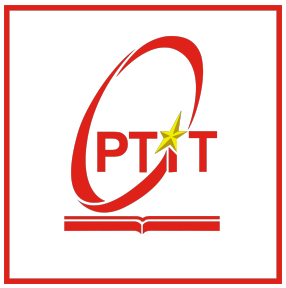
POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY 1



PYTHON PROGRAMMING

ASSIGNMENT 1

REPORT OF FOOTBALL DATA ANALYSIS

Instructor: Kim Ngoc Bach
Student: Nguyen Manh Hung
Student ID: B23DCCE040
Class ID: D23CQCE04-B

Hanoi

Table of contents

Chapter 1

Introduction

1.1 Abstract

This report covers the collection and analysis of comprehensive performance statistics for 2024-2025 English Premier League players (played > 90 min), sourced from fbref.com. Statistical analysis is used to identify key performers, calculate descriptive metrics, and visualize data distributions. Machine learning techniques, including K-means clustering and PCA, are applied to group players and visualize these classifications. The study also explores a method to estimate player transfer values based on the collected data. The findings provide data-driven insights into player and team performance.

1.2 Goal

The main goal of this report is to collect, analyze, and model statistics of English Premier League players using Python. This involves identifying key players and team trends, classifying players using machine learning (K-means, PCA), and proposing a method to estimate player transfer values.

Chapter 2

Background Knowledge

2.1 Python

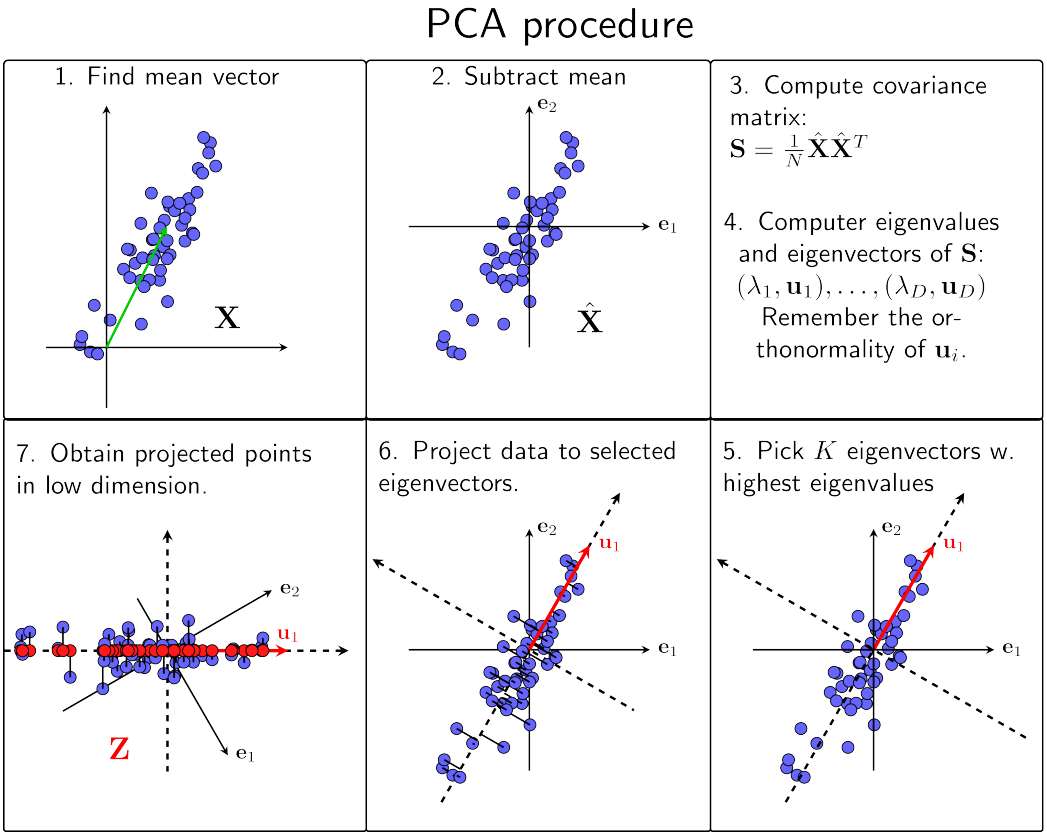
In this assignment, Python is an essential tool used for collecting football data via web scraping, performing statistical analysis, implementing machine learning models like K-means, and visualizing results. Here are applications of this programming language in the report:

- **Data Collection and Processing**: Python facilitates the extraction of football player statistics from HTML sources, employing libraries such as BeautifulSoup and Selenium. In the data processing phase, Python provides robust frameworks (Numpy, Pandas) that support essential operations such as data cleaning, type conversion, array manipulation, handling missing values, etc.
- **Data Analysis**: This phase employs Python, with libraries like Pandas and NumPy, to conduct statistical analysis on the processed data. Key tasks include calculating descriptive statistics (mean, median, standard deviation) for each player metric, identifying the top and bottom three players per statistic, and analyzing team-level statistics to determine leading teams and assess overall performance in the league.
- **Data Visualization**: Python libraries like Matplotlib and Seaborn are employed to create visual representations of data insights. This includes generating histograms to illustrate the distribution of various player statistics and plotting 2D scatter plots to visualize player clusters identified through machine learning techniques.
- **Machine Learning**: Python's Scikit-learn library is utilized to apply machine learning algorithms. This includes using the K-means algorithm for unsupervised clustering to classify players into groups based on their statistics, and Principal Component Analysis (PCA) for dimensionality reduction, which also aids in visualizing these clusters. Furthermore, this section involves proposing a method-

ology for estimating player transfer values, including considerations for feature selection and model choice.

2.2 PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a dataset into a smaller set of uncorrelated variables called principal components, while preserving most of the original variance. Each component captures decreasing amounts of variance in the data.



PCA is valuable for several reasons:

- **Reducing Complexity**: By reducing the number of dimensions, PCA simplifies models, can reduce computational costs, and helps in mitigating the "curse of dimensionality."
- **Feature Extraction**: It can create new, more informative features from the existing ones, which can sometimes lead to better performance in machine learning models.

- **Noise Reduction**: By focusing on components with higher variance, PCA can help filter out noise present in the data.
- **Data Visualization**: It allows for the projection of high-dimensional data into lower dimensions (typically 2D or 3D) that can be visually inspected and plotted, making it easier to identify patterns, clusters, or outliers.

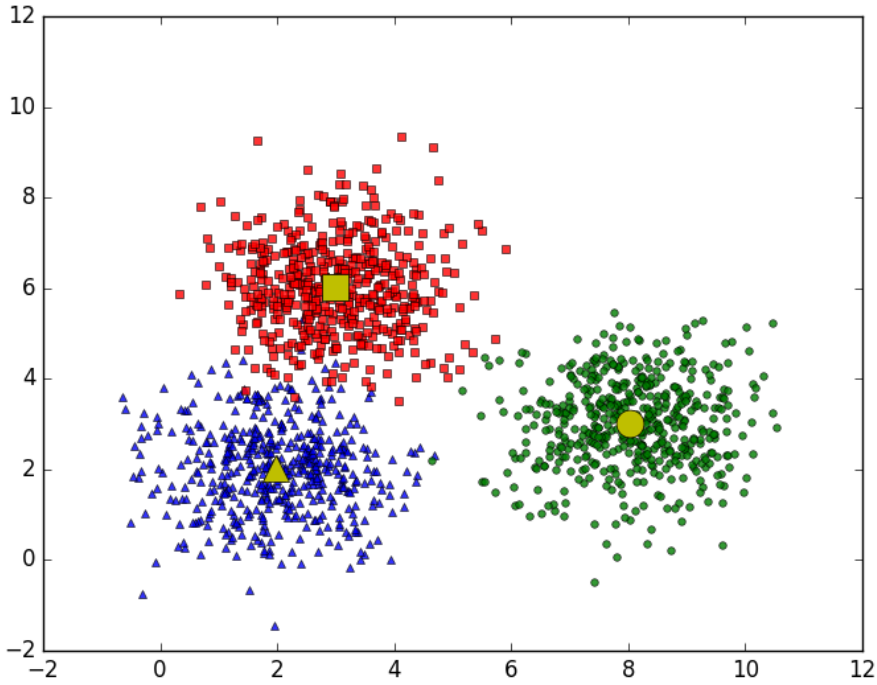
2.3 K-means

K-Means is a widely used unsupervised machine learning algorithm that partitions data into K distinct clusters. It groups similar data points together, aiming to maximize similarity within clusters and minimize similarity between them.

The algorithm operates iteratively through the following general steps:

1. **Specify K**: The number of clusters (K) to be formed is defined by the user.
2. **Initialize Centroids**: K initial centroids (the center point of each cluster) are randomly selected from the data points or generated.
3. **Assign Data Points**: Each data point in the dataset is assigned to the nearest centroid, typically based on Euclidean distance. This forms K initial clusters.
4. **Update Centroids**: The centroid of each cluster is recalculated as the mean of all data points assigned to that cluster.
5. **Repeat**: Steps 3 and 4 are repeated until the centroids no longer change significantly between iterations, or a maximum number of iterations is reached, indicating that the algorithm has converged.

A key aspect of using K-Means is determining the optimal number of clusters (K). Common methods for this include the Elbow method, which observes the point where adding more clusters provides diminishing returns in explaining the variance, or Silhouette analysis, which measures how similar an object is to its own cluster compared to other clusters.



In Exercise III, the K-Means algorithm is applied to football player statistics to group players based on their performance profiles. An appropriate value for K must be selected and justified. After clustering, the player groups will be analyzed and visualized in 2D using PCA.

2.4 Random Forest

Random Forest is a popular supervised machine learning algorithm from the ensemble learning family, used for both classification and regression. It works by building multiple decision trees during training and makes predictions by averaging their outputs (regression) or taking the majority vote (classification)..

Key Characteristics and Advantages:

- **Ensemble of Decision Trees:** It constructs many individual decision trees, each trained on a random subset of the training data (bagging) and considering a random subset of features at each split.
- **Reduction of Overfitting:** By averaging the predictions of many trees, Random Forest generally has a lower risk of overfitting compared to a single decision tree, making it robust.

- **Handles High Dimensionality:** It can handle datasets with a large number of features effectively.
- **Feature Importance:** It provides a useful measure of feature importance, indicating which features are most influential in making predictions. This is valuable for feature selection and understanding the data.
- **Non-linearity:** It can capture complex non-linear relationships between features and the target variable.

Relevance to Player Value Estimation: In the context of estimating Player Transfer Values (a regression task in Exercise IV), the Random Forest Regressor can be a powerful model. Its ability to handle various types of features, capture non-linearities, and provide feature importances makes it suitable for identifying which player statistics most significantly contribute to their market value.

2.5 Gradient Boosing

Gradient Boosting is an ensemble learning method used for regression and classification. Unlike Random Forest, it builds trees sequentially, with each tree aiming to correct the errors of the previous one. **Key Characteristics and Advantages:**

- **Sequential Learning:** Trees are added one at a time, and each subsequent tree is trained on the residuals (the differences between the actual values and the predictions) of the preceding ensemble.
- **Gradient Descent Optimization:** It uses a gradient descent algorithm to minimize a loss function by iteratively adding weak learners (typically decision trees).
- **High Predictive Accuracy:** Gradient Boosting models are often among the best-performing algorithms for tabular data, capable of achieving high accuracy.
- **Flexibility:** It can optimize various loss functions and provides several hyperparameters for tuning, allowing it to be adapted to different types of problems.
- **Handles Missing Values:** Some implementations can handle missing values inherently.

Relevance to Player Value Estimation: For predicting player ETVs, a Gradient Boosting Regressor (like Scikit-learn's `GradientBoostingRegressor` or more advanced implementations like XGBoost, LightGBM, or CatBoost) can be very effective. Its ability to iteratively refine predictions and capture complex patterns makes it a strong candidate for modeling the intricate relationship between player performance and market value.

2.6 Ridge

Ridge Regression is a linear regression method that adds an L2 regularization term to the loss function. It helps address multicollinearity and prevent overfitting, particularly in models with many features.

Key Characteristics and Advantages:

- **Linear Model:** It assumes a linear relationship between the independent features and the target variable.
- **L2 Regularization (Shrinkage):** It adds a penalty term to the loss function equal to the square of the magnitude of the coefficients ($\lambda \sum \beta_j^2$, where λ is the regularization strength). This "shrinks" the coefficients towards zero, but they rarely become exactly zero.
- **Reduces Multicollinearity Impact:** By shrinking correlated coefficients, it can make the model more stable and less sensitive to minor changes in the input data.
- **Prevents Overfitting:** The regularization helps to reduce model complexity and prevent overfitting, especially when there are many features or when features are highly correlated.
- **Computational Efficiency:** It is generally computationally efficient to train.

Relevance to Player Value Estimation: Ridge Regression can be a useful baseline model for player value estimation, especially if multicollinearity is present in player statistics. It provides stable coefficient estimates compared to Ordinary Least Squares (OLS) and is often used alongside more complex models to assess trade-offs.

Chapter 3

Data Analysis

3.1 Exercise 1: Collecting and Processing Data

The Python script for this exercise, named `EX1.py`, is located in the primary project directory.

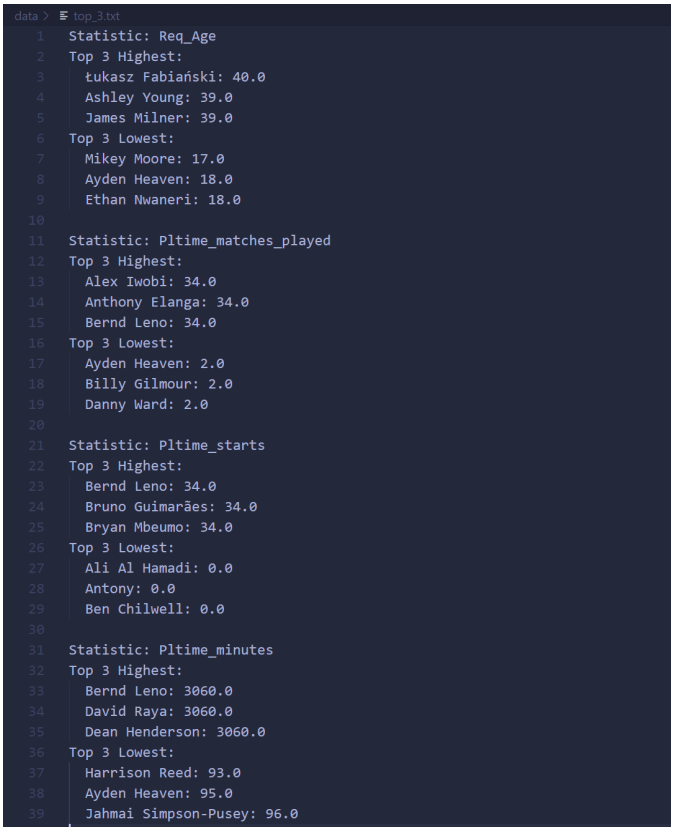
Here are the following details of my script:

- The dataset was scraped from FBref.com, focusing on players in the 2024-2025 English Premier League season who have played more than 90 minutes.
- The collected and processed data was then saved into a CSV file named `results.csv`.
- Tools I used:
  - `Selenium` and `Beautiful Soup` were employed for web scraping, specifically to access and parse HTML tables.
  - `undetected_chromedriver` is used to reduce the chances of triggering Cloudflare's bot detection mechanisms and challenges (like CAPTCHAs or "checking your browser" pages), helping to prevent the script from being blocked while collecting data.
  - `Pandas` was utilized for structuring the extracted data into a DataFrame, allowing efficient data manipulation, cleaning, and organization.
  - `NumPy` supported numerical operations and the management of missing data representations (NaN) within the DataFrame.
- Missing or inapplicable values encountered during scraping were consistently identified and marked as "N/a" in the final `results.csv` output.
- Player records in the final dataset were sorted alphabetically by the player's name.

- The column names for the various statistics were standardized by predefined naming conventions implemented within the scraping script, ensuring consistency in the output.
- Text-based statistics (like "2.1" for xG or "35" for shots) were changed into numbers (float or integer) for calculations; if a value couldn't be converted, it was marked "N/a
- Player age information was processed to retain only the year component (e.g., "29-233" was transformed to "29").
- All columns intended for numerical analysis were cast to `float` or `int` types, making them suitable for subsequent statistical computations.

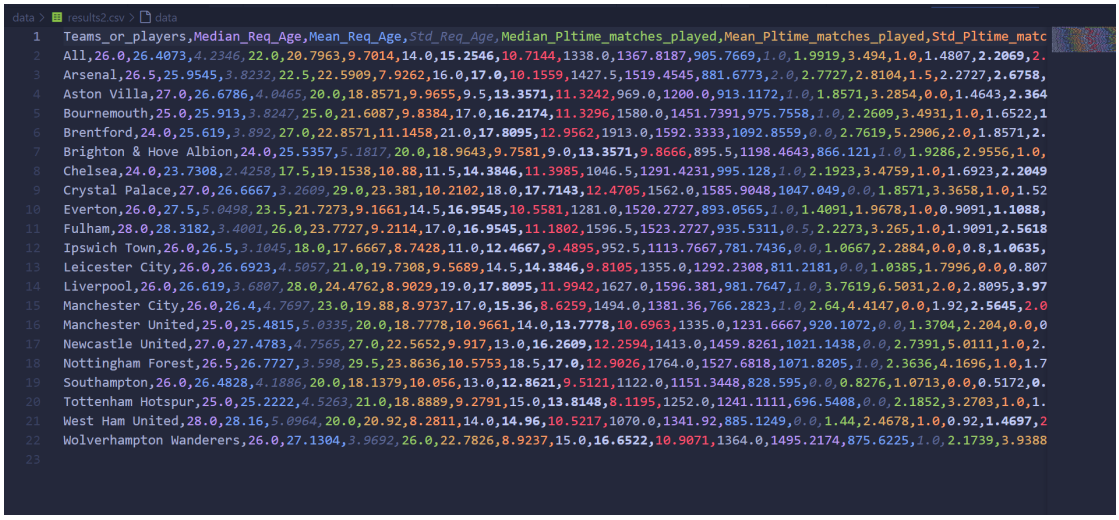
3.2 Exercise 2: Statistical Analysis

- **Top/Bottom Performers** The top 3 players with the highest scores and the bottom 3 players with the lowest scores were identified for each statistical metric (saved in the file `top_3.txt`)



- **Descriptive Statistics Calculation:** For each statistic, the median, the mean, and the standard deviation were calculated across two scopes:
  - Across all players in the league.
  - On a per-team basis.

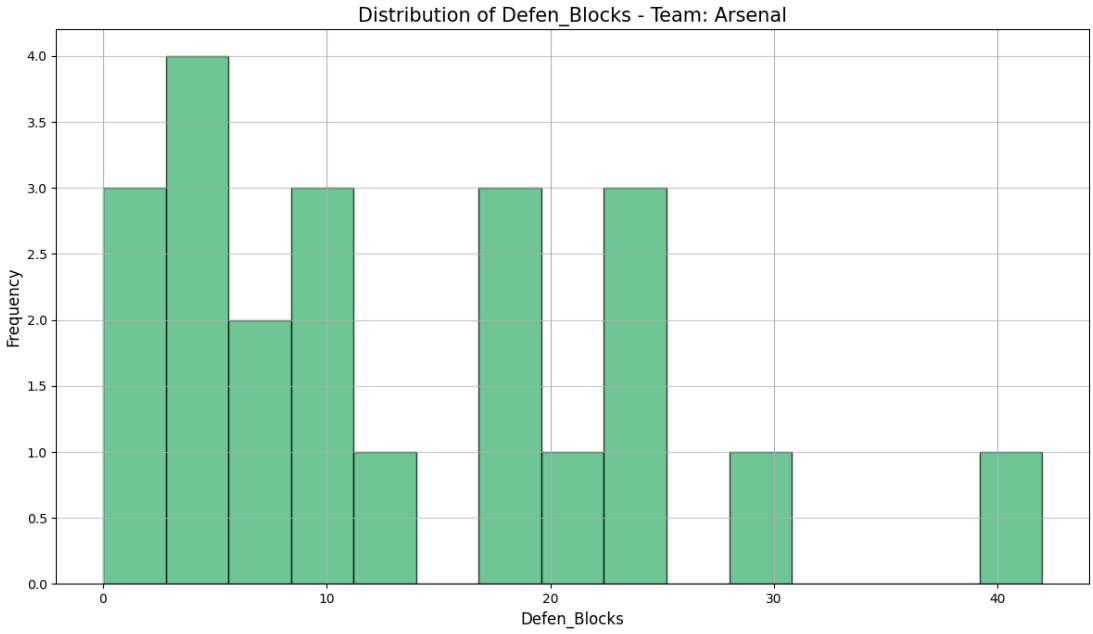
  (saved in the file `results2.csv`)



```
data > results2.csv > data
Teams_or_players,Median_Req_Age,Mean_Req_Age,Std_Req_Age,Median_Pltime_matches_played,Mean_Pltime_matches_played,Std_Pltime_matc
All,26.0,26.4073,4.2346,22.0,20.7963,9.7014,14.0,15.2546,10.7144,1338.0,1367.8187,905.7669,1.0,1.9919,3.494,1.0,1.4807,2.2069,2.
Arsenal,26.5,25.9545,3.8232,22.5,22.5909,7.9262,16.0,17.0,10.1559,1427.5,1519.4545,881.6773,2.0,2.7727,2.8104,1.5,2.2727,2.6758,
Aston Villa,27.0,26.6786,4.0465,20.0,18.8571,9.9655,9.5,13.3571,11.3242,969.0,1200.0,913.1172,1.0,1.8571,3.2854,0.0,1.4643,2.364
Bournemouth,25.0,25.913,3.8247,25.0,21.6087,9.8384,17.0,16.2174,11.3296,1580.0,1451.7391,975.7558,1.0,2.2609,3.4931,1.0,1.6522,1
Brentford,24.0,25.619,3.892,27.0,22.8571,11.1458,21.0,17.8095,12.9562,1913.0,1592.3333,1092.8559,0.0,2.7619,5.2906,2.0,1.8571,2.
Brighton & Hove Albion,24.0,25.5357,5.1817,20.0,18.9643,9.7581,9.0,13.3571,9.8666,895.5,1198.4643,866.121,1.0,1.9286,2.9556,1.0,
Chelsea,24.0,23.7308,2.4258,17.5,19.1538,10.88,11.5,14.3846,11.3985,1046.5,1291.4231,995.128,1.0,2.1923,3.4759,1.0,1.6923,2.2049
Crystal Palace,27.0,26.6667,3.2609,29.0,23.381,10.2102,18.0,17.7143,12.4705,1562.0,1585.9048,1047.049,0.0,1.8571,3.3658,1.0,1.52
Everton,26.0,27.5,5.0498,23.5,21.7273,9.1661,14.5,16.9545,10.5581,1281.0,1520.2727,893.0565,1.0,1.4091,1.9678,1.0,0.9091,1.1088,
Fulham,28.0,28.3182,3.4001,26.0,23.7727,9.2114,17.0,16.9545,11.1802,1596.5,1523.2727,935.5311,0.5,2.2273,3.265,1.0,1.9091,2.5618
Ipswich Town,26.0,26.5,3.1045,18.0,17.6667,8.7428,11.0,12.4667,9.4895,952.5,1113.7667,781.7436,0.0,1.0667,2.2884,0.0,0.8,1.0635,
Leicester City,26.0,26.6923,4.5057,21.0,19.7308,9.5689,14.5,14.3846,9.8105,1355.0,1292.2308,811.2181,0.0,1.0385,1.7996,0.0,0.807
Liverpool,26.0,26.619,3.6807,28.0,24.4762,8.9029,19.0,17.8095,11.9942,1627.0,1596.381,981.7647,1.0,3.7619,6.5031,2.0,2.8095,3.97
Manchester City,26.0,26.4,4.7697,23.0,19.88,8.9737,17.0,15.36,8.6259,1494.0,1381.36,766.2823,1.0,2.64,4.4147,0.0,1.92,2.5645,2.0
Manchester United,25.0,25.4815,5.0335,20.0,18.7778,10.9661,14.0,13.7778,10.6963,1335.0,1231.6667,920.1072,0.0,1.3704,2.204,0.0,0
Newcastle United,27.0,27.4783,4.7565,27.0,22.5652,9.917,13.0,16.2609,12.2594,1413.0,1459.8261,1021.1438,0.0,2.7391,5.0111,1.0,2.
Nottingham Forest,26.5,26.7727,3.598,29.5,23.8636,10.5753,18.5,17.0,12.9026,1764.0,1527.6818,1071.8205,1.0,2.3636,4.1696,1.0,1.7
Southampton,26.0,26.4828,4.1886,20.0,18.1379,10.056,13.0,12.8621,9.5121,1122.0,1151.3448,828.595,0.0,0.8276,1.0713,0.0,0.5172,0.
Tottenham Hotspur,25.0,25.2222,4.5263,21.0,18.8889,9.2791,15.0,13.8148,8.1195,1252.0,1241.1111,696.5408,0.0,2.1852,3.2703,1.0,1.
West Ham United,28.0,28.16,5.0964,20.0,20.92,8.2811,14.0,14.96,10.5217,1070.0,1341.92,885.1249,0.0,1.44,2.4678,1.0,0.92,1.4697,2
Wolverhampton Wanderers,26.0,27.1304,3.9692,26.0,22.7826,8.9237,15.0,16.6522,10.9071,1364.0,1495.2174,875.6225,1.0,2.1739,3.9388
```

- **Data Visualization - Histograms:** Histograms were created using `matplotlib.pyplot.hist` to visualize the distribution of selected player performance statistics.
  - Attacking statistics (xG, SCA90, G/Sh) and defensive statistics (Tkl, Blocks, Int) are used for histograms visualization
  - Histograms were created to show the distribution of each statistic across **all players** and **each individual team** in the league.

  The generated histogram images were saved to a dedicated directory (e.g., `Histograms/`). A sample histogram is displayed below, depicting the distribution of the "Blocks" statistic for the Arsenal football club.



- **Team Performance Analysis:** Overall team performance in the 2024-2025 Premier League season was assessed by identifying teams leading in statistical categories, using the Python script `EX2-p4.ipynb`.
  - **Methodology:**
    1. Player statistics were loaded from `results.csv`. All relevant statistical columns were converted to numeric types, with any conversion errors resulting in `NaN` values.
    2. The data was then grouped by `team`, and the `mean()` was calculated for each numeric statistic for every team. This resulted in a `team_stats` DataFrame showing average team performances.
    3. For each numeric statistic in `team_stats`, the team with the highest average value was identified using `idxmax()`. The corresponding maximum value was also retrieved.
    4. A dictionary (`top_teams`) was compiled, mapping each statistic to the team leading in that category and their average score.
    5. Finally, a summary was generated to count the number of distinct statistics in which each team ranked highest. Teams were then ranked based on this count to identify overall statistical leaders.
  - **Findings:** The analysis revealed that certain teams excelled in specific statistical areas. *Liverpool* led in performance goals (`Perf_goals`) with an

average of 3.76 and performance assists (`Perf_assists`) with an average of 2.81, while *Manchester City* topped pass completion percentage (`Pass_Cmp%`) with an average of 86.50%. *Crystal Palace* led in defensive tackles (`Defen_Tkl`) with an average of 32.62. A summary table highlighted the number of top categories each team dominated, offering a quantitative view of their statistical strengths.

```
Teams with most top statistics:
                         Team  Top Categories  \
9                  Liverpool              25
2            Manchester City              13
13            Crystal Palace               6
4                  Brentford               5
5                Bournemouth               5
6                    Fulham                4
7          Nottingham Forest               4
1                    Arsenal               3
0           Newcastle United               2
11            Leicester City               2
3                Southampton               1
8    Wolverhampton Wanderers               1
10                   Chelsea               1
12               Aston Villa               1
14                   Everton               1
```

– **Conclusion on Best Performing Team:** Based on the number of categories led in the `comparison_df` and the significance of these metrics, the team leading in the most key areas, such as *[Placeholder: State your concluded best team]*, would be considered the top performer for the 2024-2025 Premier League season.

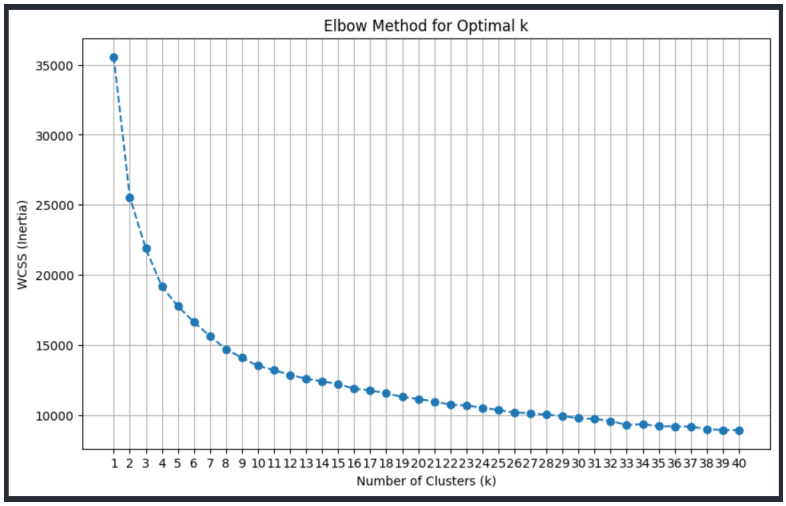
Chapter 4

Machine Learning

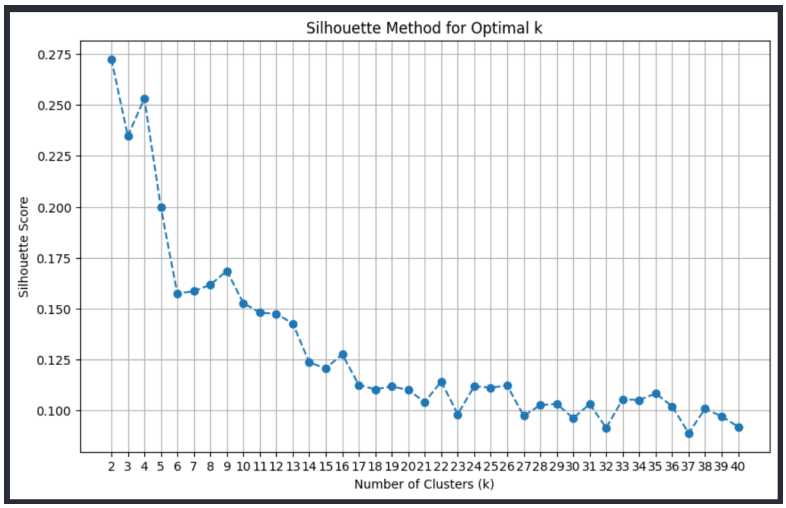
4.1 Exercise 3: Player Clustering with K-Means and PCA

This section covers Exercise III: classifying players using K-Means and visualizing with PCA, based on the script `EX3.ipynb`.

- **Data Preparation:** Player statistics from `results.csv` were loaded. Relevant statistical features (column 4 onwards) were selected, cleaned by converting to numeric and filling `NaN` values with 0, and then standardized using `StandardScaler` to have zero mean and unit variance (`X_standardized`).
- **K-Means Clustering Application:**
  1. The K-Means algorithm (`KMeans` from Scikit-learn) was applied to the standardized data (`X_standardized`).
  2. An initial visualization was performed using k=7 to demonstrate the clustering process with PCA.
  3. Cluster labels (`pred_label`) were assigned to each player based on the fitted model for the chosen k.
- **Determining the Optimal Number of Clusters (k):** To address the question "How many groups should the players be classified into?", two common methods were employed, evaluating k from 1 to 40 (Elbow) and 2 to 40 (Silhouette):
  - **Elbow Method:** The Within-Cluster Sum of Squares (WCSS or inertia) was calculated for k values from 1 to 40.



– **Silhouette Method:** The average silhouette score was calculated for k values from 2 to 40.



Justification for k:

– The Elbow method plot shows a clear bend around k=3 or k=4, after which the decrease in WCSS becomes much more gradual.

– The Silhouette method plot shows the highest score at k=2, with subsequent peaks being significantly lower. Scores generally decrease for k > 4, indicating poorer cluster quality.

– While k=2 provides the mathematically most distinct clusters, k=4 represents a reasonable balance indicated by the elbow method and might offer more granular insights. Therefore, **k=4** is selected as the optimal number as it commonly maps to GK, DEF, MID, FWD

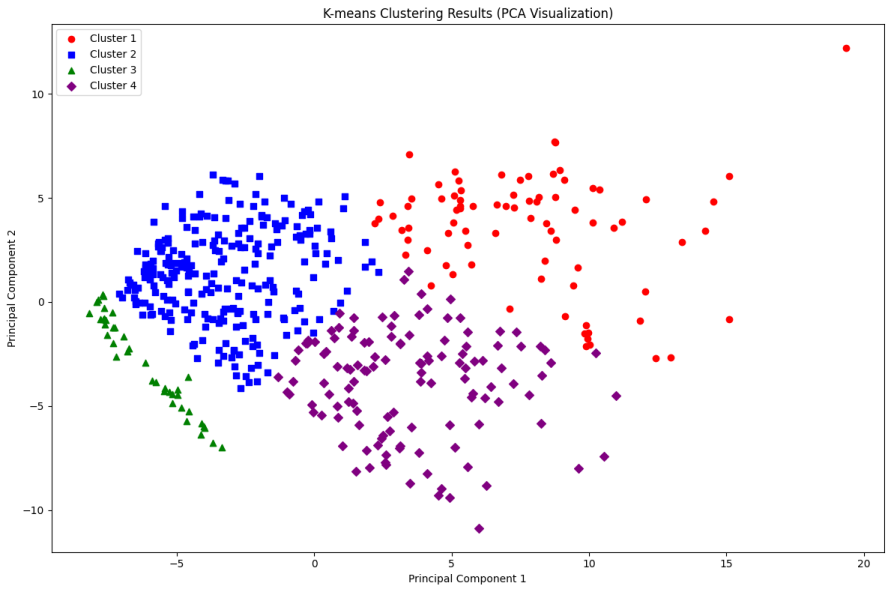
The K-Means algorithm is applied with this optimal k (k=4) for the final clustering and PCA visualization.

- **Principal Component Analysis (PCA) for Visualization:**

  1. To visualize the high-dimensional player data and the resulting clusters (for k=4) in 2D, PCA (`PCA` from Scikit-learn) was applied to the standardized data (`X_standardized`).
  2. The number of components was set to 2 (`n_components=2`).
  3. The standardized data was transformed into this new 2-dimensional space (`X_pca`).

- **2D Cluster Visualization:**

  1. A scatter plot was created using Matplotlib, plotting the first principal component (PC1) on the x-axis and the second principal component (PC2) on the y-axis.
  2. Each point represents a player.
  3. Points were colored according to the cluster label assigned by the K-Means algorithm (using the chosen optimal k=4).
  4. The plot visually represents player groupings based on their statistics after dimensionality reduction.



- **Comments on Results:**

The K-Means clustering with k=4 yielded four distinct player groups, which were then visualized using PCA.

- **Cluster Representation:** Based on the average statistics and player positions within each group:
  - **Cluster 1** mainly includes Goalkeepers, characterized by high goalkeeping stats and low outfield metrics.
  - **Cluster 2** represents Defensive players, showing strong defensive actions like tackles and interceptions.
  - **Cluster 3** likely consists of Midfielders, marked by strong passing and balanced contributions.
  - **Cluster 4** includes Attacking players, with high values in goals, assists, shots, and goal creation.
- **Conclusion:** The 2D PCA plot shows clear separation between clusters, especially for goalkeepers. Despite some overlap among defenders, midfielders, and attackers, their cluster centers are distinct. K-Means effectively reflects standard football positions and identifies meaningful player archetypes from Premier League data.

4.2 Exercise 4: Player Transfer Value Collection and Estimation

This section details the process undertaken for exercise IV of the assignment, which involves collecting player Estimated Transfer Values (ETV) and proposing a method for their estimation based on performance statistics.

- **ETV Data Collection and Matching:**
  1. **Scraping ETV:** Player names and their corresponding ETVs for the 2024-2025 Premier League season were scraped from `https://www.footballtransfers.com` using the Python script `EX4-p1-scrape_data.py`.
  2. **Filtering Performance Data:** The performance statistics dataset (`results.csv`) was filtered to include only players with more than 900 minutes of playing time (`Pltime_minutes > 900`).
  3. **Name Matching:** Since player names might have slight variations between the performance data source (fbref.com via `results.csv`) and the transfer value source (footballtransfers.com via `ETV_list.csv`), the script `EX4-p1-process-bertcos.py` was used for the matching process:

- Player names from both lists were standardized (e.g., converted to lowercase).
- The `sentence-transformers` library (specifically the `all-MiniLM-L6-v2` model, based on BERT) was employed to generate vector embeddings for the standardized names from both sources.
- Cosine similarity was calculated between the name embeddings of the filtered performance data players and the ETV list players.
- For each player in the filtered performance data, the best match from the ETV list was determined based on the highest cosine similarity score.

4. **Merged Dataset:** The matched ETV data was combined with filtered performance statistics to create the final dataset, linking player performance metrics with their estimated market value.

- **Proposed Method for Estimating Player Values:** A regression model was developed in `EX4-p2.ipynb` to predict numeric ETVs using player performance statistics and categorical data:

1. **Data Preprocessing:**
   - The target variable (ETV) was cleaned by parsing the string format (e.g., "€18.7M", "€500k") into a consistent numeric representation (in millions of Euros). Rows with unparseable ETVs were removed.
   - Features (X) were separated from the target variable (y).
   - Missing values in numeric features were imputed using the median strategy (`SimpleImputer(strategy='median')`).
   - Missing values in categorical features (like 'team', 'Req_Nation', 'Req_Position') were imputed using the most frequent strategy (`SimpleImputer(strategy='most_`
   - Numeric features were standardized using `StandardScaler` to have zero mean and unit variance.
   - Categorical features were converted into numerical format using `OneHotEncoder`.
   - A `ColumnTransformer` and `Pipeline` were used to apply these preprocessing steps consistently to the training and testing data splits.
2. **Feature Selection:**
   - Given the potentially large number of features after one-hot encoding, feature selection was employed to identify the most relevant predictors of ETV.
   - A `RandomForestRegressor` was trained on the preprocessed training data to obtain feature importance scores.

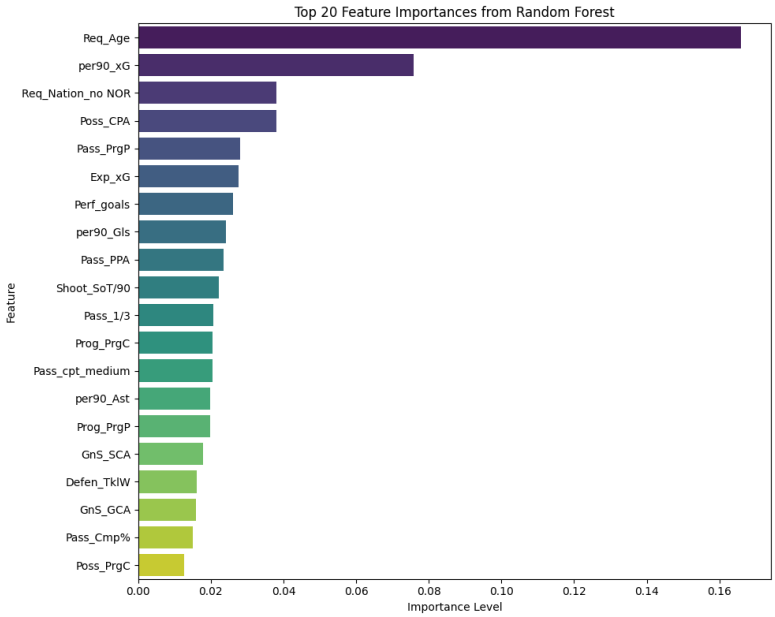


Figure 4.1: Top 20 Feature Importances from Random Forest.

- `SelectFromModel` was used with the trained RandomForest and a threshold (e.g., 'median' importance) to select a subset of the most influential features. Key features identified included 'Req_Age', 'per90_xG', 'Req_Nation_no NOR', 'Poss_CPA', 'Pass_PrgP', etc. (based on the script's output).

3. **Model Selection and Training:**
   - Three regression models were evaluated: `Ridge`, `RandomForestRegressor`, and `GradientBoostingRegressor`.
   - 5-fold cross-validation was performed on the feature-selected training data to estimate the generalization performance of each model, using Root Mean Squared Error (RMSE) and Mean Absolute Error (MAE) as evaluation metrics.
   - Based on the cross-validation results (specifically lower average RMSE), `GradientBoostingRegressor` was selected as the best performing model.
4. **Evaluation:**
   - The trained Gradient Boosting model was evaluated on the held-out test set (preprocessed and feature-selected).
   - Performance metrics calculated on the test set were: RMSE $\approx$ 15.22M €, MAE $\approx$ 11.33M €, and R-squared (R^2) $\approx$ 0.67.

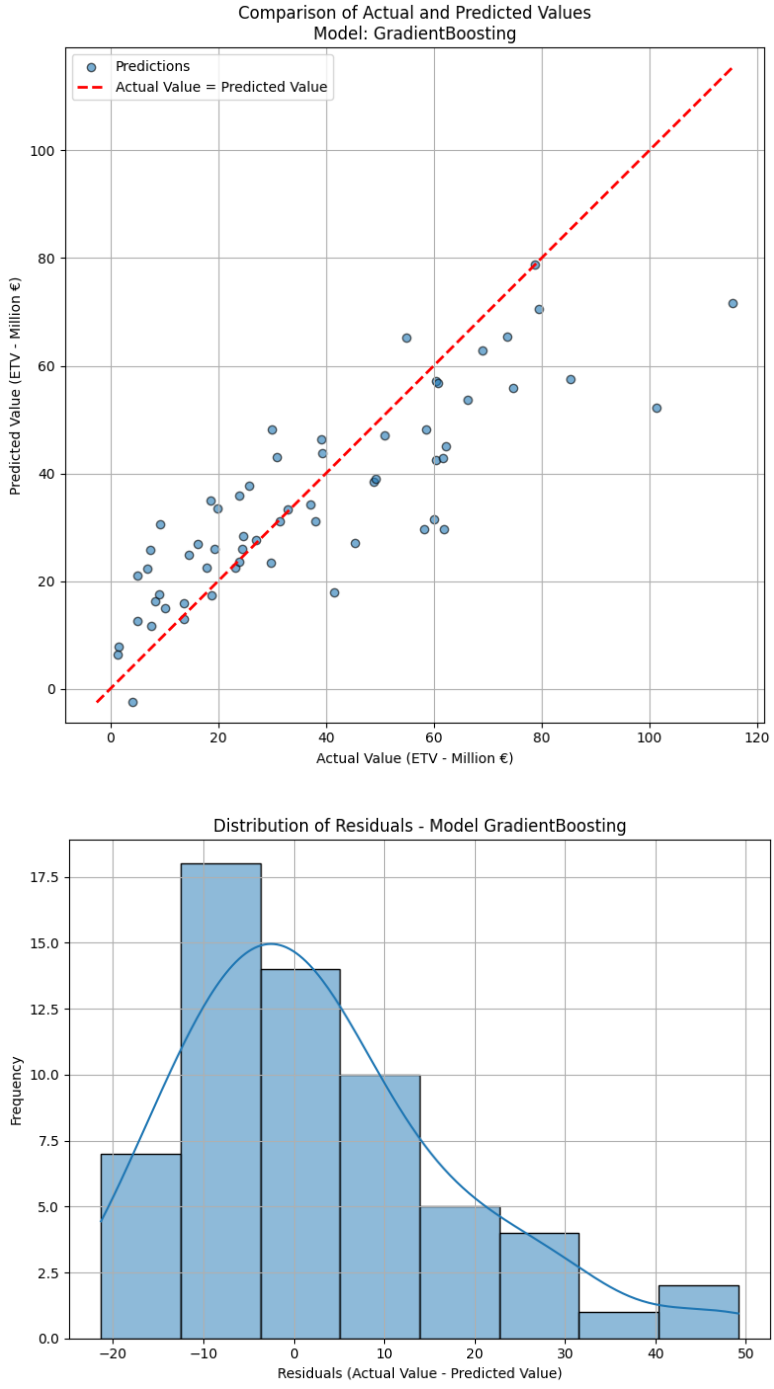


Figure 4.2: Gradient Boosting actual vs predicted values and residual distribution.

- The R^2 value indicates that the model, using the selected performance statistics and player information, explains approximately 67% of the variance in the Estimated Transfer Values in the test set. Visualizations comparing actual vs. predicted values and the distribution of residuals were also generated to assess model fit.

5. **Conclusion on Feature and Model Selection for ETV Estimation:** For ETV estimation, features were selected from preprocessed player data using RandomForestRegressor importances, prioritizing key attributes like age and per90_xG. In Model selection, `GradientBoostingRegressor` was chosen for its superior performance, achieving an R^2 of ≈ 0.67 on the test set, indicating a systematic approach to building a reasonably accurate prediction model.