POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY 1

PYTHON PROGRAMMING

ASSIGNMENT 1

FOOTBALL DATA ANALYSIS

Instructor: Kim Ngoc Bach
Student: Pham Lan Anh
Student ID: B23DCCE010
Class ID: D23CQCE04-B

Hanoi

TABLE OF CONTENTS

1. Introduction

In recent years, data analysis has played an increasingly important role in professional sports, especially football. Statistical data is now widely used by clubs, analysts, and fans to assess player performance, optimize team strategies, and inform financial decisions such as player transfers. This project aims to utilize the Python programming language to collect, process, and analyze football player statistics from the 2024–2025 English Premier League season.

The primary goal of this project is to develop a system that collects a wide range of player performance metrics from public data sources. These indicators include playing time, goals, assists, expected goals (xG), defensive actions, passing statistics, and more. Only players with more than 90 minutes of playtime are included in the dataset, which is sourced from *fbref.com*, a well-known football statistics website. The collected data is stored in a file named `results.csv`.

The second objective is to perform statistical analysis on the data. This includes identifying top-performing players, computing summary statistics such as mean, median, and standard deviation, and visualizing data distributions using histograms. Additionally, the project aims to determine which group of players performs best based on aggregated metrics.

Furthermore, clustering techniques are applied using the K-Means algorithm, in combination with Principal Component Analysis (PCA), to reduce dimensionality and group players based on similar performance characteristics. The clustering results are visualized on a 2D plot for easy interpretation and analysis.

Finally, the project collects player market values from *footballtransfers.com* and proposes a machine learning approach (using Random Forest Regression) to estimate player market value based on in-game statistics. This involves preprocessing, feature selection, model training, and evaluation using metrics such as Mean Squared Error (MSE).

Through this project, the author not only enhances practical skills in Python-based data science but also gains a deeper understanding of how statistical methods and machine learning algorithms can be applied in real-world sports analytics, laying the foundation for broader future applications.

2. Collecting statistical data of premier league football players

Scrapes multiple data tables for each team and player from FBRef using their HTML table IDs:

- `stats_standard_9`: General player statistics
- `stats_keeper_9`: Goalkeeper statistics
- `stats_shooting_9`: Shooting statistics
- `stats_passing_9`: Passing statistics
- `stats_gca_9`: Goal Creation Actions (GCA) statistics
- `stats_defense_9`: Defensive statistics
- `stats_possession_9`: Possession statistics
- `stats_misc_9`: Miscellaneous statistics

a) Web Scraping
The web scraping is performed using the Selenium library to automate a browser and extract data from HTML tables. The implementation corresponds to the following steps:

- *Send an HTTP request to the target URL using Selenium*: the script initializes a headless Chrome browser via the `init_driver()` function and navigates to the target URL with `driver.get(url)`.
- *Parse the returned HTML content using Selenium*: the script directly accesses DOM elements via Selenium's `find_element()` and `find_elements()` methods.
- *Locate the relevant HTML table using a CSS Selector*: the script uses `driver.find_element(By.CSS_SELECTOR, f"table#table_id")` to find the desired table, then iterates over its `<tbody>` rows to extract player data.

b) Extracting Table Headers
The table headers are extracted from the thead section of the table. Each header corresponds to a specific statistical category for players, such as player, nationality, team, etc. The headers are filtered to only include the relevant statistics defined in the target list.

c) Extracting Data Rows
The rows of data are extracted from the tbody section. Each row contains the statistics for a player. The values are mapped to their corresponding headers. Special handling is applied to the minutes field, converting it into an integer. Missing values are handled by inserting “N/a”.

d) Data Merging
After scraping the data from the different stat groups, the individual DataFrames are merged into a single DataFrame.

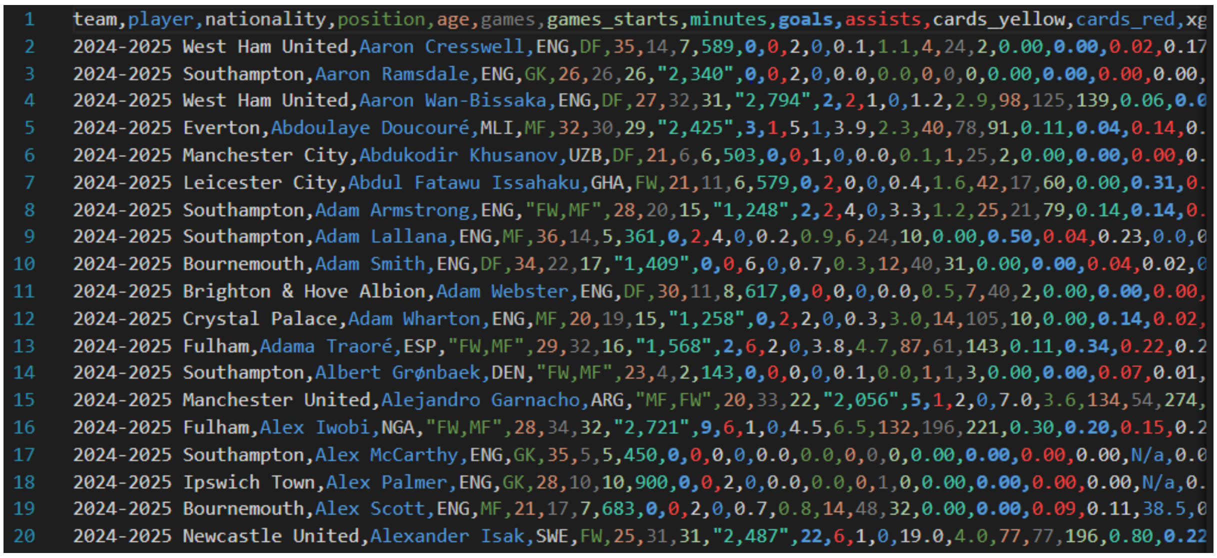
e) Merge Keys
The key columns used for merging the data are player and team. These two columns act as the primary identifiers across all statistical groups. The merging process is implemented sequentially using the `merge()` function provided by the pandas library.

f) Merging Process
For each statistical group (excluding the `stats_standard` group, which serves as the base), the script selects the relevant columns for merging. Any columns that are already present in the base DataFrame are excluded to avoid duplication, while new columns are added. The merging operation is performed using a left join, ensuring that all rows from the base dataset (i.e., all players) are retained, even if corresponding data in other stat groups is missing.

g) Cleaning the Data
After merging, the following cleaning steps are performed:

- Missing values are filled with “N/a”.
- Players who have played less than 90 minutes are filtered out.
- The columns are reordered according to the target list.

h) Final Output
The final cleaned and merged DataFrame is sorted by the player column to ensure consistent ordering of the dataset. Column names are then renamed using a predefined mapping dictionary named `target_name_dict`. Finally, the complete dataset is exported and saved as a CSV file with the filename `results.csv`.



```
team,player,nationality,position,age,games,games_starts,minutes,goals,assists,cards_yellow,cards_red,xg
2024-2025 West Ham United,Aaron Cresswell,ENG,DF,35,14,7,589,0,0,2,0,0.1,1.1,4,24,2,0.00,0.00,0.02,0.17
2024-2025 Southampton,Aaron Ramsdale,ENG,GK,26,26,26,"2,340",0,0,2,0,0.0,0.0,0,0,0,0.00,0.00,0.00,0.00,
2024-2025 West Ham United,Aaron Wan-Bissaka,ENG,DF,27,32,31,"2,794",2,2,1,0,1.2,2.9,98,125,139,0.06,0.0
2024-2025 Everton,Abdoulaye Doucouré,MLI,MF,32,30,29,"2,425",3,1,5,1,3.9,2.3,40,78,91,0.11,0.04,0.14,0.
2024-2025 Manchester City,Abdukodir Khusanov,UZB,DF,21,6,6,503,0,0,1,0,0.0,0.1,1,25,2,0.00,0.00,0.00,0.
2024-2025 Leicester City,Abdul Fatawu Issahaku,GHA,FW,21,11,6,579,0,2,0,0,0.4,1.6,42,17,60,0.00,0.31,0.
2024-2025 Southampton,Adam Armstrong,ENG,"FW,MF",28,20,15,"1,248",2,2,4,0,3.3,1.2,25,21,79,0.14,0.14,0.
2024-2025 Southampton,Adam Lallana,ENG,MF,36,14,5,361,0,2,4,0,0.2,0.9,6,24,10,0.00,0.50,0.04,0.23,0.0,0
2024-2025 Bournemouth,Adam Smith,ENG,DF,34,22,17,"1,409",0,0,6,0,0.7,0.3,12,40,31,0.00,0.00,0.04,0.02,0
2024-2025 Brighton & Hove Albion,Adam Webster,ENG,DF,30,11,8,617,0,0,0,0,0.0,0.5,7,40,2,0.00,0.00,0.00,
2024-2025 Crystal Palace,Adam Wharton,ENG,MF,20,19,15,"1,258",0,2,2,0,0.3,3.0,14,105,10,0.00,0.14,0.02,
2024-2025 Fulham,Adama Traoré,ESP,"FW,MF",29,32,16,"1,568",2,6,2,0,3.8,4.7,87,61,143,0.11,0.34,0.22,0.2
2024-2025 Southampton,Albert Grønbaek,DEN,"FW,MF",23,4,2,143,0,0,0,0,0.1,0.0,1,1,3,0.00,0.00,0.07,0.01,
2024-2025 Manchester United,Alejandro Garnacho,ARG,"MF,FW",20,33,22,"2,056",5,1,2,0,7.0,3.6,134,54,274,
2024-2025 Fulham,Alex Iwobi,NGA,"FW,MF",28,34,32,"2,721",9,6,1,0,4.5,6.5,132,196,221,0.30,0.20,0.15,0.2
2024-2025 Southampton,Alex McCarthy,ENG,GK,35,5,5,450,0,0,0,0,0.0,0.0,0,0,0,0.00,0.00,0.00,0.00,N/a,0.0
2024-2025 Ipswich Town,Alex Palmer,ENG,GK,28,10,10,900,0,0,2,0,0.0,0.0,0,1,0,0.00,0.00,0.00,0.00,N/a,0.
2024-2025 Bournemouth,Alex Scott,ENG,MF,21,17,7,683,0,0,2,0,0.7,0.8,14,48,32,0.00,0.00,0.09,0.11,38.5,0
2024-2025 Newcastle United,Alexander Isak,SWE,FW,25,31,31,"2,487",22,6,1,0,19.0,4.0,77,77,196,0.80,0.22
```

3. Data analysis

3.1. Identify the top 3 and bottom 3 players in each statistical category

This section presents the methodology and results from the analysis of a player statistical dataset. The goal is to determine individual and team performance through descriptive statistics, rankings, and data visualization.

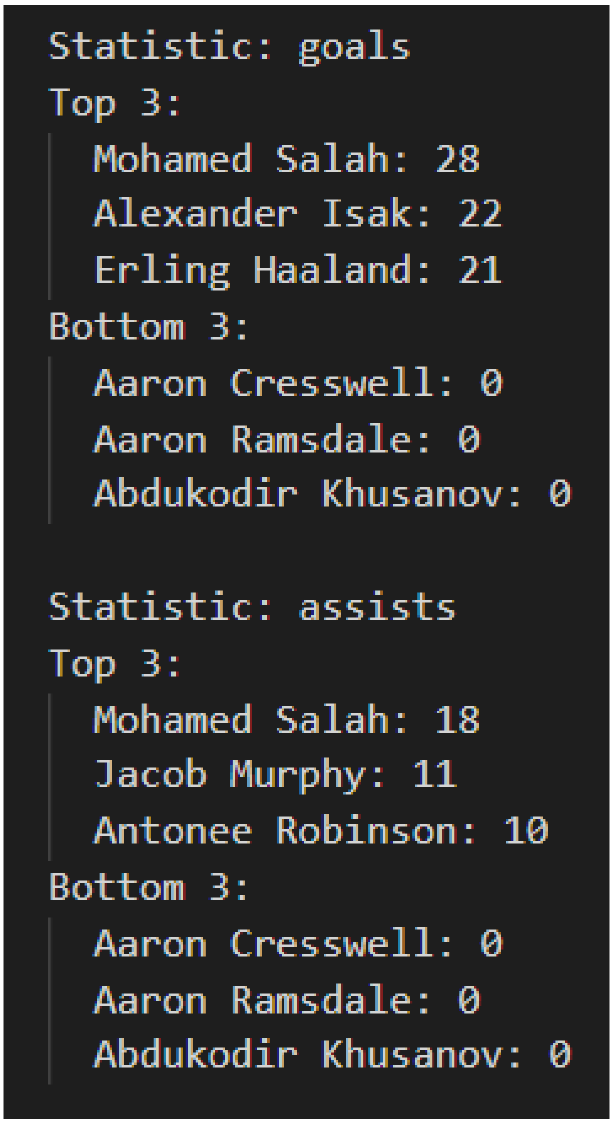
a) Loading the Data
The script starts by importing the necessary libraries: pandas for data manipulation, numpy for numerical operations, and matplotlib.pyplot for plotting. The dataset is then loaded from a file named `results.csv` using `pandas`. This dataset contains various statistics for football players, with rows representing players and columns representing different performance metrics and metadata (e.g., name, team, nationality).

b) Converting Columns to Numeric
All non-numeric entries, marked as “N/a”, are converted to “NaN” to enable accurate statistical calculations. The analysis focuses only on columns with numeric values, eliminating identifying information such as player name, nationality, and team. To ensure statistical calculations can be performed, convert all relevant columns to numeric data types.

c) Finding Top and Bottom Performers
This section describes the method used to determine the top 3 and bottom 3 players for each statistical metric. The process is implemented using pandas DataFrame functionalities and involves the following steps:

- *Eliminate Missing Values (NaN)*: Each column is checked for missing values. If a column consists entirely of `NaN` values, it is excluded from the analysis to prevent misleading or biased results. For columns with partial missing data, rows with `NaN` values are temporarily excluded during sorting.
- *Sort Data in Descending Order*: For each statistical column, the data is sorted from high to low. This allows identification of the top 3 players with the highest values—representing the best performers for that metric.
- *Sort Data in Ascending Order*: Similarly, the column is sorted from low to high to identify the 3 players with the lowest values, representing the least effective performers in that category.
- *Extract Top 3 Players*: After sorting, the top (or bottom) 3 rows are selected using `.head(3)`.

- *Format Results*: The final output is formatted as strings to clearly show the top and bottom performers. Each entry is presented in the format `<Player>:<Value>`.



```
Statistic: goals
Top 3:
  Mohamed Salah: 28
  Alexander Isak: 22
  Erling Haaland: 21
Bottom 3:
  Aaron Cresswell: 0
  Aaron Ramsdale: 0
  Abdukodir Khusanov: 0

Statistic: assists
Top 3:
  Mohamed Salah: 18
  Jacob Murphy: 11
  Antonee Robinson: 10
Bottom 3:
  Aaron Cresswell: 0
  Aaron Ramsdale: 0
  Abdukodir Khusanov: 0
```

d) Conclusion

This method effectively identifies the top 3 highest and lowest performing players for each statistical metric. Insights Provided:

- Player performance in each statistical category.
- Easy identification of standout performers.
- Detection of areas where players may need improvement.

Key Benefits:

- Saves time compared to manual analysis.
- Provides quantitative insights that are objective and data-driven.

- Applicable to diverse sports datasets, making it a versatile tool for performance analysis.

3.2. Find the median, mean, and standard deviation

a) Loading the Data
The analysis begins by loading a synthetic dataset containing soccer player performance statistics. This dataset is stored in a "results.csv" file and read into a pandas DataFrame.

b) Converting Columns to Numeric
All non-numeric entries, marked as "N/a", are converted to "NaN" to enable accurate statistical calculations. The analysis focuses only on columns with numeric values, eliminating identifying information such as player name, nationality, and team. To ensure statistical calculations can be performed, convert all relevant columns to numeric data types.

c) Calculating the Median, Mean, and Standard Deviation
This section explains how to compute three essential statistical measures for each performance metric:

- *Median*: Using the `median()` method from the pandas library to calculate the median value, which represents the middle value in a data set. It is especially valuable when the data includes outliers, as it is less affected by extreme values compared to the mean.
- *Mean (Average)*: Using the `mean()` function to calculate the mean provides an overall view of the average performance of all players, helping to determine typical performance levels.
- *Standard Deviation*: Using the `std()` function to calculate the spread of values. A high standard deviation indicates large variation in performance between players, while a low value implies consistency.
- *Scope of Calculation*: The statistics can be calculated for the entire league (to get a league-wide perspective) or for individual teams (to assess team-specific performance).

d) Calculating Statistics for Each Team
This section describes how to compute team-level statistics, which helps compare performance across different teams.

Procedure:

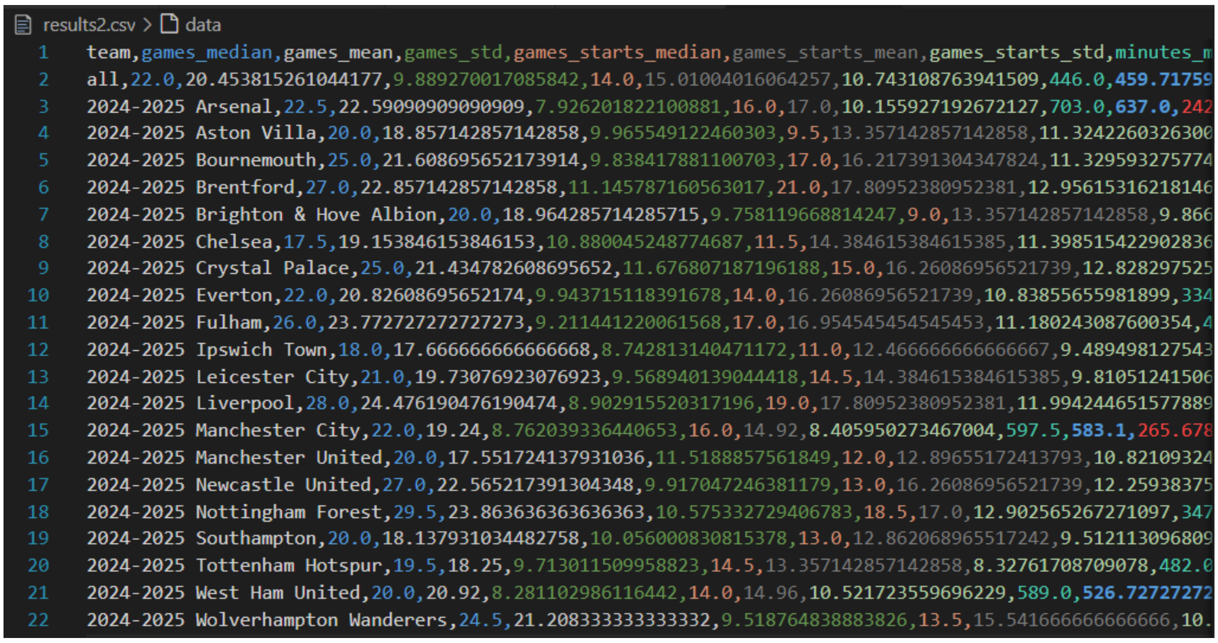
- The data is grouped using the Team column.
- Numeric columns are selected for statistical analysis.

Calculate Mean and Standard Deviation:

- The `<.agg()>` function is used to compute both `mean` and `std` (standard deviation) for each numeric column within each team group.
- The `<reset_index()>` method ensures the result is formatted as a DataFrame with a default index.

e) Creating the Final DataFrame
This part covers organizing and saving the calculated results: Steps:

- *Create a New DataFrame*: A new DataFrame is created from the calculated results. It includes columns: `'Statistic'`, `'Value'`, `'Index'`, and `'Team'`.
- *Save the DataFrame to CSV*: The results are saved in a CSV file named `results2.csv` for future use or analysis. The `index=False` parameter ensures that the row index is not saved in the file. UTF-8 encoding guarantees compatibility with most systems.



```
team,games_median,games_mean,games_std,games_starts_median,games_starts_mean,games_starts_std,minutes_m
all,22.0,20.453815261044177,9.889270017085842,14.0,15.01004016064257,10.743108763941509,446.0,459.71759
2024-2025 Arsenal,22.5,22.59090909090909,7.926201822100881,16.0,17.0,10.155927192672127,703.0,637.0,242
2024-2025 Aston Villa,20.0,18.857142857142858,9.965549122460303,9.5,13.357142857142858,11.3242260326300
2024-2025 Bournemouth,25.0,21.608695652173914,9.838417881100703,17.0,16.217391304347824,11.329593275774
2024-2025 Brentford,27.0,22.857142857142858,11.145787160563017,21.0,17.80952380952381,12.95615316218146
2024-2025 Brighton & Hove Albion,20.0,18.964285714285715,9.758119668814247,9.0,13.357142857142858,9.866
2024-2025 Chelsea,17.5,19.153846153846153,10.880045248774687,11.5,14.384615384615385,11.398515422902836
2024-2025 Crystal Palace,25.0,21.434782608695652,11.676807187196188,15.0,16.26086956521739,12.828297525
2024-2025 Everton,22.0,20.82608695652174,9.943715118391678,14.0,16.26086956521739,10.83855655981899,334
2024-2025 Fulham,26.0,23.772727272727273,9.211441220061568,17.0,16.954545454545453,11.180243087600354,4
2024-2025 Ipswich Town,18.0,17.666666666666668,8.742813140471172,11.0,12.466666666666667,9.489498127543
2024-2025 Leicester City,21.0,19.73076923076923,9.568940139044418,14.5,14.384615384615385,9.81051241506
2024-2025 Liverpool,28.0,24.476190476190474,8.902915520317196,19.0,17.80952380952381,11.994244651577889
2024-2025 Manchester City,22.0,19.24,8.762039336440653,16.0,14.92,8.405950273467004,597.5,583.1,265.678
2024-2025 Manchester United,20.0,17.551724137931036,11.5188857561849,12.0,12.89655172413793,10.82109324
2024-2025 Newcastle United,27.0,22.565217391304348,9.917047246381179,13.0,16.26086956521739,12.25938375
2024-2025 Nottingham Forest,29.5,23.863636363636363,10.575332729406783,18.5,17.0,12.90256526727 1097,347
2024-2025 Southampton,20.0,18.137931034482758,10.056000830815378,13.0,12.862068965517242,9.51211309680 9
2024-2025 Tottenham Hotspur,19.5,18.25,9.713011509958823,14.5,13.357142857142858,8.32761708709078,482.0
2024-2025 West Ham United,20.0,20.92,8.281102986116442,14.0,14.96,10.521723559696229,589.0,526.72727272
2024-2025 Wolverhampton Wanderers,24.5,21.208333333333332,9.518764838883826,13.5,15.541666666666666,10.
```

f) Conclusion
This section emphasizes the purpose and value of using basic statistical indicators—`median, mean, and standard deviation`—to evaluate performance metrics in a sports dataset. Key Takeaways:

- *Overall Performance Assessment*: These statistics help evaluate how players and teams performed in the tournament. It gives a broad understanding of strengths and weaknesses at both individual and team levels.
- *Understanding Distribution and Variation*: By calculating standard de-

viation, we can observe how much variation there is in performance. This is crucial for detecting consistency vs. inconsistency across players or teams.
- *Team vs. League Comparisons*: The data enables benchmarking individual team performance against the league average. This comparative view highlights which teams are above or below average in specific metrics.

These statistical insights allow analysts to better understand player and team quality. They form a foundation for data-driven decisions, such as talent scouting, strategy planning, or identifying areas needing improvement.

3.3. Draw histograms to show the distribution of each statistic

a) Loading and Preparing the Data
The process begins by importing player performance data from the `results.csv` file. The goal at this stage is to isolate the numeric/statistical features relevant for analysis:

- Non-numeric columns like `'Player'`, `'Nation'`, `'Position'`, and `'Team'` are excluded.
- Only columns with actual performance metrics (e.g., goals, assists, passes) are selected for visualization.
- A key step here is converting all relevant columns to numeric format and replacing missing or invalid values with zeros. This ensures data consistency and all metrics can be plotted without runtime errors.

b) Setting Up Directories for Saving Histograms
Before generating visualizations, the script organizes the output using a directory structure for clarity. A main folder named `histograms` is created. Inside it:

- A `league` subfolder holds charts representing the entire dataset.
- A `team` sub-folder contains team-specific charts, with each team having its own subdirectory.

Purpose of this structure:

- Enhances organization.
- Makes future retrieval and comparison of charts more intuitive.
- Prepares for scalability, such as adding charts for player positions or

nationalities later.

c) Plotting the Histogram for All Players
Once the folders are set up, the script proceeds to generate league-wide histograms for each numeric statistic.

Process:

- Loop through all numeric columns.
- Use `matplotlib` to create a histogram that visualizes how often different values appear in the dataset.
- Save the chart into the `histograms/league` folder

Purpose and Insight:

- Understand overall performance trends (e.g., how many players score very high vs. very low).
- Spot outliers (e.g., a few players with extremely high values).
- Establish a performance baseline across the entire league.

This step gives a macro-level view of player statistics.

d) Plotting Team-Specific Histograms
After analyzing league-wide data, the focus shifts to individual team performance.

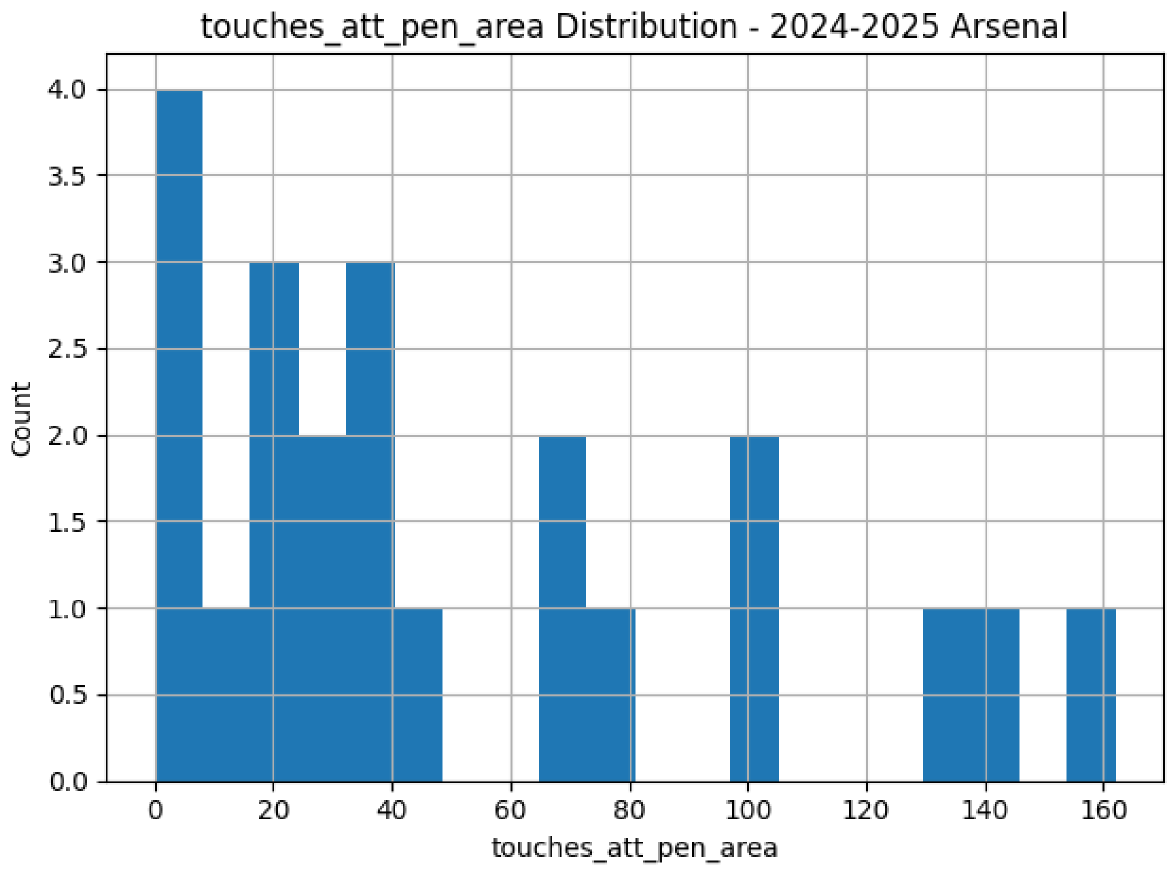
Process: For each team:

- Filter the dataset to include only players from that team.
- Create a subdirectory inside `histograms/teams` using the team's name (spaces replaced with underscores).
- Loop through all numeric stats and generate histograms.
- Save each histogram to the respective team's folder.

Benefits of Team-Level Analysis:

- Detects internal variation in team performance.
- Helps identify if performance is dominated by a few players or evenly distributed.
- Useful for comparing team structures and strategies.

For example, a team with one player scoring most of the goals will have a histogram skewed toward that player, while balanced teams will show a more uniform distribution.



e) Conclusion

The methodology for creating histograms allows visualization of the distribution of player statistics, both for the league and for individual teams. By creating these histograms, we can better understand the distribution of different performance metrics and draw data-driven insights about the league and teams. The histograms are stored in organized folders for easy access and further analysis.

3.4. Determine the player with the highest overall score

a) Aggregating Team Statistics

The first step in evaluating team performance is summing all player statistics within each team:

- This groups the dataset by the `Team` column.
- Then, it computes the total output for each statistic (e.g., total goals, total assists) across all players in that team.
- The result is a table showing how each team performs collectively, which forms the foundation for identifying top teams per metric.

This approach is effective for comparing overall productivity between teams.

b) Identifying Top Teams for Each Statistic
Next, the script identifies the team with the highest value for each statistic:

- This loop skips any statistic that has no data and finds the team with the maximum sum for each valid statistic using `<idxmax()>`.
- The result is a dictionary mapping each statistic to the top-performing team for that metric.

This stage highlights specialization: some teams may dominate certain areas, such as defense or passing.

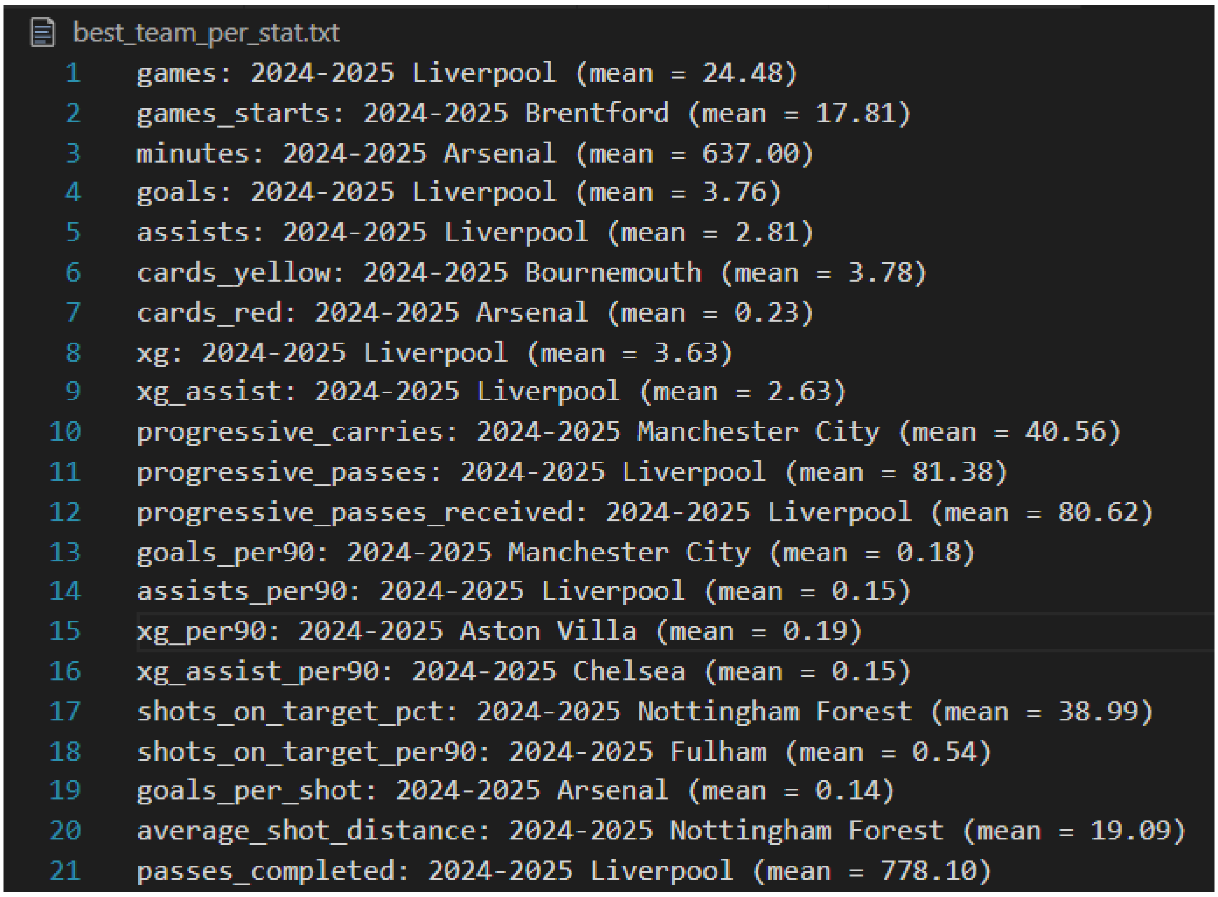
c) Determining Overall Best-Performing Team
To assess which team performs best across all statistics, the code tallies how often each team appears at the top:

- This produces a ranked list showing how frequently each team led a particular stat.
- The team with the highest count is considered the overall best-performing team.

This technique rewards broad dominance, not just excelling in one area.

d) Outputting Results
The analysis then outputs:

- The best team for each individual statistic
- The overall best-performing team


best_team_per_stat.txt

```
games: 2024-2025 Liverpool (mean = 24.48)
games_starts: 2024-2025 Brentford (mean = 17.81)
minutes: 2024-2025 Arsenal (mean = 637.00)
goals: 2024-2025 Liverpool (mean = 3.76)
assists: 2024-2025 Liverpool (mean = 2.81)
cards_yellow: 2024-2025 Bournemouth (mean = 3.78)
cards_red: 2024-2025 Arsenal (mean = 0.23)
xg: 2024-2025 Liverpool (mean = 3.63)
xg_assist: 2024-2025 Liverpool (mean = 2.63)
progressive_carries: 2024-2025 Manchester City (mean = 40.56)
progressive_passes: 2024-2025 Liverpool (mean = 81.38)
progressive_passes_received: 2024-2025 Liverpool (mean = 80.62)
goals_per90: 2024-2025 Manchester City (mean = 0.18)
assists_per90: 2024-2025 Liverpool (mean = 0.15)
xg_per90: 2024-2025 Aston Villa (mean = 0.19)
xg_assist_per90: 2024-2025 Chelsea (mean = 0.15)
shots_on_target_pct: 2024-2025 Nottingham Forest (mean = 38.99)
shots_on_target_per90: 2024-2025 Fulham (mean = 0.54)
goals_per_shot: 2024-2025 Arsenal (mean = 0.14)
average_shot_distance: 2024-2025 Nottingham Forest (mean = 19.09)
passes_completed: 2024-2025 Liverpool (mean = 778.10)
```

e) Conclusion

This final phase of the script provides a holistic and comparative performance analysis of teams, offering insight into:

- Which teams dominate specific metrics
- Which team performs best overall
- How consistently teams perform across the board (via std and median)

By integrating sum totals, means, medians, and standard deviations, this approach offers a multi-dimensional view of team performance—ideal for scouting, strategy, and performance evaluation in professional soccer analytics.

4. Plot 2D visualization of statistical data

In this section, we outline the methodology used to perform clustering on football player statistics using the K-means algorithm and visualize the resulting clusters in a two-dimensional space.

a) Overview of K-means Clustering
K-means clustering is an unsupervised machine learning algorithm that partitions the data into a predefined number of clusters. Each cluster is represented by its centroid, and data points are assigned to the cluster whose centroid is nearest. The K-means algorithm minimizes the within-cluster sum of squares (inertia) and iterates until convergence.

The goal of this analysis is to group players based on their performance metrics and categorize them into distinct clusters. These clusters can be visualized and labeled based on common features that are prominent within each group. In the context of football player data, the clusters typically represent different player roles such as goalkeepers, defenders, and offensive players.

b) Preprocessing the Data
Before applying the K-means clustering algorithm, several preprocessing steps are executed on the dataset:

- *Handling Non-Numeric Data*: Columns containing non-numeric information such as player names, team names, and positions are excluded from the dataset. These fields are irrelevant for clustering since the K-means algorithm operates solely on numerical performance metrics.
- *Age Conversion*: Player age values are initially stored as strings (e.g., "25.4 years"). The script extracts the numerical component and converts it into a float, making it suitable for numerical analysis and clustering.
- *Standardization*: The dataset is standardized using the `StandardScaler` from the `sklearn.preprocessing` module. This step is crucial because K-means clustering is sensitive to the scale of input features. Standardization ensures that all features have a mean of 0 and a standard deviation of 1, which prevents any single feature from disproportionately influencing the clustering process.

c) Determining the Optimal Number of Clusters
To determine the optimal number of clusters for K-means, the script utilizes the elbow method. This approach involves calculating the inertia (sum of

squared distances between data points and their nearest cluster center) for various values of **k** (the number of clusters). The optimal k is identified at the point where the rate of decrease in inertia slows down significantly, forming an "elbow" in the graph. In this analysis, the chosen value for **k** is 4.

d) Performing K-means Clustering
After identifying the optimal number of clusters (k = 4), the script applies the **K-means clustering algorithm**. The algorithm partitions the data into three clusters based on player performance metrics. Each data point is assigned to the nearest cluster based on its proximity to the cluster centroid. The assignment process is refined through multiple iterations to improve the accuracy of clustering.

e) Labeling the Clusters
Once clustering is complete, descriptive labels are assigned to each cluster to enhance interpretability. In this case, three primary player roles are used:

- *Goalkeepers*: Players specialized in shot-saving and goal prevention.
- *Defensive Players*: Players focused on defense, including making tackles and interceptions.
- *Offensive Players*: Players involved in scoring and attacking play.

The cluster centroids are analyzed by comparing key feature values to support the labeling process.

f) Visualizing the Clusters in 2D
The next step is to visualize the clusters in two dimensions to better understand the grouping of the data points. This is achieved using Principal Component Analysis (PCA), a dimensionality reduction technique that transforms the data into a lower-dimensional space while retaining as much variance as possible.

- *Data Preprocessing*: The dataset is first loaded from a CSV file `results.csv`. Only numeric columns are selected for clustering, and any rows containing missing values are removed to ensure data integrity. The data is then standardized using `StandardScaler` to normalize the features.
- *Choosing the Number of Clusters (Elbow Method)*: To determine the optimal number of clusters, the Elbow Method is employed. A loop runs **KMeans** clustering for k ranging from 2 to 10, and the sum of squared errors (SSE) is recorded for each k. The resulting SSE values are plotted to visually identify the “elbow point,” which indicates a suitable number of clusters. In this example, k = 4 is chosen.

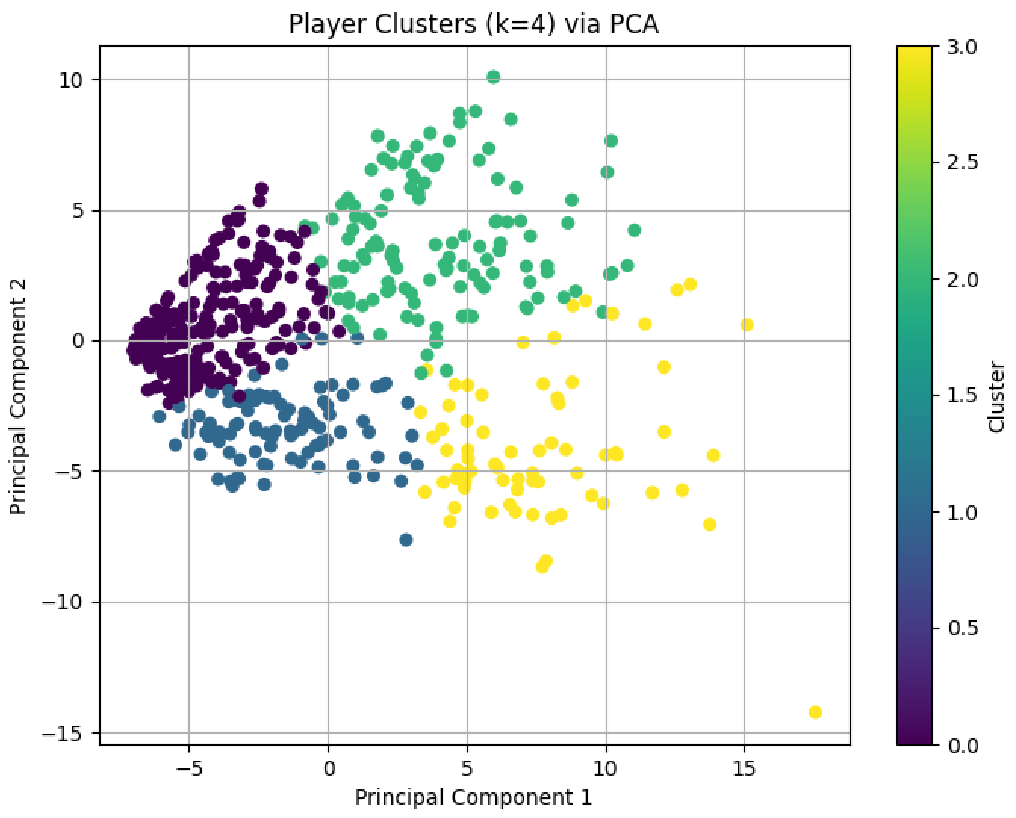
- *Clustering with KMeans*: Using the optimal value **k = 4**, the **KMeans** algorithm is applied to the standardized data to group the players into four distinct clusters. Each player is assigned a cluster label based on the algorithm's result.
- *PCA Transformation*: PCA is applied to the standardized data to reduce the dimensionality to two principal components. This transformation allows for a 2D visualization of the high-dimensional data while preserving the core variance and structure.
- *Plotting the Clusters*: The 2D PCA-transformed data is plotted using a scatter plot. Each point is color-coded according to its assigned cluster. The x-axis and y-axis represent the first and second principal components, respectively, which capture the majority of the variance in the data.
- *Cluster Visualization*: A color bar is added to the plot to indicate cluster labels. The scatter plot provides a visual representation of how the KMeans algorithm has grouped the players, helping to interpret the underlying structure of the dataset.

This resulting visualization offers a clear and intuitive view of the player clusters, enabling better insights into how players group based on similar performance characteristics.

g) Interpreting the Results

Once the 2D plot is generated, the clusters can be analyzed to draw insights about the players' performance:

- *Separation between clusters*: provides insight into the distinct roles and playing styles of different types of players. Ideally, the clusters should be well-separated, indicating that the features used in the clustering process are effective in distinguishing between player types.
- *Density of the points within each cluster*: provide information about how similar the players are within each group. Dense clusters suggest that the players within that group share similar performance statistics, while sparse clusters may indicate more variability within the group.
- *Position distribution across clusters*: allows us to verify whether the clustering aligns with the actual player positions (e.g., goalkeepers being grouped together, offensive players in another group). This helps validate the clustering results and supports meaningful interpretation based on real-world player roles.



h) Conclusion

The K-means clustering algorithm, combined with PCA for dimensionality reduction, provides an effective way to group football players based on their performance statistics. The 2D visualization of the clusters offers a clear representation of how players from different positions and playing styles are grouped. This methodology not only aids in understanding player roles but also provides a foundation for further analysis of player performance across different metrics.

The results can be used to identify patterns in player performance, highlight strengths and weaknesses within specific clusters, and inform strategies for team composition based on player characteristics.

5. Conclusion

a) Overview of K-means Clustering
This project systematically analyzed football player performance data to assess team efficiency, categorize player roles, and predict market values using machine learning techniques. By leveraging methods such as data preprocessing, K-means clustering, PCA, and Random Forest Regression, we constructed a comprehensive pipeline that connects performance statistics with role classification and economic evaluation. Visualization techniques were used to enhance the interpretability of the results, ultimately offering deeper insights into the key factors influencing a player's market value.

b) Overview of K-means Clustering
Throughout the execution of the project, we gained valuable hands-on experience in various domains of data science and football analytics. Key learning outcomes include:

- Gaining proficiency in collecting and extracting data from online sources.
- Cleaning and standardizing real-world sports data for analysis.
- Applying clustering methods to discover natural groupings of players based on performance.
- Using regression models to predict target outcomes, such as player market value.
- Employing PCA and visualization techniques to better interpret complex data.
- Drawing meaningful conclusions from statistical patterns within the football context.

c) Future Improvements
Despite the effectiveness of the current approach, several directions can be pursued to enhance the model's utility and performance:

- Expanding the dataset to include more seasons, leagues, and player samples to improve model robustness and generalizability.
- Integrating contextual features such as injuries, match time, or team economic data for a more holistic analysis.
- Experimenting with advanced regression techniques like XGBoost or deep learning models to increase prediction accuracy.
- Developing an interactive dashboard to allow stakeholders to dynamically explore player statistics and cluster insights.

Overall, this work contributes a solid base for data-driven decision-making in modern football, supporting talent identification, performance evaluation, and market valuation strategies.