Posts and Telecommunications Institute of Technology

Python-Assignment

Instructors	:	**Kim Ngọc Bách**
Class	:	**D23CQCE04-B**
Student code	:	**B23DCVT188**
Student	:	**Phạm Quốc Hùng**

Index

Problem 1

Topic

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:

  o Each column corresponds to a statistic.

  o Players are sorted alphabetically by their first name.

  o Any statistic that is unavailable or inapplicable should be marked as "N/a".

• [*] The required statistics are:

  o Nation

  o Team

  o Position

  o Age

  o Playing Time: matches played, starts, minutes

  o Performance: goals, assists, yellow cards, red cards

  o Expected: expected goals (xG), expedted Assist Goals (xAG)

  o Progression: PrgC, PrgP, PrgR

  o Per 90 minutes: Gls, Ast, xG, xGA

  o Goalkeeping:

    ▪ Performance: goals against per 90mins (GA90), Save%, CS%

    ▪ Penalty Kicks: penalty kicks Save%

o Shooting:

▪ Standard: shoots on target percentage (SoT%), Shoot on Target per 90min (SoT/90), goals/shot (G/sh), average shoot distance (Dist)

o 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

o Goal and Shot Creation:

▪ SCA: SCA, SCA90

▪ GCA: GCA, GCA90

o Defensive Actions:

▪ Tackles: Tkl, TklW

▪ Challenges: Att, Lost

▪ Blocks: Blocks, Sh, Pass, Int

o 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%

o Reference: https://fbref.com/en/squads/822bd0ba/Liverpool-Stats

1.1 Problem solving method:

• Use requests to get the links of the football teams from the website https://fbref.com/en/ from which we can access the links of the football teams

• Use Beautifulsoup to get the HTML code from the website

• Use Devtools to get the tags of the HTML code (div, id, ...) from which we can get the measurements of each player's index

1.2 Libraries needed:

• requests + bs4: Collect and parse web data.

• time + random: Control the request time to avoid blocking.

• pandas: Store and process tabular data.

• os: library used to get data or export data to another folder

1.3 Specific process:

Step 1. Collect the list of EPL 2024-2025 football teams

• Objective: Get the URLs of all football teams participating in EPL 2024-2025.

• How to do it:

o Send a request to the FBref homepage and parse the HTML using BeautifulSoup.

o Find the div with a specific ID (all_results2024-202591) that contains the team rankings.

o Extract the team name and the stat page URL of each team from the <td data-stat="team"> tags.

Code:

```
def get_infor_of_each_team():

  url = 'https://fbref.com/en/'

  response = requests.get(url)

  soup = bs4.BeautifulSoup(response.content, 'html.parser')
```

```
    div = soup.find('div', {'id': "all_results2024-202591"})
    tds = div.find_all('td', {'data-stat': "team"})
    link_and_name_of_team = []
    for td in tds:
        a = td.find('a')
        if a and 'href' in a.attrs:
            link = "https://fbref.com" + a['href']
            team_name = td.get_text()
            link_and_name_of_team.append((link, team_name))
    return link_and_name_of_team
```

Here we will get the link with the name of each football team, which will be convenient for us to be able to combine the 'Team' index for each player in that football team. We get the football team link by combining the string "https://fbref.com" + the 'href' part of the a tag in the td tags with data-stat ="team" in the div tag with id = "all_results2024-202591". Similarly, we can also get the name of the football team by getting the td.get_text() of the td tags with data-stat ="team" in the div tag with id = "all_results2024-202591".

Step 2. Filter players with over 90 minutes of play

- Objective: Only keep players who have played over 90 minutes.
- How to do it:
    - Access the statistics page of each team.
    - Parse the basic statistics table (div_stats_standard_9).
    - Calculate the total minutes played from the minutes_90s column (number of 90 minutes).
    - Only keep players with minutes_90s > 1 (equivalent to >90 minutes).

Code:

```
table = soup.find('div', id='div_stats_standard_9')
  if not table:
    return []
  tbody = table.find('tbody')
  players = {}
  for tr in tbody.find_all('tr'):
    th = tr.find('th')
    name = th.get_text()
    td = tr.find('td', {'data-stat': 'minutes_90s'})
    minutes_90s = float('0' + td.get_text()) if td else 0
    if minutes_90s > 1:
      players[name] = [name]
```

- Use the find() and find_all() functions flexibly to find minutes_90s, from which we can extract the number of times the 90-minute participation threshold has been reached.

• To handle exceptions such as (players on 2 pages have different names, players have been transferred to another league but the results file still exists...), we will handle them by adding them manually

Example:

```
table_ids = ['div_stats_standard_9', 'div_stats_keeper_9',
'div_stats_shooting_9', ...]
data_stats = [
  [
    'nationality',
    'position',
    'age',
    'games',
        'games_starts',
         ...
    ],
...
]
for table_id, dstats in zip(table_ids, data_stats):
        table = soup.find('div', id=table_id)
                        ...
```

o Iterate through each table, parse the data and assign it to the players dictionary.

Here we have a special type of index that needs to be normalized, which is Nation. For example, when getting the Nation index in text form of an England player, it will be in the form eng ENG, so we will split the above string into 2 parts and take the last part, if when getting the text form of the Nation index of a player is empty, we assign that index with the value "N/a"

o Handle cases of missing data or empty data cells by assigning the value "N/a"

In Step 2 and Step 3 (Because these 2 steps belong to the same function to get the value of the index for the players) use the anti-scraping method

Implement by:

- Add random delay between requests (time.sleep(random.uniform(1, 6))).
- Handle response code 429 (too many requests) by retrying after waiting.

Step 4: Rename the columns, create a CSV file and sort the data.

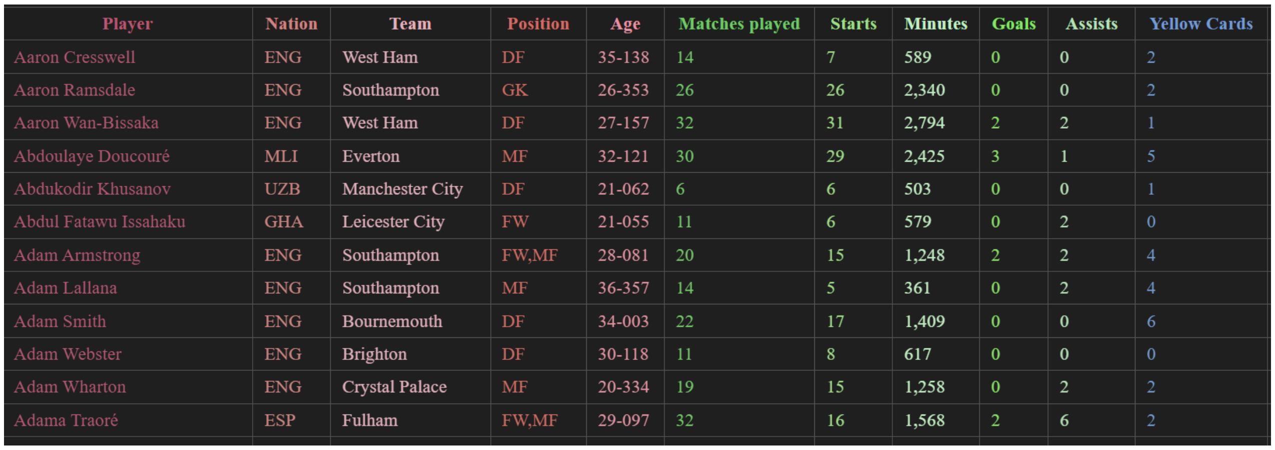
We rename the columns to see the format to be most suitable and beautiful

columns = ['Player', 'Nation', 'Team', ...]

We use pandas to create a table, then sort the data by the index 'Player' (ie sort the players by name) and then write to the file results.csv

1.4 Results

Player stats are recorded in the results.csv file:



Player	Nation	Team	Position	Age	Matches played	Starts	Minutes	Goals	Assists	Yellow Cards
Aaron Cresswell	ENG	West Ham	DF	35-138	14	7	589	0	0	2
Aaron Ramsdale	ENG	Southampton	GK	26-353	26	26	2,340	0	0	2
Aaron Wan-Bissaka	ENG	West Ham	DF	27-157	32	31	2,794	2	2	1
Abdoulaye Doucouré	MLI	Everton	MF	32-121	30	29	2,425	3	1	5
Abdukodir Khusanov	UZB	Manchester City	DF	21-062	6	6	503	0	0	1
Abdul Fatawu Issahaku	GHA	Leicester City	FW	21-055	11	6	579	0	2	0
Adam Armstrong	ENG	Southampton	FW,MF	28-081	20	15	1,248	2	2	4
Adam Lallana	ENG	Southampton	MF	36-357	14	5	361	0	2	4
Adam Smith	ENG	Bournemouth	DF	34-003	22	17	1,409	0	0	6
Adam Webster	ENG	Brighton	DF	30-118	11	8	617	0	0	0
Adam Wharton	ENG	Crystal Palace	MF	20-334	19	15	1,258	0	2	2
Adama Traoré	ESP	Fulham	FW,MF	29-097	32	16	1,568	2	6	2

See full results in the folder of lesson I

Problem 2:

Topic

• 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 Atttribute 1	Mean of Atttribute 1	Std of Atttribute 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

2.1 Problem solving method:

• Use two functions nlargest() and nsmallest() to identify the top 3 players with the highest and lowest scores for each statistic (nlargest() to find the highest and nsmallest() to find the lowest

• Use the functions .median(), .mean(), .std() of pandas to calculate the median, the mean and standard deviation for each statistic across all players and for each team. (should use this function of pandas because it can automatically ignore Nan values, this is very convenient for calculation)

• Use the library matplotlib.pyplot to plot a histogram showing the distribution of each statistic for all players in the league and each team.

2.2 Libraries needed:

• os: library used to get data or export data to another directory

• pandas: has functions .median(), .mean(), .std() convenient for calculation, and to draw data into a table format to the file results2.csv

• numpy : has 2 functions nlargest() and nsmallest() to identify the top 3 players with the highest and lowest scores for each statistic

• matplotlib : to draw the Histogram chart

2.3 Specific process:

Part 1: Identify the top 3 players with the highest and lowest scores for each statistic

• The indicators to get the top 3 and bottom 3 are other indicators ['Player', 'Nation', 'Team', 'Position', 'Age']

• Normalize cells with index 'N/a' to 'NaN' to easily remove cells with empty indexes

• We will re-normalize the special indicators Minutes (remove commas), Age (normalize to date format), then we convert all other indicators ['Player', 'Nation', 'Team', 'Position', 'Age'] to numeric format for easy comparison.

• For each index, we need to get the top 3 and bottom 3:

o Use the nlargest function to get the top 3 players with the highest index and nsmallest to get the top 3 players with the lowest index

  - o For the Age index, before writing it to the pandas table and file, we will standardize it back to its original form (Years-Days) to make it more beautiful
  - o We put the collected data into the pandas table and then adjust it (align, ...) so that when we monitor the results, it will be easier to see

- • Then we print the top_3.txt file

Part 2: Find the median and calculate the mean and standard deviation for each statistic across all players and for each team

- • Identify the indexes that need to find the median and calculate the mean and standard deviation as other indexes ['Player', 'Nation', 'Team', 'Position', 'Age']
- • Standardize cells with the index 'N/a' to 'NaN' to easily remove cells has an empty index
- • We will re-standardize the special indexes Minutes (remove commas), Age (standardize to year), then we convert all other indexes ['Player', 'Nation', 'Team', 'Position', 'Age'] to numeric form for easy comparison.
- • For each index, we need to get median, mean, std(All, each team):
  - o We use the functions .mead(), .median(), .std() (these functions will automatically ignore NaN values) to calculate median, mean, standard deviation for each index.
- • We put the collected data into a pandas table and export it to the file results2.csv

Part 3: Plot a histogram showing the distribution of each statistic for all players in the league and each team

- • Copy a separate copy of the results.csv file from lesson 1 to avoid direct editing of the original data, then normalize the index cells with 'N/a' data to NaN data, this helps us process the data later
- • Select 3 attack indexes and 3 defense indexes to represent

attack = ['Goals', 'Assists', 'Expected Goals']

defense = ['Tackles', 'Interceptions', 'Blocks']

- • Convert the indexes to numbers, if any cell is NaN, convert it to 0

- Use the functions in the matplotlib.pyplot library to plot the performance table of all players and from the team according to the 3 attack indexes and 3 defense indexes mentioned above on

- Save the charts as images to Histogram img file

Part 4: 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?

- Data processing: Convert Age from "year-day" format to decimal, normalize the Minutes column, replace error values (N/a → NaN).

- Find the strongest team: For each statistical index (except the classification column), find the team with the MAX value and print the result.

- Scoring system:

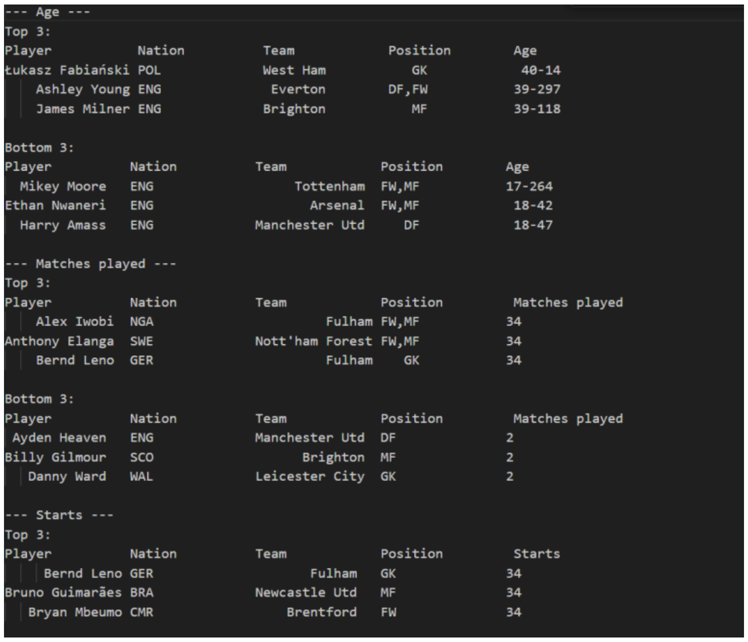
  o For "Mean" indexes, the best team in each column (MAX/MIN depending on the type) gets +1 point.

  o Indexes such as Goals Against, Red Cards… are rated BETTER if the value is LOW.

- Result: The team with the highest total points wins.

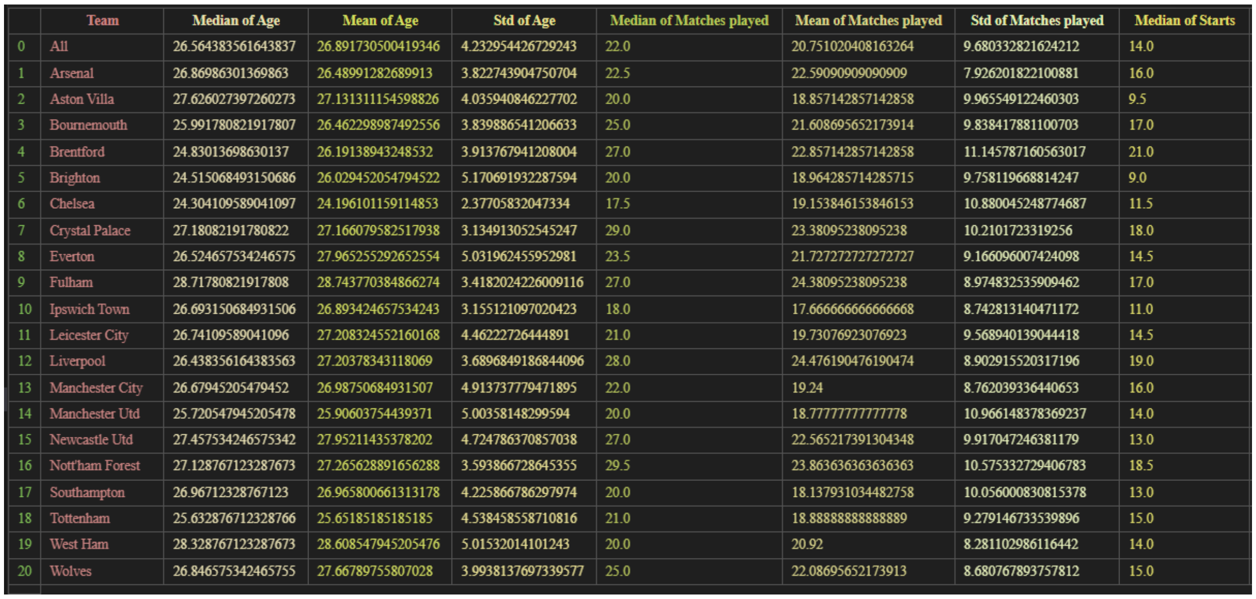
2.4 Result

Part 1:



```
--- Age ---
Top 3:
Player          Nation          Team            Position        Age
Łukasz Fabiański POL            West Ham           GK            40-14
    Ashley Young ENG             Everton        DF,FW           39-297
    James Milner ENG            Brighton           MF           39-118

Bottom 3:
Player          Nation          Team            Position        Age
  Mikey Moore   ENG                  Tottenham  FW,MF           17-264
Ethan Nwaneri   ENG                    Arsenal  FW,MF            18-42
  Harry Amass   ENG             Manchester Utd     DF            18-47

--- Matches played ---
Top 3:
Player          Nation          Team            Position         Matches played
    Alex Iwobi  NGA                       Fulham FW,MF          34
Anthony Elanga  SWE             Nott'ham Forest FW,MF          34
    Bernd Leno  GER                       Fulham    GK          34

Bottom 3:
Player          Nation          Team            Position         Matches played
 Ayden Heaven   ENG             Manchester Utd  DF              2
Billy Gilmour   SCO                   Brighton  MF              2
   Danny Ward   WAL             Leicester City  GK              2

--- Starts ---
Top 3:
Player          Nation          Team            Position         Starts
     Bernd Leno GER                     Fulham   GK             34
Bruno Guimarães BRA             Newcastle Utd   MF              34
   Bryan Mbeumo CMR                  Brentford   FW             34
```

Part 2:

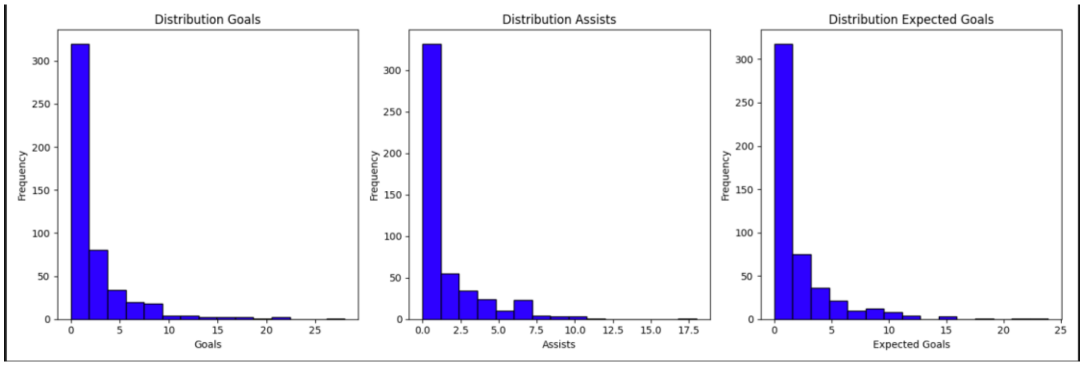


	Team	Median of Age	Mean of Age	Std of Age	Median of Matches played	Mean of Matches played	Std of Matches played	Median of Starts
0	All	26.564383561643837	26.891730500419346	4.232954426729243	22.0	20.751020408163264	9.680332821624212	14.0
1	Arsenal	26.86986301369863	26.48991282689913	3.822743904750704	22.5	22.59090909090909	7.926201822100881	16.0
2	Aston Villa	27.626027397260273	27.131311154598826	4.035940846227702	20.0	18.857142857142858	9.965549122460303	9.5
3	Bournemouth	25.991780821917807	26.462298987492556	3.839886541206633	25.0	21.608695652173914	9.838417881100703	17.0
4	Brentford	24.83013698630137	26.19138943248532	3.913767941208004	27.0	22.857142857142858	11.145787160563017	21.0
5	Brighton	24.515068493150686	26.029452054794522	5.170691932287594	20.0	18.964285714285715	9.758119668814247	9.0
6	Chelsea	24.304109589041097	24.196101159114853	2.37705832047334	17.5	19.153846153846153	10.880045248774687	11.5
7	Crystal Palace	27.18082191780822	27.166079582517938	3.134913052545247	29.0	23.38095238095238	10.2101723319256	18.0
8	Everton	26.524657534246575	27.965255292652554	5.031962455952981	23.5	21.727272727272727	9.166096007424098	14.5
9	Fulham	28.71780821917808	28.743770384866274	3.4182024226009116	27.0	24.38095238095238	8.974832535909462	17.0
10	Ipswich Town	26.693150684931506	26.893424657534243	3.155121097020423	18.0	17.666666666666668	8.742813140471172	11.0
11	Leicester City	26.74109589041096	27.208324552160168	4.46222726444891	21.0	19.73076923076923	9.568940139044418	14.5
12	Liverpool	26.438356164383563	27.20378343118069	3.6896849186844096	28.0	24.476190476190474	8.902915520317196	19.0
13	Manchester City	26.67945205479452	26.98750684931507	4.913737779471895	22.0	19.24	8.762039336440653	16.0
14	Manchester Utd	25.720547945205478	25.90603754439371	5.00358148299594	20.0	18.77777777777778	10.966148378369237	14.0
15	Newcastle Utd	27.457534246575342	27.95211435378202	4.724786370857038	27.0	22.565217391304348	9.917047246381179	13.0
16	Nott'ham Forest	27.128767123287673	27.265628891656288	3.593866728645355	29.5	23.863636363636363	10.575332729406783	18.5
17	Southampton	26.96712328767123	26.965800661313178	4.225866786297974	20.0	18.137931034482758	10.056000830815378	13.0
18	Tottenham	25.632876712328766	25.65185185185185	4.538458558710816	21.0	18.88888888888889	9.279146733539896	15.0
19	West Ham	28.328767123287673	28.608547945205476	5.01532014101243	20.0	20.92	8.281102986116442	14.0
20	Wolves	26.846575342465755	27.66789755807028	3.9938137697339577	25.0	22.08695652173913	8.680767893757812	15.0

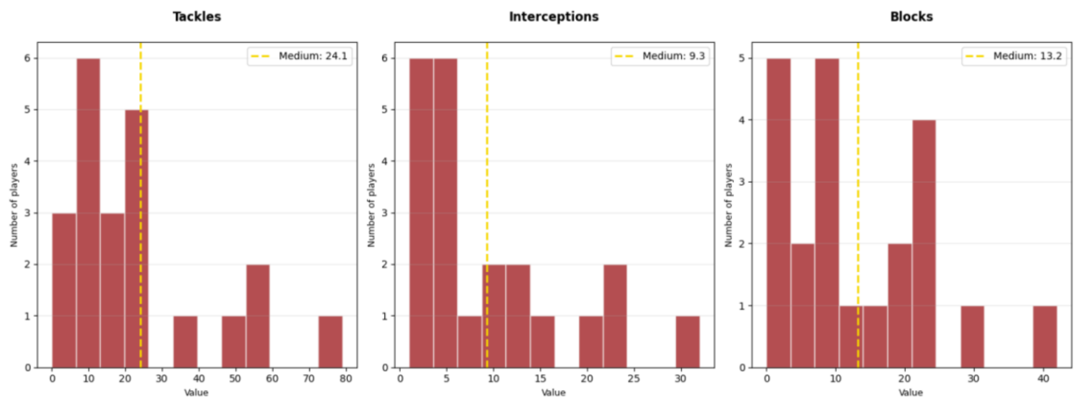
See full results in the folder of lesson I

Part 3:

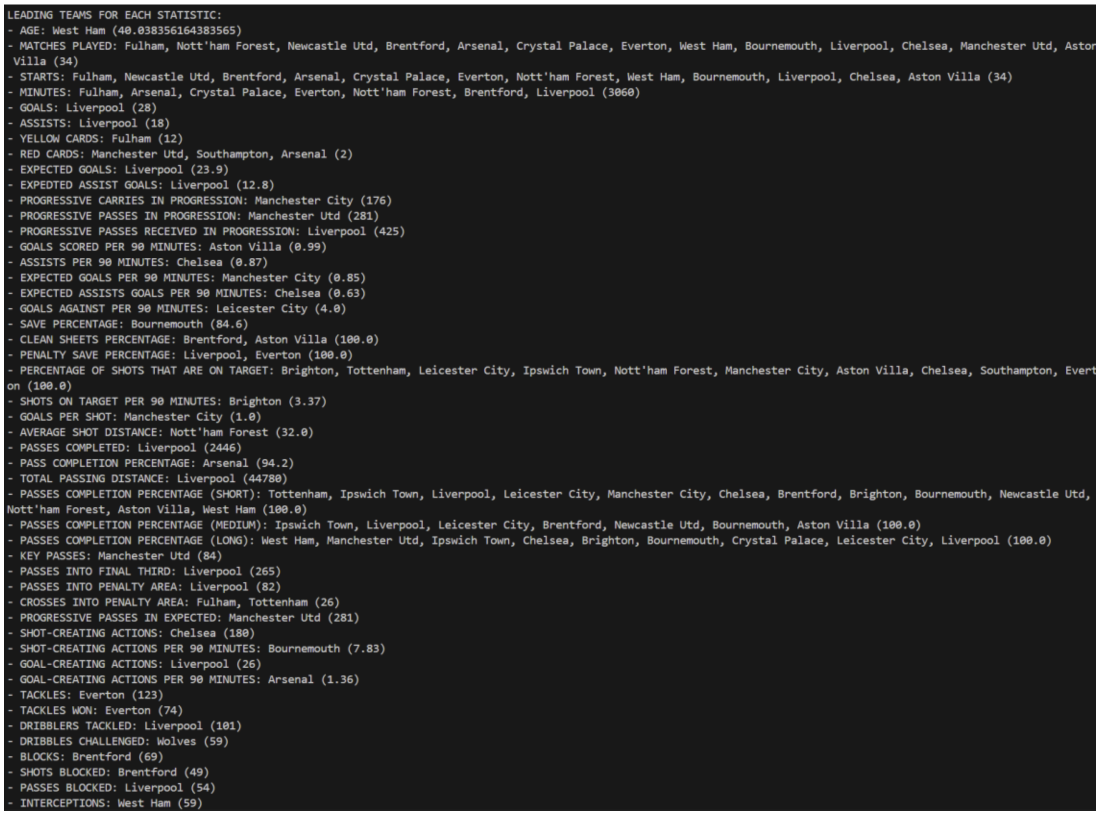
All:



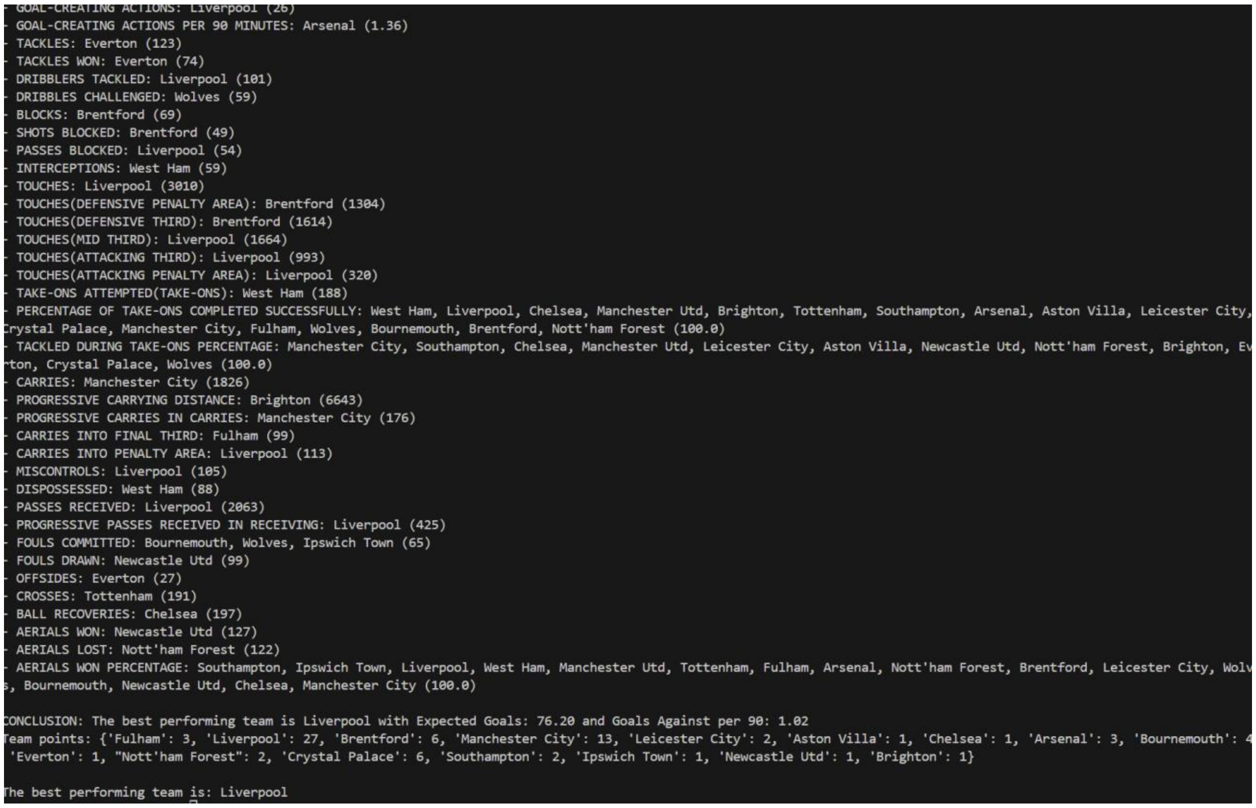
Each Team:



Part 4:



```
LEADING TEAMS FOR EACH STATISTIC:
- AGE: West Ham (40.038356164383565)
- MATCHES PLAYED: Fulham, Nott'ham Forest, Newcastle Utd, Brentford, Arsenal, Crystal Palace, Everton, West Ham, Bournemouth, Liverpool, Chelsea, Manchester Utd, Astor
 Villa (34)
- STARTS: Fulham, Newcastle Utd, Brentford, Arsenal, Crystal Palace, Everton, Nott'ham Forest, West Ham, Bournemouth, Liverpool, Chelsea, Aston Villa (34)
- MINUTES: Fulham, Arsenal, Crystal Palace, Everton, Nott'ham Forest, Brentford, Liverpool (3060)
- GOALS: Liverpool (28)
- ASSISTS: Liverpool (18)
- YELLOW CARDS: Fulham (12)
- RED CARDS: Manchester Utd, Southampton, Arsenal (2)
- EXPECTED GOALS: Liverpool (23.9)
- EXPEDTED ASSIST GOALS: Liverpool (12.8)
- PROGRESSIVE CARRIES IN PROGRESSION: Manchester City (176)
- PROGRESSIVE PASSES IN PROGRESSION: Manchester Utd (281)
- PROGRESSIVE PASSES RECEIVED IN PROGRESSION: Liverpool (425)
- GOALS SCORED PER 90 MINUTES: Aston Villa (0.99)
- ASSISTS PER 90 MINUTES: Chelsea (0.87)
- EXPECTED GOALS PER 90 MINUTES: Manchester City (0.85)
- EXPECTED ASSISTS GOALS PER 90 MINUTES: Chelsea (0.63)
- GOALS AGAINST PER 90 MINUTES: Leicester City (4.0)
- SAVE PERCENTAGE: Bournemouth (84.6)
- CLEAN SHEETS PERCENTAGE: Brentford, Aston Villa (100.0)
- PENALTY SAVE PERCENTAGE: Liverpool, Everton (100.0)
- PERCENTAGE OF SHOTS THAT ARE ON TARGET: Brighton, Tottenham, Leicester City, Ipswich Town, Nott'ham Forest, Manchester City, Aston Villa, Chelsea, Southampton, Evert
on (100.0)
- SHOTS ON TARGET PER 90 MINUTES: Brighton (3.37)
- GOALS PER SHOT: Manchester City (1.0)
- AVERAGE SHOT DISTANCE: Nott'ham Forest (32.0)
- PASSES COMPLETED: Liverpool (2446)
- PASS COMPLETION PERCENTAGE: Arsenal (94.2)
- TOTAL PASSING DISTANCE: Liverpool (44780)
- PASSES COMPLETION PERCENTAGE (SHORT): Tottenham, Ipswich Town, Liverpool, Leicester City, Manchester City, Chelsea, Brentford, Brighton, Bournemouth, Newcastle Utd,
Nott'ham Forest, Aston Villa, West Ham (100.0)
- PASSES COMPLETION PERCENTAGE (MEDIUM): Ipswich Town, Liverpool, Leicester City, Brentford, Newcastle Utd, Bournemouth, Aston Villa (100.0)
- PASSES COMPLETION PERCENTAGE (LONG): West Ham, Manchester Utd, Ipswich Town, Chelsea, Brighton, Bournemouth, Crystal Palace, Leicester City, Liverpool (100.0)
- KEY PASSES: Manchester Utd (84)
- PASSES INTO FINAL THIRD: Liverpool (265)
- PASSES INTO PENALTY AREA: Liverpool (82)
- CROSSES INTO PENALTY AREA: Fulham, Tottenham (26)
- PROGRESSIVE PASSES IN EXPECTED: Manchester Utd (281)
- SHOT-CREATING ACTIONS: Chelsea (180)
- SHOT-CREATING ACTIONS PER 90 MINUTES: Bournemouth (7.83)
- GOAL-CREATING ACTIONS: Liverpool (26)
- GOAL-CREATING ACTIONS PER 90 MINUTES: Arsenal (1.36)
- TACKLES: Everton (123)
- TACKLES WON: Everton (74)
- DRIBBLERS TACKLED: Liverpool (101)
- DRIBBLES CHALLENGED: Wolves (59)
- BLOCKS: Brentford (69)
- SHOTS BLOCKED: Brentford (49)
- PASSES BLOCKED: Liverpool (54)
- INTERCEPTIONS: West Ham (59)
```



```
- GOAL-CREATING ACTIONS: Liverpool (26)
- GOAL-CREATING ACTIONS PER 90 MINUTES: Arsenal (1.36)
- TACKLES: Everton (123)
- TACKLES WON: Everton (74)
- DRIBBLERS TACKLED: Liverpool (101)
- DRIBBLES CHALLENGED: Wolves (59)
- BLOCKS: Brentford (69)
- SHOTS BLOCKED: Brentford (49)
- PASSES BLOCKED: Liverpool (54)
- INTERCEPTIONS: West Ham (59)
- TOUCHES: Liverpool (3010)
- TOUCHES(DEFENSIVE PENALTY AREA): Brentford (1304)
- TOUCHES(DEFENSIVE THIRD): Brentford (1614)
- TOUCHES(MID THIRD): Liverpool (1664)
- TOUCHES(ATTACKING THIRD): Liverpool (993)
- TOUCHES(ATTACKING PENALTY AREA): Liverpool (320)
- TAKE-ONS ATTEMPTED(TAKE-ONS): West Ham (188)
- PERCENTAGE OF TAKE-ONS COMPLETED SUCCESSFULLY: West Ham, Liverpool, Chelsea, Manchester Utd, Brighton, Tottenham, Southampton, Arsenal, Aston Villa, Leicester City,
rystal Palace, Manchester City, Fulham, Wolves, Bournemouth, Brentford, Nott'ham Forest (100.0)
- TACKLED DURING TAKE-ONS PERCENTAGE: Manchester City, Southampton, Chelsea, Manchester Utd, Leicester City, Aston Villa, Newcastle Utd, Nott'ham Forest, Brighton, Ev
ton, Crystal Palace, Wolves (100.0)
- CARRIES: Manchester City (1826)
- PROGRESSIVE CARRYING DISTANCE: Brighton (6643)
- PROGRESSIVE CARRIES IN CARRIES: Manchester City (176)
- CARRIES INTO FINAL THIRD: Fulham (99)
- CARRIES INTO PENALTY AREA: Liverpool (113)
- MISCONTROLS: Liverpool (105)
- DISPOSSESSED: West Ham (88)
- PASSES RECEIVED: Liverpool (2063)
- PROGRESSIVE PASSES RECEIVED IN RECEIVING: Liverpool (425)
- FOULS COMMITTED: Bournemouth, Wolves, Ipswich Town (65)
- FOULS DRAWN: Newcastle Utd (99)
- OFFSIDES: Everton (27)
- CROSSES: Tottenham (191)
- BALL RECOVERIES: Chelsea (197)
- AERIALS WON: Newcastle Utd (127)
- AERIALS LOST: Nott'ham Forest (122)
- AERIALS WON PERCENTAGE: Southampton, Ipswich Town, Liverpool, West Ham, Manchester Utd, Tottenham, Fulham, Arsenal, Nott'ham Forest, Brentford, Leicester City, Wolv
s, Bournemouth, Newcastle Utd, Chelsea, Manchester City (100.0)

ONCLUSION: The best performing team is Liverpool with Expected Goals: 76.20 and Goals Against per 90: 1.02
eam points: {'Fulham': 3, 'Liverpool': 27, 'Brentford': 6, 'Manchester City': 13, 'Leicester City': 2, 'Aston Villa': 1, 'Chelsea': 1, 'Arsenal': 3, 'Bournemouth': 4
 'Everton': 1, "Nott'ham Forest": 2, 'Crystal Palace': 6, 'Southampton': 2, 'Ipswich Town': 1, 'Newcastle Utd': 1, 'Brighton': 1}

he best performing team is: Liverpool
```

See full results in the folder of lesson II

Problem 3

Topic

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.

3.1 Problem solving method

1. Combining Elbow method with Silhouette index to determine the optimal number of clusters

2. Clustering by K-means

3. Visualization by PCA (2D Projection)

3.2 Libraries needed:

1. Data processing:

    o pandas + numpy: Preprocess raw data → suitable format for ML.

    o SimpleImputer + StandardScaler: Prepare "clean" data for the algorithm.

2. Clustering Core:

    o scikit-learn: to use K-means algorithm.

    o silhouette_score: evaluate cluster quality.

3. Visualization:

    o matplotlib + seaborn: Turn complex data into visual images.

4. Path:

    o os: library used to get data or export data to another directory

3.3 Specific process:

Step 1: Preprocess data to find the optimal number of clusters:

Select 13 important performance indicators

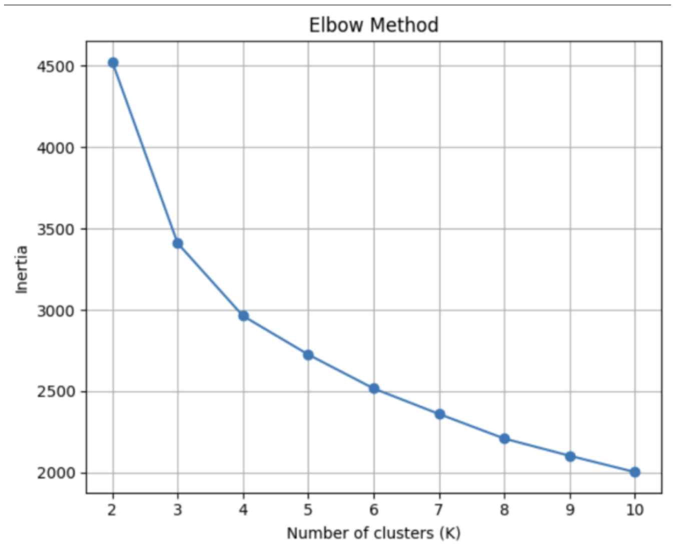
Handle missing values by replacing with the mean value (K-means algorithm cannot handle NaN data, Replacing with the mean value will help keep the data set intact)

Standardize data with StandardScaler

Step 2: Determine the optimal number of clusters:

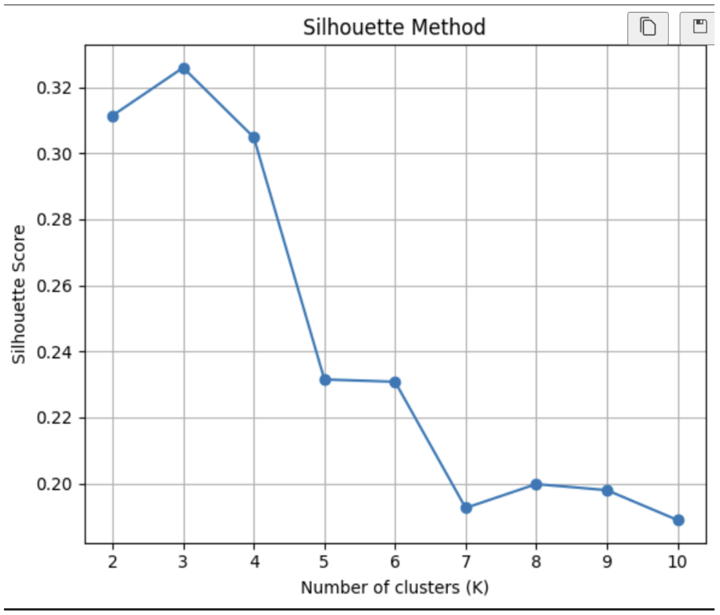
• Use the Elbow method to find the "elbow" point where the variance in the cluster (inertia) decreases slowly.

How to find: calculate inertia for k values from 2 to max_k, then find the point where the second derivative of the inertia changes the most (the inflection point), showing the slow decrease of the inertia.



• Use Silhouette analysis method to find k with the highest average Silhouette score, reflecting the clustering quality.

How to find: calculate Silhouette score for each k, Choose k with the highest value:



• Combining both methods to choose the optimal number of clusters k, here we find k = 3 in the range [3,7]

Combination logic:

Take the average of the k values from Elbow and Silhouette. Limit the result to the range [3, 7] to avoid values that are too low, high or inappropriate.

So the optimal number of clusters is 3 because:

Combining the Elbow Method and Silhouette Analysis to determine the optimal number of clusters k is an effective approach to balance between mathematics and cluster quality.

The Elbow Method relies on inertia (the sum of squared distances to the cluster center) to find the "elbow" point - where the rate of inertia decrease slows down significantly, often indicating that kk is suitable to describe the data structure. However, this method is sometimes unclear or subjective. Meanwhile, Silhouette Analysis measures cluster quality by calculating the similarity within each cluster and the separation between clusters, with values ranging from -1 to 1 (higher is better). Silhouette helps assess the separation of clusters but may suggest too large k if the data is noisy.

By combining the two, your code averages the k values suggested by Elbow and Silhouette, and limits k to [3, 7] to ensure practicality. This limit is suitable for many problems, avoiding kk that is too small (lack of detail) or too large (difficult to interpret). For example, if Elbow recommends k=4 and Silhouette recommends k=5, the final value will be k=5, which balances mathematical optimization and cluster quality.

This method is especially useful when the data is well-structured but not too complex, and helps automate the selection of k instead of relying on subjective intuition.

Step 3: Clustering using K-means.

Code:

```
kmeans = KMeans(n_clusters=k, random_state=42, n_init=10)

kmeans.fit(X_scaled)

labels = kmeans.labels_
```

• n_clusters=k: Number of clusters determined from the previous step, here we have k = 3 (found)

• random_state=42: Ensure the results are reproducible.

• n_init=10: Run the algorithm 10 times with different initial points, choose the best result.

How it works

1. Initialize k random cluster centers (here k = 3)

2. Insert each data point into the nearest cluster (using Euclidean distance).

3. Update the cluster center position with the average value of the points in the cluster.

4. Repeat until the cluster center is stable.

Step 4: Cluster Analysis & Interpretation

• Position statistics: Count the position ratio (FW/MF/DF) in each cluster.

• Output: Save CSV file containing player information + cluster label.

• Quality assessment:

Code:

```
def interpret_silhouette(score):
    if score > 0.7: return "Very well separated clusters"
    elif score > 0.5: return "Well separated clusters"
    elif score > 0.3: return "Reasonable structure found"
    else: return "Clusters overlap considerably"
```

o Here we find (Silhouette Score = 0.326) > 0.3 → Found suitable structure.

o Position concentration → Check the logic of the cluster.

Step 5: Preprocess data to transfer to PCA:

• Select 13 important performance indicators

• Handle missing values by replacing with the average value (K-means algorithm cannot handle NaN data, Replacing with the average value will help keep the data set intact)

• Standardize data with StandardScaler

Step 6: Visualize with PCA (2D Projection)

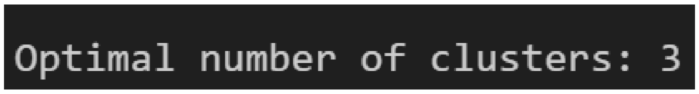
PCA (Principal Component Analysis) is a method of reducing data dimensionality, converting data from a multidimensional space (13 features in this problem) to a 2D or 3D space for easy visualization.

• Reduce data dimensionality: convert 13 features → 2 principal components (PC1, PC2).

• 2D chart: Color distinguishes clusters. Annotate the names of typical players near the cluster center. Explain the variance (eg: PC1 = 45%, PC2 = 25%)

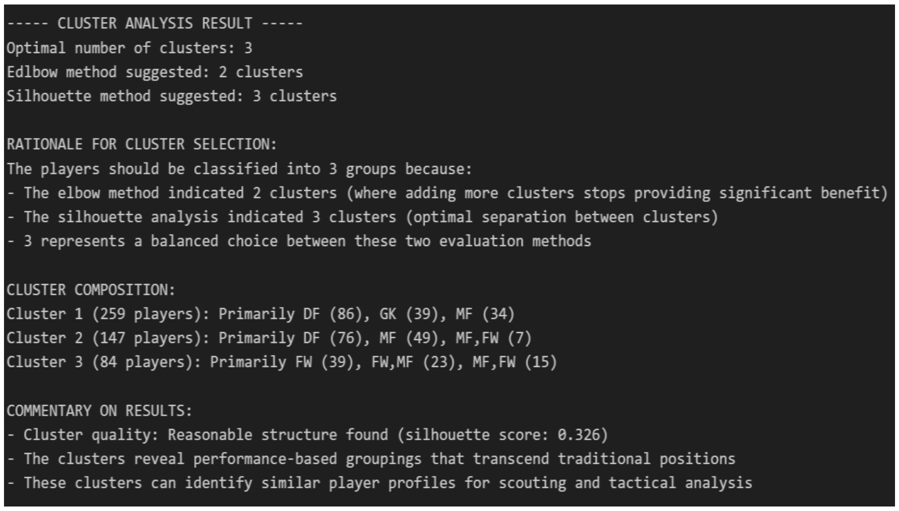
3.4 Results:

- ✓ Find the optimal number of clusters k = 3:



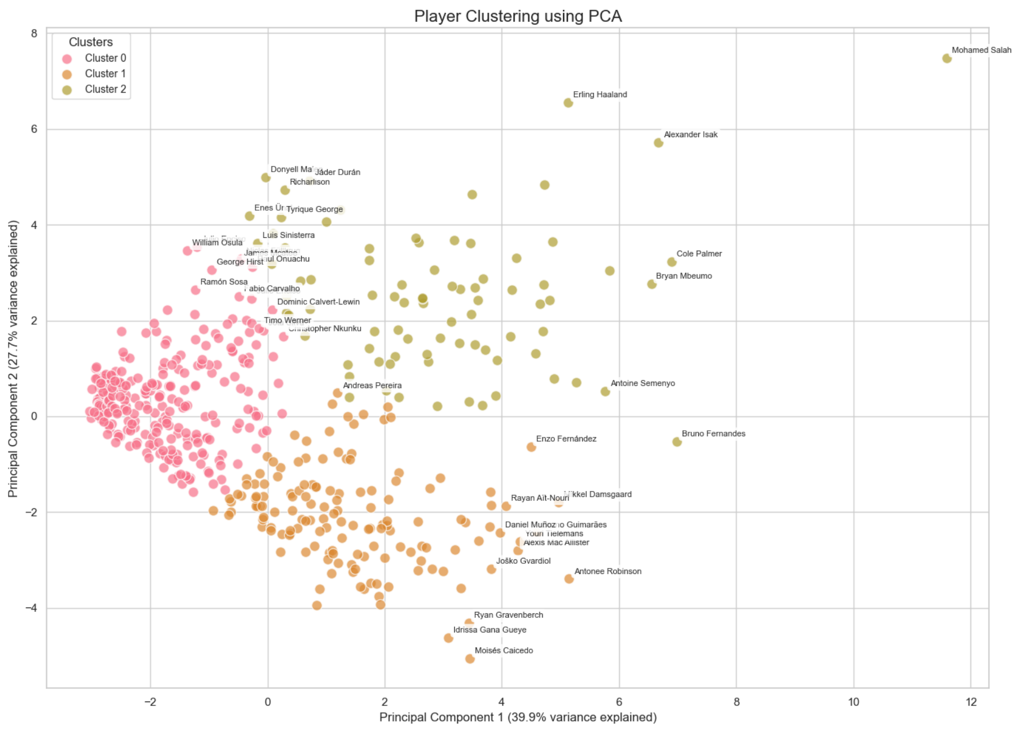
```
Optimal number of clusters: 3
```

- ✓ Classify players into clusters:



```
----- CLUSTER ANALYSIS RESULT -----
Optimal number of clusters: 3
Edlbow method suggested: 2 clusters
Silhouette method suggested: 3 clusters

RATIONALE FOR CLUSTER SELECTION:
The players should be classified into 3 groups because:
- The elbow method indicated 2 clusters (where adding more clusters stops providing significant benefit)
- The silhouette analysis indicated 3 clusters (optimal separation between clusters)
- 3 represents a balanced choice between these two evaluation methods

CLUSTER COMPOSITION:
Cluster 1 (259 players): Primarily DF (86), GK (39), MF (34)
Cluster 2 (147 players): Primarily DF (76), MF (49), MF,FW (7)
Cluster 3 (84 players): Primarily FW (39), FW,MF (23), MF,FW (15)

COMMENTARY ON RESULTS:
- Cluster quality: Reasonable structure found (silhouette score: 0.326)
- The clusters reveal performance-based groupings that transcend traditional positions
- These clusters can identify similar player profiles for scouting and tactical analysis
```

- ✓ Visualization using PCA (2D Projection):



Problem 4.

TOPIC

• 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?

4.1 Problem solving method:

• From the results.csv file, we can get information about players with more than 900 minutes of playing time, then we get the data of the players (including name and transfer value) for the Premier League from the page https://www.footballtransfers.com, connect the names of the 2 players (the same person) from the 2 data files just obtained (in some cases, manual processing is required) from there we will assign transfer value to the players. (This method should be used because if crawling data directly from the page https://www.footballtransfers.com will be very time-consuming)

4.2 Libraries needed:

- Selenium + BeautifulSoup: Automate the browser (Chrome hidden) and parse HTML to crawl player data.
- pandas: Read/write CSV, filter data (playing minutes ≥ 900), update transfer value, clean data.
- concurrent.futures: Multi-threaded processing to speed up crawling multiple players at the same time.
- fuzzywuzzy: Match player names (similarity >90%) to avoid duplicates.
- logging/warnings: Turn off redundant logs and warnings.
- os/csv: Manage CSV files (path, read/write).
- time: Create a delay waiting for page loading.
- typing: Declare function data types.
- Numpy (np): Arithmetic calculations (square roots, array processing).
- LightGBM (lgb): Build transfer fee prediction models.
- Sklearn:
    - train_test_split: Split train/test data.

  - StandardScaler: Standardize data.
  - mean_squared_error, r2_score: Evaluate the model.
- Matplotlib/Seaborn: Plot histograms (feature importance).
- Re: Check data format (number series).
- Pickle: Save the model for reuse.
- os: library used to get data or export data to another directory

4.3 Specific process:

Part 1:

• Get information about players who played more than 900 minutes from the results.csv file of lesson 1 including the indexes ['Player', 'Nation', 'Team', 'Position']

Get the index 'Player' to get the names of the players suitable for connecting names later, the remaining indexes are used to more clearly identify the identity of the player (in case the player provides his/her full name)

• Scrape data (including names and transfer prices) of players playing for the 2024-2025 Premier League from the website https://www.footballtransfers.com, here we will use selenium because this is a dynamic website, use BeautifulSoup to take HTML of web

• Connect 2 names of the same player in 2 scraped data files, then assign the corresponding transfer value to that player

The connection method is to use the fuzz function in the fuzzywuzzy library:

Code
```
def is_name_match(crawled_name, existing_names):
  if not existing_names:
    return True
  for existing_name in existing_names:
    similarity = fuzz.ratio(crawled_name.lower(), existing_name.lower())
    if similarity >= FUZZY_THRESHOLD:
      return True
  return False
```

•To handle exceptions such as (players on 2 pages have different names, players have been transferred to another league but the results file still exists...), we will handle them by adding them manually

Part 2:

cking Penalty Area, Carries, Pass Completion %, Goals/90, and Take-Ons metrics

This approach reduces dimensionality while maintaining predictive power, as demonstrated by comparable performance between the full and reduced models.

## Model Selection

A LightGBM regression model was selected because:

1. It handles high-dimensional data efficiently
2. Provides built-in feature importance metrics
3. Performs well with numeric features
4. Captures non-linear relationships in the data

Model hyperparameters:

- Objective: Regression
- Estimators: 100
- Learning rate: 0.05
- Max depth: 7
- Leaves: 31

## Evaluation Results

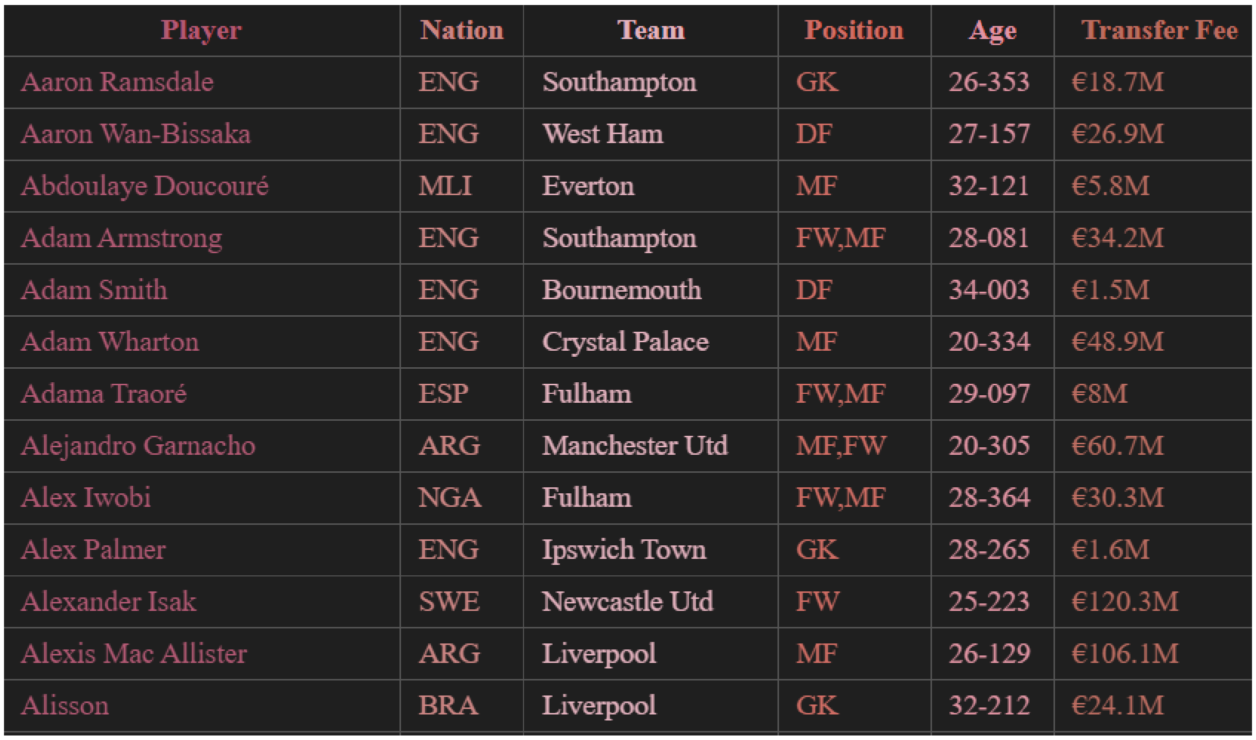
The model demonstrates strong predictive performance:

- R^2 score of approximately 0.73-0.75 (explaining ~75% of variance in transfer fees)
- Comparable performance between full feature model and top-10 feature model
- Effective for valuing players across different ages and positions

The methodology successfully identifies market value patterns based on performance metrics, allowing for objective player valuations.

4.4 Results

Part 1:



Player	Nation	Team	Position	Age	Transfer Fee
Aaron Ramsdale	ENG	Southampton	GK	26-353	€18.7M
Aaron Wan-Bissaka	ENG	West Ham	DF	27-157	€26.9M
Abdoulaye Doucouré	MLI	Everton	MF	32-121	€5.8M
Adam Armstrong	ENG	Southampton	FW,MF	28-081	€34.2M
Adam Smith	ENG	Bournemouth	DF	34-003	€1.5M
Adam Wharton	ENG	Crystal Palace	MF	20-334	€48.9M
Adama Traoré	ESP	Fulham	FW,MF	29-097	€8M
Alejandro Garnacho	ARG	Manchester Utd	MF,FW	20-305	€60.7M
Alex Iwobi	NGA	Fulham	FW,MF	28-364	€30.3M
Alex Palmer	ENG	Ipswich Town	GK	28-265	€1.6M
Alexander Isak	SWE	Newcastle Utd	FW	25-223	€120.3M
Alexis Mac Allister	ARG	Liverpool	MF	26-129	€106.1M
Alisson	BRA	Liverpool	GK	32-212	€24.1M

Part 2: in the folder of lesson IV

See full results in the folder of lesson IV