictionary **all_team**:

```
all_team[f'Median of {col}'] = df[col].median()
all_team[f'Mean of {col}'] = df[col].mean()
all_team[f'Std of {col}'] = df[col].std()
```

- Thêm dictionary all_team vào list epl_clb: **epl_clb.append(all_team)**

```
for team in df['Team'].unique():
    club = {'Team': team}
    df_team = df[df['Team'] == team]
    for col in numeric_col:
        club[f'Median of {col}'] = df_team[col].median()
        club[f'Mean of {col}'] = df_team[col].mean()
        club[f'Std of {col}'] = df_team[col].std()
    epl_clb.append(club)
epl_df = pd.DataFrame(epl_clb)
epl_df.to_csv('Problem 2/results2.csv', index = True)
```

- **for team in df['Team'].unique()::** Duyệt qua từng câu lạc bộ một.

- Mỗi câu lạc bộ tạo một dictionary là **club** với cặp key value đầu là **'Team': team** và tạo ra một dataframe lưu trữ toàn bộ cầu thủ của câu lạc bộ đó:

```
club = {'Team': team}
df_team = df[df['Team'] == team]
```

- Duyệt qua từng cột và làm tương tự như toàn giải

```
for col in numeric_col:
    club[f'Median of {col}'] = df_team[col].median()
    club[f'Mean of {col}'] = df_team[col].mean()
    club[f'Std of {col}'] = df_team[col].std()
epl_clb.append(club)
```

- Cuối cùng đưa list về dạng dataframe và lưu vào file csv:

```
epl_df = pd.DataFrame(epl_clb)
epl_df.to_csv('Problem 2/results2.csv', index = True)
```

2.3 Câu 2 ý 3

2.3.1 Ý tưởng làm bài

Lấy những dữ liệu cột dưới dạng số sau đó dùng thư viện matplotlib để vẽ. Dùng subplots để có thể vẽ nhiều biểu đồ cùng một lúc

2.3.2 Phân tích code câu 2 ý 3

```
import pandas as pd
import matplotlib.pyplot as plt
df = pd.read_csv('Problem 1/results.csv', na_values='N/a')
df.drop(columns='Unnamed: 0', inplace=True)
numeric_col = df.select_dtypes(include='number').columns

for col in numeric_col:
    if df[col].isna().all():
        continue
    teams = df['Team'].unique()
    len_teams = len(teams)
    fig, axes = plt.subplots(3, 7, figsize = (15, 10))
    axes = axes.flatten() if len_teams > 1 else [axes]

    axes[0].hist(df[col].dropna(), bins = 30, color='blue', edgecolor='black')
    axes[0].set_title(f'Epl {col}', fontsize=7)

    for i,team in enumerate(teams[:len(axes)-1], 1):
        df_team = df[df['Team'] == team][col].dropna()
        if len(df_team) > 1:
            axes[i].hist(df_team, bins=30, color='green', edgecolor='black')
            axes[i].set_title(f'{team} {col}', fontsize=7)
    plt.tight_layout()
    plt.show()
    plt.close()
```

- Đọc và xử lý dữ liệu như hai ý trên của câu 2.
- Duyệt qua các cột:

  - Nếu cột toàn **na** ta sẽ chuyển sang cột khác:

    ```
    for col in numeric_col:
        if df[col].isna().all(): continue
    ```

  - **teams = df['Team'].unique()**: Lấy danh sách các đội bóng không trùng trong cột 'Team'.
  - **fig, axes = plt.subplots(3, 7, figsize=(15, 10))**: Tạo ra một figure với 3 hàng, 7 cột ô (tức là 21 ô) để vẽ các biểu đồ nhỏ (subplots).

- **axes = axes.flatten() if len_teams > 1 else [axes]**: Vì axes ban đầu là một mảng 2D (3x7), nên bẻ phẳng thành 1D list cho dễ lặp. Còn nếu chỉ có 1 team thì axes không phải mảng, nên cho vào list để tránh lỗi.

- **axes[0].hist(df[col].dropna(), bins = 30, color='blue', edgecolor='black')**: Vẽ biểu đồ tần suất của cả giải trên subplot đầu tiên **axes[0]**. **dropna()** để loại bỏ giá trị NaN (trống) trước khi vẽ. **bins=30**: chia thành 30 cột nhỏ. **color='blue'**: màu bên trong cột. **edgecolor='black'**: viền cột màu đen.

- **axes[0].set_title(f'Epl col', fontsize=7)**: Gắn tựa đề cho subplot. . **fontsize=7**: cỡ chữ cho tiêu đề.

- **enumerate(teams[:len(axes)-1], 1)**: Bắt đầu từ i = 1 (vì axes[0] đã dùng để vẽ EPL tổng). teams[:len(axes)-1]: tránh bị lỗi nếu số đội > số subplot.

- **df_team = df[df['Team'] == team][col].dropna()**: Lọc ra dữ liệu cột col của từng đội. **dropna()** để không vẽ các giá trị thiếu.

- Đoạn code:

  ```
  if len(df_team) > 1:
      axes[i].hist(df_team, bins=30, color='green', edgecolor='black')
      axes[i].set_title(f'{team} {col}', fontsize=7)
  ```

  kiểm tra **len(df_team)** > 1 không để đảm bảo không lỗi. Hai câu sau giống như cách vẽ biểu đồ tổng khác ở màu của cột là màu xanh lá cây.

- **plt.tight_layout()**: Loại bỏ các biểu đồ rỗng.

- **plt.show()**: Hiển thị các biểu đồ khi run code.

- **plt.close()**: Xóa figure đã vẽ xong ra khỏi bộ nhớ.

2.4 Câu 2 ý 4

2.4.1 Ý tưởng làm bài

Tính trung bình cộng tất cả các chỉ số dưới dạng số liệu chọn ra đội đứng đầu ở nhiều nhất các chỉ số là đội có màn trình diễn tốt nhất.

2.4.2 *Phân tích code câu 2 ý 4*

```
import pandas as pd
from collections import Counter
df = pd.read_csv('Problem 1/results.csv', na_values='N/a')
df.drop(columns='Unnamed: 0', inplace=True)
numeric_col = df.select_dtypes(include='number').columns

team_high = {}
for col in numeric_col:
    if df[col].isna().all():
        continue
    team = df.groupby('Team')[col].mean()
    top_team = team.idxmax()
    top_value = team.max()
    team_high[col] = (top_team, top_value)

print("Team with highest score for each statistic:")
for stat, (team, value) in team_high.items():
    print(f"{stat}: {team} ({value})")

team_counts = Counter(team for team, _ in team_high.values())
highest = -10**9
for team, count in team_counts.most_common():
    if count > highest:
        highest = count
        best_peformance_team = team
print()
print('The Best-Peforming team in 2024-2025 Premier League Season:', best_peformance_team)
```

- Xử lý dữ liệu như các ý trên của câu 2.
- **for col in numeric_col:**: Duyệt qua các cột giá trị.
- **if df[col].isna().all(): continue**: Cột nào toàn giá trị NaN thì chuyển qua cột khác.
- **team = df.groupby('Team')[col].mean()**: Sử dụng hàm **groupby()** và hàm **mean()** để tính trung bình cộng các chỉ số của từng đội.
- **top_team = team.idxmax()**: Dùng hàm **idxmax()** tìm ra đội có chỉ số trung bình cộng cao nhất.
- **top_value = team.max()**: Dùng hàm **max()** tìm ra chỉ số cao nhất.
- **team_high[col] = (top_team, top_value)**: Dùng dictionary để lưu trữ với key là tên cột và value là một tuple gồm hai giá trị **top_team** và **top_value**.
- Đoạn code:

```
print("Team with highest score for each statistic:")
for stat, (team, value) in team_high.items():
    print(f"{stat}: {team} ({value})")
```

- Dùng để in các đội đứng đầu tại chỉ số và chỉ số đó

- **team_counts = Counter(team for team, _ in team_high.values())**: Tạo mảng đếm để đếm xem mỗi đội đứng đầu mỗi chỉ số bao lần, đội đứng đầu nhiều nhất là đội có best-peforming.
- Đoạn code:

```
highest = -10**9
for team, count in team_counts.most_common():
    if count > highest:
        highest = count
        best_peformance_team = team
```

- Đoạn code này dùng để tìm đội có số lần đứng đầu và gán cho biến **best_peformance_team** là đội bóng có số lần đứng đầu nhiều nhất.

3 PHÂN TÍCH CÂU 3

* **Truy cập link github để có thể xem code tốt nhất:** https://github.com/DoTienThanh325/Big_exercise/tree/main/Problem%203

3.1 Ý tưởng làm bài

Sử dụng thuật toán K-means để phân chia các nhóm cầu thủ sử dụng phương pháp Elbow để tìm số cụm phân chia. Sử dụng PCA để giảm chiều dữ liệu xuống và vẽ lại biểu đồ phân cụm.

3.2 Phân tích code câu 3

```python
import pandas as pd
from sklearn.cluster import KMeans
from sklearn.preprocessing import StandardScaler
import matplotlib.pyplot as plt
from kneed import KneeLocator
from sklearn.decomposition import PCA
df = pd.read_csv('Problem 1/results.csv')
columns_str = ['Unnamed: 0', 'Player', 'Nation', 'Team', 'Pos', 'Age']
df_2 = df.drop(columns=columns_str, errors='ignore')
df_2 = df_2.replace('N/a', 0)
df_2 = df_2.fillna(0)

scaler = StandardScaler()
scaled_features = scaler.fit_transform(df_2)

inertias = []
k_range = range(1,11)

for k in k_range:
    k_means = KMeans(n_clusters=k, random_state=42)
    labels = k_means.fit_predict(scaled_features)
    inertias.append(k_means.inertia_)

plt.figure(figsize=(12,5))
plt.plot(k_range, inertias, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.grid(True)
plt.savefig('Problem 3/elbow_method.png')

elbow_pos = KneeLocator(k_range, inertias, curve='convex', direction='decreasing')
k = elbow_pos.knee
print(f'Điểm elbow chon được là: {k}')
print(f'Chọn {k} là do từ {k - 1} đến {k} thì inertia giảm mạnh còn từ {k} đến {k+1} và sau nữa độ giảm inertia không còn lớn như trước')
```

- Đoạn code:

```
columns_str = ['Unnamed: 0', 'Player', 'Nation', 'Team', 'Pos', 'Age']
df_2 = df.drop(columns=columns_str, errors='ignore')
df_2 = df_2.replace('N/a', 0)
df_2 = df_2.fillna(0)
```

Xử lý dữ liệu gần giống câu 2 nhưng không trực tiếp lấy cột có dạng số mà loại bỏ các cột có dạng chữ. Thay tất cả 'N/a' và giá trị NaN sang giá trị 0.
- Chuẩn hóa dữ liệu để áp dụng các hàm sẵn có cho thuật toán k_means:

```
    scaler = StandardScaler()
    scaled_features = scaler.fit_transform(df_2)
```

- Khởi chạy đoạn code theo phương pháp elbow:

```
inertias = []
k_range = range(1,11)

for k in k_range:
    k_means = KMeans(n_clusters=k, random_state=42)
    labels = k_means.fit_predict(scaled_features)
    inertias.append(k_means.inertia_)
```

- Khởi tạo danh sách **inertias**: Dùng để lưu trữ giá trị inertia cho mỗi số lượng cụm k.
- Lặp qua các giá trị **k** từ 1 đến 10 (với **range(1,11)**):
  - Đối với mỗi giá trị **k**, ta tạo một đối tượng **KMeans** và áp dụng phương thức **fit_predict** để phân cụm dữ liệu.
  - **fit_predict** sẽ trả về labels (nhãn cụm cho từng điểm dữ liệu) và cập nhật đối tượng **k_means**.
- Lưu giá trị inertia vào danh sách **inertias**: Inertia là tổng bình phương khoảng cách từ các điểm đến tâm của cụm gần nhất, được tính qua **k_means.inertia_**.

- Tiếp theo thực hiện vẽ biểu đồ Elbow method với trục x là số nhóm còn y là Inertia, có thể dựa vào biểu đồ tìm elbow hoặc có thể dùng thư viện knee để chính xác nhất:

```
plt.figure(figsize=(12,5))
plt.plot(k_range, inertias, marker='o')
plt.title('Elbow Method')
plt.xlabel('Number of clusters')
plt.ylabel('Inertia')
plt.grid(True)
plt.savefig('Problem 3/elbow_method.png')

elbow_pos = KneeLocator(k_range, inertias, curve='convex',direction=
'decreasing')
k = elbow_pos.knee
print(f'Điểm elbow chon được là: {k}')
```

```
target_name = ['Group_' + str(x) for x in range(1,k+1)]
k_means = KMeans(n_clusters=k, random_state=42)
cluster = k_means.fit_predict(scaled_features)
pca = PCA(n_components=2)
pca_result = pca.fit_transform(scaled_features)
plt.figure(figsize=(12,8))
for i in range(k):
    plt.scatter(
        pca_result[cluster == i, 0],
        pca_result[cluster == i, 1],
        label=target_name[i],
        cmap='viridis',
        s=50
    )
plt.legend(loc='best', shadow=False, scatterpoints=1)
plt.title('PCA of Epl player in 2024 - 2025 season')
plt.savefig('Problem 3/cluster_pca_2d.png')
```

- Tạo tên nhóm cụm (**target_name**):

  - Tạo một danh sách tên nhóm có dạng [’Group_1’, ’Group_2’, ..., ’Group_k’], với k là số cụm.

- Áp dụng **KMeans**:

  - **k_means = KMeans(n_clusters=k, random_state=42)**: Khởi tạo đối tượng KMeans với số cụm **k**.
  - **cluster = k_means.fit_predict(scaled_features)**: Dự đoán nhóm cho các mẫu trong **scaled_features** và lưu kết quả vào cluster.

- Áp dụng PCA:

  - **pca = PCA(n_components=2)**: Khởi tạo đối tượng PCA để giảm chiều dữ liệu xuống 2 chiều.
  - **pca_result = pca.fit_transform(scaled_features)**: Áp dụng PCA lên dữ liệu chuẩn hóa **scaled_features**.

- Vẽ biểu đồ phân cụm:

  - **plt.scatter()**: Vẽ các điểm trên biểu đồ 2D với màu sắc khác nhau cho từng nhóm cụm (cluster == i).
  - **label=target_name[i]**: Đặt tên cho mỗi nhóm tương ứng với các cụm.
  - **plt.legend()**: Thêm chú thích vào biểu đồ.
  - **plt.savefig()**: Lưu biểu đồ vào tệp **cluster_pca_2d.png**.

4 PHÂN TÍCH CÂU 4

* **Truy cập link github để có thể xem code tốt nhất:** https://github.com/DoTienThanh325/Big_exercise/tree/main/Problem%204

4.1 Câu 4 ý 1

4.1.1 Ý tưởng làm bài

Sử dụng selenium để lấy dữ liệu của trang web, chuẩn hóa tên thu được sao cho dữ liệu tên mới thu được khớp với tên cầu thủ trong từ bài một thu được để đối chiếu thời gian thi đấu cầu thủ.

4.1.2 Phân tích code câu 4 ý 1

```
def crawl_data(url):
    chromedriver_path = "D:\chromedriver-win64\chromedriver-win64\chromedriver.exe"
    try:
        service = Service(chromedriver_path)
        options = webdriver.ChromeOptions()
        options.add_argument('--headless')
        options.add_argument('--user-agent=Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/91.0.4472.124 Safari/5
        driver = webdriver.Chrome(service=service, options=options)
        driver.get(url)
        time.sleep(3)
        soup = bs(driver.page_source, 'html.parser')
        table = soup.find('table')

        if table:
            data = []
            rows = table.find_all('tr')

            for row in rows:
                player = row.find_all('span', attrs={'class': 'd-none'})
                price = row.find_all('td', attrs={'class': 'text-center'})
                price = [p.text.strip() for p in price]
                player = [p.text.strip() for p in player]
                cols = player + price
                if cols:
                    data.append(cols)
            data = data[2:]
            df = pd.DataFrame(data)
            return df
        driver.quit()

    except Exception as e:
        print(f"Đã xảy ra lỗi: {e}")
        driver.quit()
```

- Hàm **crawl_data**() Áp dụng Selenium:

- **chromedriver_path** : Khởi tạo biến đường dẫn chromedriver.
- **service = Service(chromedriver_path)**: Khởi tạo một đối tượng Service để chỉ định đường dẫn đến trình điều khiển chromedriver (tệp cần thiết để Selenium tương tác với trình duyệt Chrome).
- **webdriver.ChromeOptions**(): Tạo một đối tượng **ChromeOptions** để thiết lập các tùy chọn cho Chrome.
    - **–headless**: Chạy Chrome mà không mở giao diện người dùng (trình duyệt sẽ chạy ở chế độ ẩn).

    - **–user-agent=Mozilla/...**: Thiết lập User-Agent để giả lập trình duyệt Chrome, tránh bị phát hiện là bot.

- **driver = webdriver.Chrome(service=service, options=options)**: Khởi tạo trình duyệt Chrome với các tùy chọn đã thiết lập (bao gồm chế độ headless và user-agent).
- **driver.get(url)**: Truy cập vào URL.
- **time.sleep(3)**: Đợi 3 giây để trang web tải xong trước khi thu thập dữ liệu.
- **soup = bs(driver.page_source, 'html.parser')**: Lấy nội dung trang web (HTML) và phân tích nó bằng BeautifulSoup.
- **table = soup.find('table')**: Tìm thẻ <table> đầu tiên trong nội dung trang.
- **if table**:: Kiểm tra nếu bảng dữ liệu tồn tại trên trang.
- **rows = table.find_all('tr')**: Tìm tất cả các thẻ <tr> trong bảng, mỗi thẻ tương ứng với một hàng trong bảng.
- Lấy dữ liệu cầu thủ và giá chuyển nhượng:
    - **player = row.find_all('span', attrs='class': 'd-none')**: Lấy các phần tử **** có class **'d-none'**.
    - **price = row.find_all('td', attrs='class': 'text-center')**: Lấy các thẻ **<td>** có class **'text-center'**.
- **price = [p.text.strip() for p in price]** và **player = [p.text.strip() for p in player]**: Loại bỏ khoảng trắng thừa từ các giá trị lấy được.
- **cols = player + price**: Kết hợp thông tin người chơi và giá vào một danh sách cols
- **if cols:**: Kiểm tra nếu danh sách cols không rỗng trước khi thêm vào data.
- **df = pd.DataFrame(data)**: Chuyển danh sách data thành một DataFrame của pandas.
- Sử dụng **try** và **except** để phòng trường hợp bị lỗi.

- Hàm chuẩn hóa tên cầu thủ: Do lấy dữ liệu từ hai trang web khác nhau nên tên cầu thủ không được giống nhau nên ta cần chuẩn hóa để cầu thủ hai nguồn dữ liệu khớp nhau:

```python
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from bs4 import BeautifulSoup as bs
import pandas as pd
import time
import unicodedata
def remove_diacritics(text):
    if pd.isna(text):
        return ''
    return ''.join(
        c for c in unicodedata.normalize('NFD', text)
        if unicodedata.category(c) != 'Mn'
    )
```

- **if pd.isna(text): return ”**: Kiểm tra nếu đầu vào là giá trị NaN của pandas, trả về một chuỗi rỗng. Điều này giúp tránh lỗi khi hàm nhận giá trị NaN.
- **unicodedata.normalize(’NFD’, text)**: Sử dụng thư viện **unicodedata** của Python để chuẩn hóa chuỗi văn bản theo dạng "NFD"(Normalization Form Decomposed). Trong NFD, các ký tự có dấu được tách thành hai phần: ký tự cơ bản và dấu phụ. Ví dụ, chữ "é"sẽ được tách thành "e"và dấu sắc.
- **unicodedata.category(c) != ’Mn’**: Kiểm tra xem ký tự **c** có phải là dấu phụ hay không. **Mn** là mã phân loại cho các dấu phụ trong Unicode, và bạn loại bỏ các ký tự này trong quá trình ghép chuỗi.
- **”.join(c for c in ...)**: Kết hợp các ký tự lại thành một chuỗi mà không chứa dấu phụ.

```python
df = crawl_data('https://www.footballtransfers.com/us/players/uk-premier-league')
for i in range(2,23):
    url = 'https://www.footballtransfers.com/us/players/uk-premier-league' + '/' + str(i)
    df = pd.concat([df, crawl_data(url)],ignore_index=False)
df = df[[0, 2]]
df = df.rename(columns={
    0: 'Player',
    2: 'Price',
})
df['Player'] = df['Player'].replace('Rasmus Winther Højlund', 'Rasmus Højlund')
df['Player'] = df['Player'].replace('Idrissa Gueye', 'Idrissa Gana Gueye')
df['Player'] = df['Player'].replace('Manuel Ugarte', 'Manuel Ugarte Ribeiro')
df['Player'] = df['Player'].replace('Omari Giraud-Hutchinson', 'Omari Hutchinson')
df['Player'] = df['Player'].replace('Rayan Aït Nouri', 'Rayan Ait-Nouri')
df['Player'] = df['Player'].replace('Heung-min Son', 'Son Heung-min')
df['Player'] = df['Player'].replace('Victor Kristiansen', 'Victor Bernth Kristiansen')
df['Player'] = df['Player'].apply(remove_diacritics)
df_epl = pd.read_csv('Problem 1/results.csv', na_values='N/a')
df_epl.drop(columns='Unnamed: 0', inplace=True)
df_epl['Player'] = df_epl['Player'].apply(remove_diacritics)
df = df.merge(df_epl, on='Player', how='outer')
df = df[df['Playing_time: Min'] > 900]
df = df[['Player', 'Price', 'Playing_time: Min']]
df.dropna(subset=['Price'],inplace=True)
df.reset_index(drop=True, inplace=True)
df.to_csv('D:/file python/bài tập lớn/Problem 4/results.csv',index = True)
```

- Chạy chương trình trong hàm main:

- Lấy hết dữ liệu từ 22 url.
- **df = df[[0, 2]]**: Chỉ lấy hai cột chỉ số 0 và 2.
- **df = df.rename(columns=0: 'Player', 2: 'Price')**: Đổi tên cột thành 'Player' và 'Price'.
- **df['Player'] = df['Player'].replace(...)**: Thực hiện thay thế tên một số cầu thủ đặc biệt sao khớp với dữ liệu câu 1.
- **df['Player'] = df['Player'].apply(remove_diacritics)**: Áp dụng hàm **remove_diacritics** để loại bỏ dấu phụ, chuẩn hóa tên.
- **df_epl = pd.read_csv('Problem 1/results.csv', na_values='N/a')**: Đọc dữ liệu thu được từ câu 1.
- **df_epl.drop(columns='Unnamed: 0', inplace=True)**: Xóa cột **'Unnamed: 0'** không cần thiết.
- **df_epl['Player'] = df_epl['Player'].apply(remove_diacritics)**: Áp dụng hàm **remove_diacritics** cho cột 'Player' của tệp CSV.
- **df = df.merge(df_epl, on='Player', how='outer')**: Kết hợp dữ liệu từ hai DataFrame (**df và df_epl**) theo cột **'Player'**, dùng kiểu kết hợp **'outer'** để giữ tất cả các cầu thủ trong cả hai DataFrame.
- **df = df[df['Playing_time: Min'] > 900]**: Lọc ra các cầu thủ có thời gian chơi hơn 900 phút.
- **df = df[['Player', 'Price', 'Playing_time: Min']]**: Chọn chỉ các cột **'Player'**, **'Price'**, và **'Playing_time: Min'**.
- **df.dropna(subset=['Price'], inplace=True)**: Loại bỏ các dòng có giá trị 'Price' là NaN.
- **df.reset_index(drop=True, inplace=True)**: Đặt lại index của DataFrame.
- **df.to_csv('D:/file python/bài tập lớn/Problem 4/results.csv', index=True)**: Lưu lại DataFrame hoàn chỉnh.

4.2 Câu 4 ý 2

4.2.1 Ý tưởng làm bài

Xử lý hai bảng giá trị đã có ở câu 4 ý 1 và câu 1 sao cho khớp dữ liệu sau đó huấn luyện mô hình theo mô hình LogisticRegression.

4.2.2 Phân tích code câu 4 ý 2

- **Bước 1:** Xứ lý dữ liệu cầu thủ và giá tiền:

```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.preprocessing import StandardScaler
from sklearn.metrics import  classification_report, confusion_matrix
import matplotlib.pyplot as plt
import unicodedata
import joblib
def remove_diacritics(text):
    if pd.isna(text):
        return ''
    return ''.join(
        c for c in unicodedata.normalize('NFD', text)
        if unicodedata.category(c) != 'Mn'
    )

# Xử lý dữ liệu
df = pd.read_csv('Problem 4/results.csv')
df.drop(columns='Unnamed: 0', inplace=True)
df = df[df['Price'] != 'N/a']
df['Price'] = df['Price'].apply(lambda x: x[1:len(x)-1])
df_epl = pd.read_csv('Problem 1/results.csv')
df_epl.drop(columns='Unnamed: 0', inplace=True)
df_epl['Player'] = df_epl['Player'].apply(remove_diacritics)
df_epl = df_epl[df_epl['Player'].isin(df['Player'])]
df_epl = df_epl[df_epl['Playing_time: Min'] > 900]
df = df.merge(df_epl, on = ['Player', 'Playing_time: Min'], how='outer')
df.to_csv('cauthu.csv', index = True)
df['Price'] = df['Price'].astype(float)
numeric_col = df.select_dtypes(include='number').columns
numeric_col = numeric_col.drop(['Performance: CrdY', 'Performance: CrdR', 'Price'])
```

- Đoạn code trên lấy dữ liệu từ hai file csv của câu 1 và ý 1 câu 4 xử lý lại tên sao cho khớp dữ liệu sau đó gộp các thuộc tính cầu thủ lại. Thu lấy các cột chỉ số có dạng number để phân tích dự đoán giá cầu thủ. Loại bỏ các chỉ số như giá cầu thủ, thẻ đỏ, thẻ vàng vì không ảnh hưởng tới giá của cầu thủ.

- **Bước 2:** Huấn luyện mô hình:

```python
#Phân nhóm giá cầu thủ
try:
    df['Price'] = pd.qcut(df['Price'], q = 3, labels=['Low', 'Medium', 'High'])
except:
    print('Lỗi phân chia giá cầu thủ')
    exit()

x = df[numeric_col].fillna(0)
y = df['Price']

scaler = StandardScaler()
x_scaled = scaler.fit_transform(x)

# Chia dữ liệu thành tập huấn luyện và kiểm tra
x_train, x_test, y_train, y_test = train_test_split(x_scaled, y, test_size=0.25, random_state=42)

# Huấn luyện mô hình LogisticRegression
model = LogisticRegression(multi_class='multinomial', max_iter=1000, random_state=42)
model.fit(x_train, y_train)

# Dự đoán và đánh giá mô hình

accuracy = model.score(x_test, y_test)
print(f'Accuracy: {accuracy:.2f}')
y_pred = model.predict(x_test)
print(f'\nClassification Report:\n')
print(classification_report(y_test, y_pred))
print('\nConfusion Matrix:')
print(confusion_matrix(y_test, y_pred))
```

- Phân nhóm giá trị Price thành 3 nhóm: **Low, Medium, High** bằng **qcut**.
- Chuẩn hóa các cột số **numeric_col** bằng **StandardScaler**.
- Chia dữ liệu thành tập huấn luyện và kiểm tra. - Huấn luyện mô hình **LogisticRegression** đa lớp (**multi_class='multinomial'**) để dự đoán nhóm giá. - Đánh giá độ chính xác (**accuracy**), báo cáo phân loại (**classification_report**), và ma trận nhầm lẫn (**confusion_matrix**).

- **Bước 3:** Thu lấy dataframe tầm quan trọng thuộc tính:

```python
# Tầm quan trọng đặc trưng
features_importance = pd.DataFrame({
    'Feature': numeric_col,
    'Importance': np.abs(model.coef_[0])
})
features_importance = features_importance.sort_values('Importance', ascending=False, ignore_index=True)
print('Features importance dataframe:')
print(features_importance)

# Vẽ biểu đồ
plt.figure(figsize=(10, 6))
plt.barh(features_importance['Feature'], features_importance['Importance'])
plt.xlabel('Features importance of Player transfer value classification')
plt.yticks(fontsize=5)
plt.gca().invert_yaxis()
plt.tight_layout()
plt.savefig('Problem 4/Feature_importance.png')

joblib.dump(model, 'Problem 4/logistic_transfer_model.pkl')
joblib.dump(scaler, 'Problem 4/scaler.pkl')
```

- Tạo dataframe để lưu thuộc tính và tầm quan trọng của nó.

- Sắp xếp các giá trị theo thứ tự giảm dần.
- In ra và vẽ dưới dạng biểu đồ. - Cuối cùng lưu mô hình vào file.