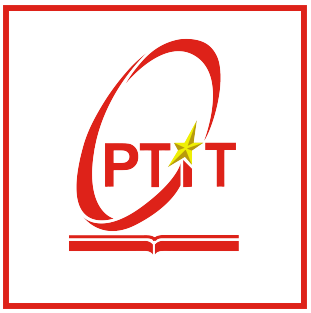
POSTS AND TELECOMMUNICATIONS
INSTITUE OF TECHNOLOGY
INFORMATION TECHNOLOGY




## PYTHON PROGRAMMING

---

### REPORT

# FOOTBALL DATA ANALYSIS

---

Name: NGUYEN PHAM TRUNG HIEU
Student ID: B23DCCE031
Class ID: B23CQCE04-B
Instructor: KIM NGOC BACH

Hanoi

# Table of contents

**1 Introduction** **2**
1.1 Abstract . . . 2
1.2 Goal . . . 2

**2 Background Knowledge** **2**
2.1 Python . . . 2
2.2 K-means . . . 3
2.3 PCA . . . 4
2.4 XgBoost . . . 5
2.5 Random Forest . . . 6

**3 Data Analysis** **7**
3.1 Collect and Process Data . . . 7
3.2 Statistical Analysis . . . 8
3.3 Data Visualization . . . 8
3.4 Team Performance Analysis . . . 9

**4 Machine Learning** **10**
4.1 Elbow Method and Silhouette Score . . . 10
4.2 Dimensionality Reduction with PCA . . . 11

**5 Estimating Player Transfer Values** **11**
5.1 Data Collection . . . 11
5.2 Data Cleaning . . . 12
5.3 Feature Selection and Importance Analysis . . . 13
5.4 Model Selection and Hyperparameter Tuning . . . 17

# Table of contents

# 1 Introduction

## 1.1 Abstract

This report explores player data from the English Premier League (EPL) on 2024 - 2025 seasons through a combination of data analysis and machine learning techniques. The goal is twofold: first, to extract and examine key information about players—such as names, positions, nationalities,etc. And second, to apply machine learning algorithms to gain deeper insights and make predictive assessments based on that data about the valuable of players.

## 1.2 Goal

By leveraging data science methods, this analysis aims to uncover patterns within player attributes and team compositions, and to build predictive models that can forecast outcomes transfer value of players. Techniques such as clustering, regression, and classification have been explored to better understand the Premier League landscape and support data-driven decision-making in football analytics.

# 2 Background Knowledge

## 2.1 Python

Python is a versatile, high-level programming language known for its clear syntax and readability. It has become an essential tool in the field of data science and analytics thanks to its rich libraries and large, supportive community. In this report, Python is used for the following purposes:

- **Data collection:** Python can be used to collect data from web sources like FBref.com and footballtransfers.com using libraries like **BeautifulSoup** and **Selenium**. These libraries allow extracting specific data from HTML pages, such as football player statistics and valuable of players .
- **Data processing: pandas** is a powerful Python library that provides data structures and functions for manipulating and cleaning data. It can be used to load the collected data into tabular data structures (DataFrames), handle missing values, convert data types, and perform other data transformations.
- **Data analysis:** Python offers a wide range of libraries for performing statistical and numerical data analysis. For example, the **numpy** library can be used to perform numerical computations on the data, while the scikit-learn library provides algorithms for machine learning, such as **K-means** clustering and **Principal Component Analysis (PCA).**
- **Data visualization: matplotlib** is a popular library for creating visual representations of data in Python. It can be used to generate various types of plots, such as histograms, scatter plots, and line charts, to help understand and communicate the findings of the analysis.

## 2.2 K-means

K-means clustering is a popular and widely used unsupervised machine learning algorithm. It's used to partition a dataset into K distinct, non-overlapping subgroups (clusters), where each data point belongs to the cluster with the nearest mean (centroid). To understand Kmeans algorithm how to work, Here's a breakdown of the K-means clustering process

- **Step 1:** Initialization
  - Randomly select the first centroid from the data points.
  - For each data point, calculate the squared distance to the nearest already-selected centroid.
  - Choose the next centroid with a probability proportional to the squared distance from the data points to their closest existing centroid. This means points that are far from existing centroids are more likely to be chosen as the next centroid.
  - Repeat steps 2 and 3 until K centroids have been selected.
- **Step 2:** Forming the Clusters Once the initial centroids are established, the algorithm proceeds to assign each data point to the cluster whose centroid is closest to it. This "closeness" is determined by a **distance metric**.
  - **Euclidean Distance:** This is the most commonly used distance metric in K-means. For two data points $p = (p_1, p_2, ..., p_n)$ in an n-dimensional space, the Euclidean distance is calculated as:
    $d(p,q) = \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + ... + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n}(p_i - q_i)^2}$
  - **Manhattan Distance (L1 Norm):** Another distance metric, calculated as the sum of the absolute differences between the coordinates:
    $d(p,q) = \sum_{i=1}^{n} |p_i - q_i|$
  - **Minkowski Distance:** A generalized distance metric where Euclidean and Manhattan distances are special cases (when the parameter is 2 and 1, respectively).
    $d(p,q) = (\sum_{i=1}^{n} |p_i - q_i|^r)^{1/r}$

  The choice of distance metric can influence the shape and characteristics of the resulting clusters. K-means is implicitly based on pairwise Euclidean distances as it aims to minimize the within-cluster sum of squared errors, which is related to these distances.
- **Step 3:** Adjusting the Centers
  - After all data points have been assigned to clusters, the algorithm recalculates the centroid for each cluster. The new centroid is the **mean** of all the data points belonging to that cluster.
  - If cluster $C_j$ contains $m_j$ data points $x_1, x_2, ..., x_{m_j}$ the new centroid $_j$ is calculated as:
    $\mu_j = \frac{1}{m_j} \sum_{i=1}^{m_j} x_i$
  - This step aims to find the point that minimizes the sum of squared distances of all points in the cluster to that point.
- **Step 4:** Iteration and Convergence: Refining the Clusters The assignment and update steps (2 and 3) are repeated iteratively. In each iteration:

- Data points are reassigned to the nearest centroid based on the updated centroid locations.
- The centroids are recalculated based on the new cluster memberships.

This iterative process continues until a **stopping criterion** is met. Common stopping criteria include:

- **No significant change in centroids:** The positions of the centroids stabilize, meaning they don't move much between iterations.
- **No change in cluster assignments:** Data points remain in the same clusters from one iteration to the next.
- **Reaching a maximum number of iterations:** The algorithm stops after a predefined number of iterations to prevent infinite loops, especially if convergence is slow.

Once the stopping criterion is satisfied, the algorithm has converged, and the resulting clusters are considered the final output.

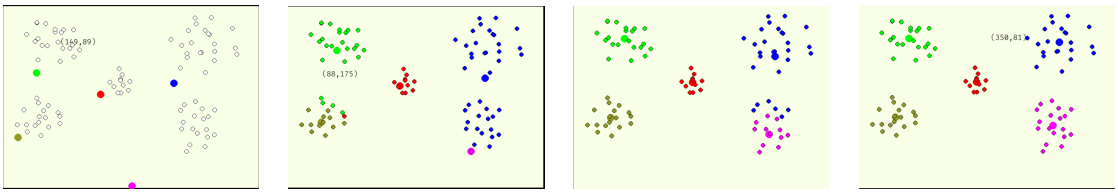


Figure 1: K-means algorithm with $k = 5$. Visualization available at this GitHub repository.

## 2.3 PCA

Principal Component Analysis (PCA) is a dimensionality reduction technique that transforms a high-dimensional dataset into a lower-dimensional one while retaining as much of the original information as possible. It identifies the principal components, which are the directions of maximum variance in the data, and projects the data onto these components.
Key Points:

- **Dimensionality Reduction:** PCA reduces the number of variables in a dataset, simplifying complex data and making it easier to analyze.
- **Information Retention:** PCA aims to preserve the most important information by focusing on the directions of maximum variance.
- **Uncorrelated Components:** The resulting principal components are uncorrelated, which can be beneficial for some machine learning algorithms.
- **Applications:** PCA is used in various fields, including data visualization, feature extraction, noise reduction, and data compression.

## 2.4 XgBoost

XGBoost is a highly efficient and optimized implementation of the Gradient Boosting framework. It's a popular and powerful machine learning algorithm used for both regression and classification tasks.

- **Key Features and Concepts:**
  - **Gradient Boosting:** XGBoost, like other gradient boosting methods, builds a predictive model by combining the predictions of multiple weaker models, typically decision trees. It does this in a stage-wise fashion, where each new tree is trained to correct the errors made by the previous trees.
  - **Regularization:** XGBoost incorporates both L1 and L2 regularization to prevent overfitting, which is a common problem in complex models. This helps improve the model's generalization performance.
  - **Speed and Performance:** XGBoost is designed to be highly efficient and fast. It employs various optimization techniques, such as parallel processing and cache optimization, to speed up the training process.
  - **Handling Missing Values:** XGBoost can handle missing values in the data without requiring imputation. It learns the best way to handle missing values during the training process.
  - **Tree Pruning:** XGBoost uses a technique called "tree pruning" to remove unnecessary branches from the trees, which further helps to prevent overfitting and improve generalization.
  - **Flexibility:** XGBoost can be used with various types of data and can be applied to a wide range of machine learning problems.
- **How XGBoost Works:**
  1. **Initialization:** The algorithm starts by initializing the model with a constant value.
  2. **Iteration:** For each iteration, the algorithm does the following:
     - Calculates the gradients and Hessians of the loss function with respect to the current predictions.
     - Fits a new decision tree to the negative gradients (residuals).
     - Adds the new tree to the model, weighting it by a learning rate to control the contribution of each tree.
  3. **Stopping Criteria:** The iteration continues until a stopping criterion is met, such as reaching a maximum number of iterations or when the model's performance on a validation set stops improving.
  4. **Prediction:** To make a prediction, XGBoost passes the input data through each tree in the ensemble and sums up the predictions of all the trees.
- **Advantages of XGBoost:**
  - High predictive accuracy
  - Fast training speed
  - Ability to handle large datasets

  - Effective handling of missing values
  - Regularization to prevent overfitting
  - Feature importance information

- **Applications:**

  XGBoost is widely used in various applications, including:

  - Classification problems (e.g., spam detection, image classification)
  - Regression problems (e.g., house price prediction, sales forecasting)
  - Ranking problems (e.g., search engine ranking)
  - And many more

## 2.5 Random Forest

- **Ensemble Learning and Decision Trees** At its core, Random Forest combines the power of multiple decision trees.

  - **Decision Trees**: These are tree-like structures where each node represents a question about the data, each branch represents a possible answer, and each leaf represents a prediction.
  - **Ensemble Learning**: Instead of relying on a single decision tree, Random Forest creates a "forest" of them. The final prediction is made by aggregating the predictions of all the individual trees.

- **How Random Forest Works** Here's a simplified breakdown:

  1. **Bootstrap Sampling**: Random Forest creates multiple subsets of the original data, sampling with replacement. This means some data points may appear in a subset multiple times, while others may not appear at all.
  2. **Random Feature Selection**: When building each decision tree, the algorithm randomly selects a subset of features to consider at each node split. This prevents the trees from becoming too similar and encourages diversity.
  3. **Tree Building**: Each decision tree in the "forest" is constructed using the bootstrapped data and the data and the random subset of features. The trees are typically grown to maximum depth without pruning.
  4. **Prediction**:
     - **Classification**: The Random Forest assigns a class to a new data point by taking a majority vote from all the trees.
     - **Regression**: The Random Forest predicts a numerical value by averaging the predictions of all the trees.

- **Key Advantages**

  - **High Accuracy**: By averaging or voting the results from many trees, Random Forest reduces the risk of overfitting and generally provides more accurate predictions than individual decision trees.

  - **Handles Complex Data**: Random Forest can handle both categorical and numerical data, and it can effectively model non-linear relationships.
  - **Feature Importance**: Random Forest provides a way to assess the importance of each feature in the dataset, which can be useful for feature selection and understanding the data.
  - **Robustness**: Random Forest is relatively robust to outliers and noisy data.
  - **Scalability**: Random Forest can handle large datasets with a reasonable amount of computational resources.

- **Key Parameters**

  - **n_estimators**: The number of decision trees in the ensemble. Increasing this number generally improves performance, but the gains diminish after a certain point, and training time increases.
  - **max_features**: The number of features to consider when looking for the best split at each node. Using fewer features reduces correlation between trees, potentially improving generalization and reducing overfitting.
  - **max_depth**: The maximum depth of each decision tree. Constraining the depth can limit model complexity and help mitigate overfitting, particularly in datasets with noise or irrelevant features.

- **Applications**

  Random Forest is used in a wide range of applications, including:

  - **Image Classification**: Identifying objects in images.
  - **Object Detection**: Locating objects within images.
  - **Medical Diagnosis**: Predicting the likelihood of a disease.
  - **Financial Modeling**: Predicting stock prices or credit risk.
  - **Natural Language Processing**: Sentiment analysis or text classification.

# 3 Data Analysis

## 3.1 Collect and Process Data

- The dataset was scraped from FBref.com, focusing on players with more than 90 minutes of playing time
- Tools used:
  - **Beautiful Soup**, **requests**, **Selenium** for scraping and parsing HTML tables.
  - Data saved into **results.csv**
  - Missing or inapplicable values were marked as "N/a".
  - Player records were sorted alphabetically by first name.
- Normalized column names.
- Converted statistical strings (like ”85.4”) to numerical values.

- Handled missing values by imputing median where appropriate or keeping them as "N/a" for analysis.
- Ensured numeric columns were float/int type for statistical computations.
- Eliminate the number of days the player's age only retains the number of age (like "24−123" to "24")

## 3.2 Statistical Analysis

- Top 3 and bottom 3 players for each metric were saved in **top_3.txt**


Figure 2: For example with matches_played

- Per-statistic aggregations: Median, Mean, Standard Deviation (per statistic, per team, and league-wide) saved in **results2.csv**.

| Team/Stat | Median of age | Mean of age | Std of age | Median of matches_pla | Mean of matches_p | Std of matches_pl |
|---|---|---|---|---|---|---|
| all | 26 | 26.4 | 4.22 | 21 | 19.6 | 9.03 |
| 2024-2025 West Ham United Stats | 28 | 28.12 | 5 | 20 | 19.76 | 7.88 |
| 2024-2025 Southampton Stats | 26 | 26.45 | 4.2 | 20 | 17.14 | 9.49 |
| 2024-2025 Everton Stats | 26 | 27.5 | 5.05 | 23 | 20.36 | 8.89 |
| 2024-2025 Manchester City Stats | 26 | 26.4 | 4.77 | 21 | 18.16 | 8.49 |
| 2024-2025 Leicester City Stats | 26 | 26.69 | 4.51 | 20 | 18.58 | 9.11 |
| 2024-2025 Bournemouth Stats | 25 | 25.87 | 3.73 | 22 | 19.78 | 8.89 |
| 2024-2025 Brighton & Hove Albion | 24 | 25.33 | 5.23 | 19 | 18.3 | 9.06 |
| 2024-2025 Crystal Palace Stats | 27 | 26.62 | 3.22 | 26 | 21.29 | 9.01 |
| 2024-2025 Fulham Stats | 28 | 28.27 | 3.4 | 23 | 21.68 | 8.57 |
| 2024-2025 Manchester United Stats | 25 | 25.6 | 5.01 | 20 | 18.92 | 10.1 |
| 2024-2025 Ipswich Town Stats | 26 | 26.47 | 3.09 | 17.5 | 16.63 | 8.4 |
| 2024-2025 Newcastle United Stats | 27 | 27.68 | 4.69 | 26 | 20.95 | 9.07 |
| 2024-2025 Liverpool Stats | 26 | 26.62 | 3.68 | 26 | 23 | 8.33 |
| 2024-2025 Aston Villa Stats | 27 | 26.68 | 4.05 | 19.5 | 17.71 | 9.46 |
| 2024-2025 Wolverhampton | 26 | 27.09 | 4.01 | 24 | 20.78 | 8.26 |
| 2024-2025 Nottingham Forest Stats | 26 | 26.57 | 3.75 | 28 | 23.33 | 9.26 |
| 2024-2025 Tottenham Hotspur Stats | 25 | 25.04 | 4.53 | 19 | 17.74 | 9.08 |
| 2024-2025 Chelsea Stats | 24 | 23.88 | 2.3 | 19 | 18.52 | 10.12 |
| 2024-2025 Brentford Stats | 24.5 | 25.8 | 3.79 | 25 | 22.35 | 10.13 |
| 2024-2025 Arsenal Stats | 26.5 | 25.95 | 3.82 | 21.5 | 21.27 | 7.77 |

Figure 3: For example with age and matched_played

## 3.3 Data Visualization

- Histograms generated for each attack statistic like xG, goals and SoT% and defense statistic like Tkl, TklW and Def 3rd using **matplotlib.pyplot.hist**.
- One histogram per:

  - League-wide statistic. The result saved in **histogram/top_3_attack and histogram/top_3_defense**
  - Team-specific statistic. The result saved in **histogram/all_players**

## 3.4 Team Performance Analysis

- **Methodology**:
  - To evaluate team performances during the 2024–2025 Premier League season, we analyzed players' statistical averages grouped by their respective teams.
  - The following steps were performed:
    * grouped players by their **team**.
    * Computed the average value of each statistic (e.g., goals, assists, expected goals, passes completed) for each team.
    * For each individual statistic, identified the team that achieved the highest average.
    * Counted the number of statistics in which each team ranked first.
    * Selected the team with the most top-ranking statistics as the overall best performer.
    * This process was automated using a custom Python function **analyze_data()**.
- **Findings**: The leadership table was saved into **data/leadership.csv**, listing which team led each statistic.


Figure 4: leadership

- **Conclusion**: Based on the number of statistics led and the strength of their performances in key areas such as scoring, passing efficiency, and chance creation, **Liverpool** is considered the best-performing team in the 2024–2025 Premier League season.

# 4 Machine Learning

## 4.1 Elbow Method and Silhouette Score

- **Feature preparation:** The players' statistics were normalized to ensure that each feature contributes equally to the distance calculations. Percentage-based statistics were converted to the range $[0, 1]$, and only numerical features were retained.

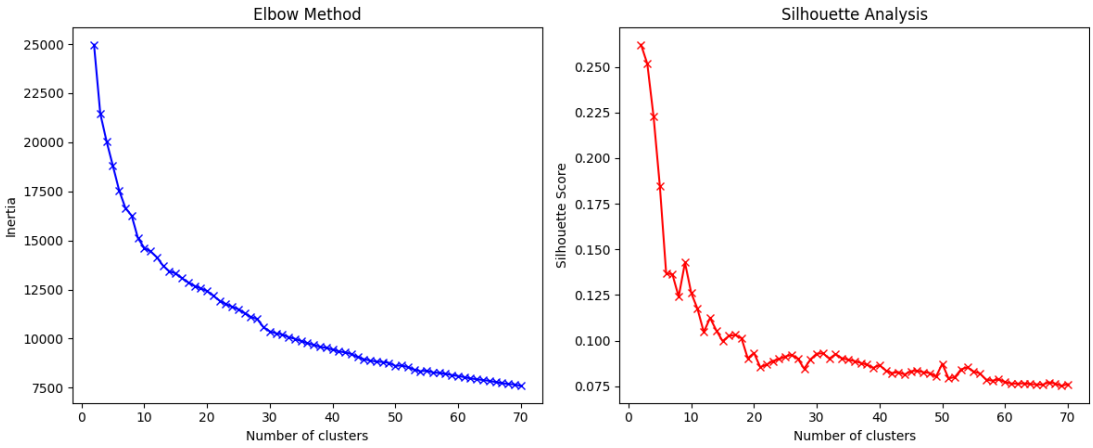



Figure 5: Visualization of the Elbow Method and Silhouette Score

- **Choosing the number of clusters (k):**
  - **Elbow Method:**
    - The "elbow" point, where the rate of decrease in the Within-Cluster Sum of Squares (WCSS) sharply slows down, suggests the optimal number of clusters.
    - Based on the Elbow plot, values of $k$ between 5 and 10 were considered appropriate.
  - **Silhouette Score Analysis:**
    - For each $k$, the Silhouette Score was computed to measure how well-separated the resulting clusters are.
    - The score ranges from $-1$ to 1, where a higher score indicates better cluster separation.
    - At $k = 7$, the Silhouette Score reached approximately 0.175, suggesting that 7 clusters provide meaningful segmentation of players.
- The dataset contains numerical performance statistics of players (e.g., goals, assists, passes, tackles) without explicit positional labels.
- The **Elbow Method** and **Silhouette Score** analysis indicate that $k = 7$ provides a reasonable balance between model complexity and data structure. Although the **Silhouette Score** ( 0.175) is relatively low, it is acceptable given the high-dimensional and noisy nature of player performance data.

- Selecting $k = 7$ allows us to group players into clusters that likely reflect different styles or performance archetypes — such as scoring ability (Gls), shooting accuracy (SoT%), and defensive strength (TklW) — rather than strictly defined playing positions.
- Thus, $k = 7$ captures meaningful variations among players based on their statistical contributions on the pitch.

## 4.2 Dimensionality Reduction with PCA

- To visualize the clustering results, we used **Principal Component Analysis (PCA)** to reduce the high-dimensional player statistics into two principal components.
- **Process:**
  - PCA was applied with $n\_components = 2$.
  - Players were then colored by their cluster labels obtained from K-Means clustering with $k = 7$.
- **Insights:**
  - The PCA plot showed that players within the same cluster tend to group closely together, visually validating the clustering results.
  - Some overlaps were observed between clusters, particularly between midfielders and attackers, which is expected due to role versatility in modern football.

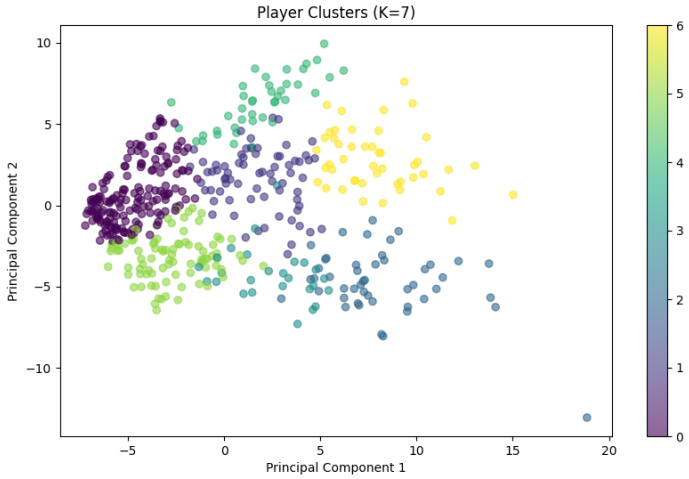



Figure 6: PCA visualization with $k = 7$

# 5 Estimating Player Transfer Values

## 5.1 Data Collection

- To estimate player transfer values, market data for Premier League players during the 2024–2025 season was collected from **FootballTransfers.com**. Only players with more

than 900 minutes of playing time were considered to ensure that only statistically significant contributors were included.

- Player names between FBref and FootballTransfers were not always consistent (e.g., *Son Heung-min* vs. *Heung-min Son*). To accurately merge records, we applied **cosine similarity** between vectorized name strings to identify the best-matching pairs across both datasets.
- Once matched, each player's transfer value from FootballTransfers was merged with their performance statistics from FBref saved in **results.csv**. This resulted in a clean dataset combining both quantitative performance features and corresponding market value labels, ready for modeling. The result of merge two data was saved in merge_df.csv
- **Method:**
  - Player names were vectorized using **models/gemini-embedding-exp-03-07** in **Gemini**. The embedding of two data from Fbref and Footballtransfers were saved in embed_df.csv and embed_df2.csv, respectively.
  - Cosine similarity was computed between each name from FBref and each name from FootballTransfers.
  - The closest match above a defined similarity threshold (e.g., 0.85) was accepted as the same player.
- **Outcome**: Using this method, we were able to:
  - Resolve inconsistent name formats between the two sources
  - Successfully merge player statistics and transfer values for most key players
  - Avoid manual mapping or excessive cleaning

## 5.2 Data Cleaning

Before estimating player transfer values, the merged dataset required careful preprocessing to ensure all features were clean, consistent, and suitable for machine learning models.

- **Handling Missing Values:** All occurrences of "N/a" in the dataset were replaced with numeric zero ('0'). This allowed for proper type conversion and avoided errors during numerical processing.
- **Converting Transfer Values:** Player market values from FootballTransfers were originally stored as strings with currency symbols and units (e.g., "€80M"). These were cleaned using regular expressions to remove non-numeric characters and converted to 'float' values (e.g., '80.0'), representing the value in millions.
- **Nationality Encoding:** The raw nationality column contained strings like "eng ENG". A custom parser extracted the actual country abbreviation (e.g., "ENG"), and this field was then encoded using **Label Encoding** into integer labels suitable for modeling.
- **Position Encoding:** Players often play multiple positions (e.g., "LW, ST"). The position column was split into a list of individual positions using regular expressions. Then, the positions were encoded using **MultiLabelBinarizer**, resulting in binary one-hot columns such as 'pos_LW', 'pos_CM', 'pos_CB', etc.

- **Final Output:** The cleaned and fully encoded dataset was saved as a new CSV file named `players_played_more_than_900m.csv`, containing only numeric values and ready for model training.

This cleaning process ensured that all features were well-structured and free from inconsistencies, enabling accurate estimation of player transfer values in subsequent stages.

## 5.3 Feature Selection and Importance Analysis

To accurately estimate player transfer values, we applied three complementary techniques to evaluate and compare the importance of features in the dataset:

- **Random Forest Feature Importance:** We trained a `RandomForestRegressor` on the training set. Feature importances were extracted and sorted to identify the top 15 most influential features. The result is visualized in Figure 7.
- **Recursive Feature Elimination (RFE):** Using a recursive process, features were eliminated based on model performance until only the top 15 were retained.
- **Univariate Feature Selection (SelectKBest):** We used `SelectKBest` with `f_regression` to select the 15 features most correlated with transfer value on a univariate basis.

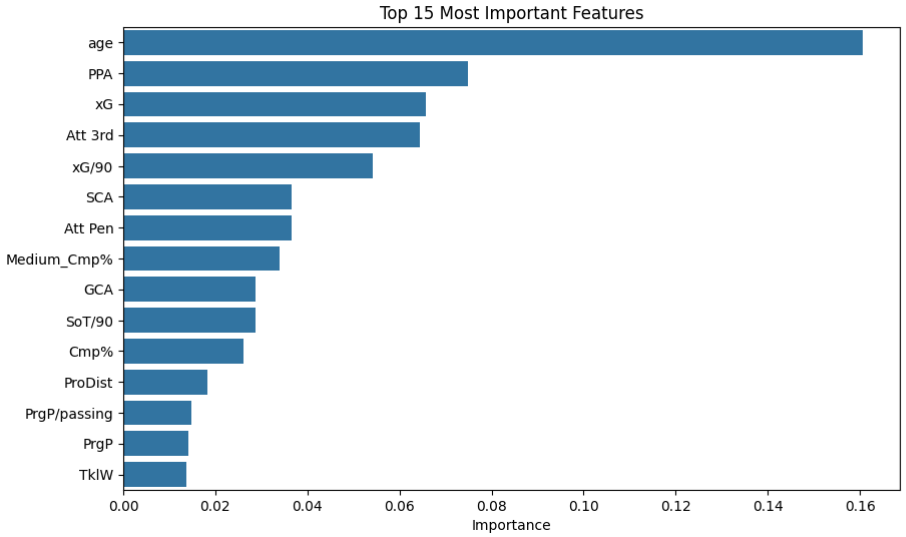



Figure 7: Top 15 most important features from Random Forest

**Correlation Analysis:** We also computed a Pearson correlation matrix for all numeric features. Only features with a correlation coefficient (absolute) greater than 0.3 with respect to the target (`Value`) were retained and visualized in a heatmap.

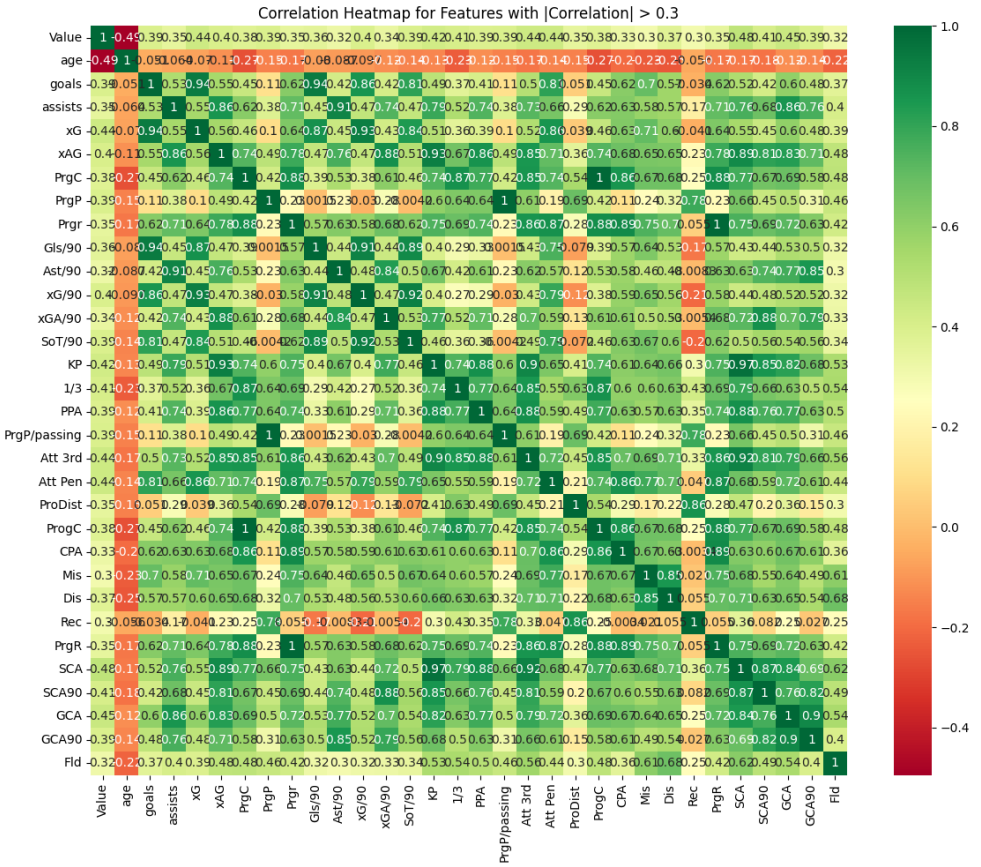



Figure 8: Correlation heatmap for features with |correlation| > 0.3

**Comparison of Methods:** We compared the features selected by all three methods and visualized their overlaps. Each selected feature was marked by method, and the count of methods selecting it was used to rank feature consensus.

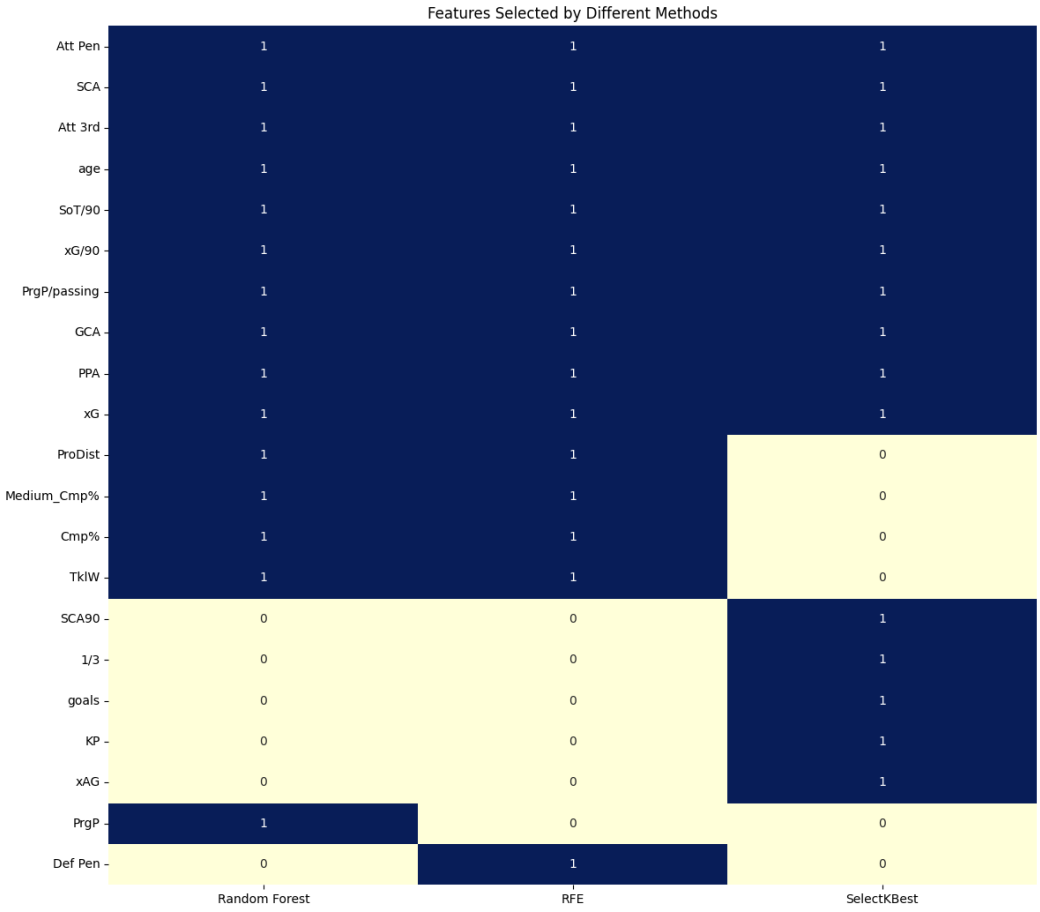



Figure 9: Comparison of selected features by Random Forest, RFE, and SelectKBest

**Common Features:** Features selected by all three methods were extracted and plotted again by their Random Forest importances. These features are considered the most reliable predictors of player value.

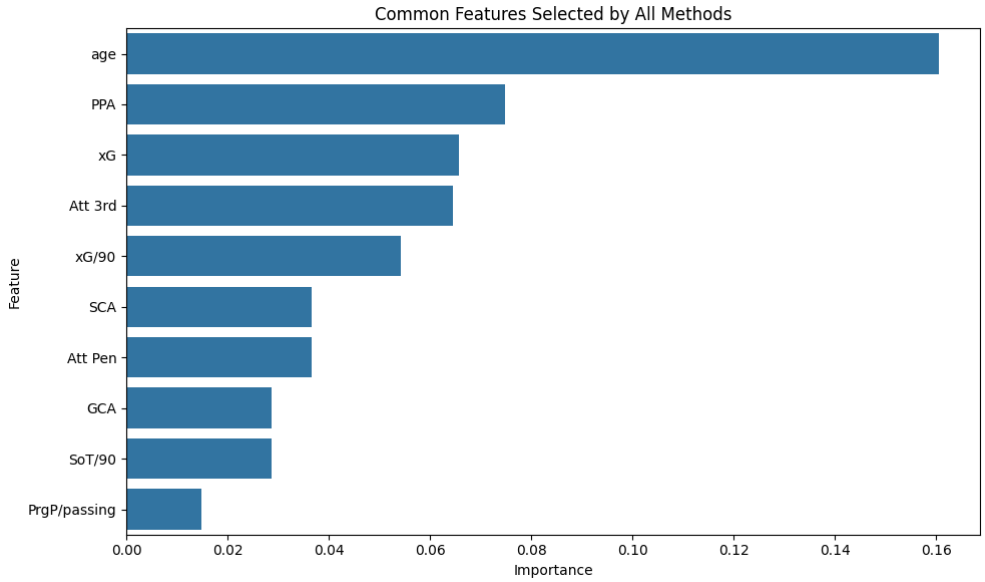



Figure 10: Common features selected by all methods and their importance

This multi-method feature selection approach improves the robustness and interpretability of the player value estimation model. **Conclusion:** > We select statistics have more than two important in three methods

- `age` – Player's age
- `xG` – Expected goals
- `Att Pen` – Passes into the penalty area
- `SCA` – Shot-creating actions
- `xG/90` – Expected goals per 90 minutes
- `PrgP/passing` – Progressive passes per total passes
- `PPA` – Passes into the penalty area
- `SoT/90` – Shots on target per 90 minutes
- `GCA` – Goal-creating actions
- `Att 3rd` – Passes into attacking third
- `Medium_Cmp%` – Medium pass completion
- `TklW` – Tackles won
- `ProDist` – Progressive passing distance
- `Cmp%` – Overall pass completion

## 5.4 Model Selection and Hyperparameter Tuning

To build a reliable regression model for estimating player transfer values, we evaluated and tuned three models:

- **Linear Regression (Baseline):** A simple model for comparison.
- **Random Forest Regressor:** A tree-based ensemble method that handles non-linear relationships and variable importance.
- **XGBoost Regressor:** A gradient boosting model optimized for high accuracy on structured datasets.

**Feature Set:** We used 14 high-importance features identified in the earlier feature selection stage, including:

- `age` – Player's age
- `xG` – Expected goals
- `Att Pen` – Passes into the penalty area
- `SCA` – Shot-creating actions
- `xG/90` – Expected goals per 90 minutes
- `PrgP/passing` – Progressive passes per total passes
- `PPA` – Passes into the penalty area
- `SoT/90` – Shots on target per 90 minutes
- `GCA` – Goal-creating actions
- `Att 3rd` – Passes into attacking third
- `Medium_Cmp%` – Medium pass completion
- `TklW` – Tackles won
- `ProDist` – Progressive passing distance
- `Cmp%` – Overall pass completion

—

**Tuning Strategy:**
To optimize each model's hyperparameters, we combined the strengths of two commonly used search techniques:

1. **Randomized Search (Exploration Phase):** RandomizedSearchCV was used to explore a wide hyperparameter space efficiently by randomly sampling parameter combinations. This step enables fast discovery of promising regions in the search space.
2. **Grid Search (Refinement Phase):** GridSearchCV was then applied to a narrower parameter grid centered around the best values found in the previous step. This fine-tuning process ensures more precise optimization.

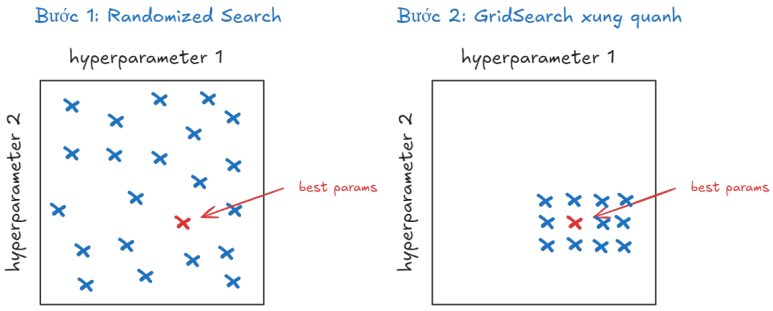



Figure 11: Visualize an idea for select models and hyperparameters

*In practice, combining these two methods allows us to leverage the speed and coverage of Randomized Search along with the precision of Grid Search.* All tuning used 5-fold cross-validation and was scored using the `neg_mean_absolute_error` metric.

—

**Results Summary:**

| Model | Best MAE (CV) | Total Tuning Time (s) |
|---|---|---|
| Linear Regression | 13.11 | 2.91 |
| Random Forest | **12.14** | **331.60** |
| XGBoost | 12.87 | 101.88 |

Table 1: Model performance after hyperparameter tuning

—

**Final Evaluation:**
The model with the lowest MAE on cross-validation was the **Random Forest Regressor**. Its performance on the held-out test set was:

- **Mean Absolute Error (MAE):** `13.06 million €`
- **Root Mean Squared Error (RMSE):** `16.91 million €`
- **$R^2$ Score:** `0.5489`

**Conclusion:** > The Random Forest model provided the most accurate and robust performance for estimating player market values. Despite its longer tuning time, its predictive power and resistance to overfitting make it well-suited to this regression task.

# References

[1] FBref. *Football Stats and History*. Retrieved from https://fbref.com

[2] FootballTransfers. *Player Market Values*. Retrieved from https://www.footballtransfers.com

[3] Pedregosa et al., 2011. *Scikit-learn: Machine Learning in Python*, JMLR 12, pp. 2825-2830.

[4] Nguyen, H. (2025). *Hyperparameter tuning with RandomizedSearchCV*. Data Science Dances. Retrieved from https://datasciencedances.com/blog/2025/03/hyperparameter-tuning-RandomizedSearchCV/

[5] Tiep Vu (Vũ Hữu Tiệp). *Machine Learning Cơ Bản*. Retrieved from https://machinelearningcoban.com/