POSTS AND TELECOMMUNICATIONS INSTITUTE OF TECHNOLOGY
FACULTY OF INFORMATION TECHNOLOGY I
DEPARTMENT OF PYTHON PROGRAMMING

PYTHON PROGRAMMING REPORT

Lecturers of the Department : KIM NGỌC BÁCH
Class : D23CQCE04 - B
Student ID : B23DCCE076
Full name : NGUYỄN HỮU NIÊM

Contents

1 Collecting Player Data from FBRef

1.1 Objective

To collect statistical data for players who played more than 90 minutes in the 2024–2025 Premier League season from `https://fbref.com`.

1.2 Methodology

1.2.1 Step 1: Identify Data Sources

Data is collected from the following statistical tables on: https://fbref.com/en/comps/9/Premier-League-Stats

- Standard Stats
- Goalkeeping
- Shooting
- Passing
- GCA/SCA (Goal and Shot Creation)
- Defense
- Possession
- Miscellaneous

1.2.2 Step 2: Set Up Selenium

The `webdriver_manager` library is used to automatically download the compatible version of ChromeDriver. The browser runs in headless mode to optimize performance and avoid displaying a window during data collection.

1.2.3 Step 3: Extract Data from Each Table

```
def scrape_table_with_selenium(url, table_id):
    try:
        print(f"Scraping {url}...")
        driver.get(url)
        time.sleep(5)  # Increased wait time

        # Wait for table to load
        table = driver.find_element(By.ID, table_id)
        html = table.get_attribute('outerHTML')

        # Use StringIO to avoid the FutureWarning
        df = pd.read_html(StringIO(html))[0]

        # Clean multi-index columns
        if isinstance(df.columns, pd.MultiIndex):
            df.columns = ['_'.join(col).strip() for col in df.columns.values]
```

```
        # Standardize column names
        df.columns = df.columns.str.replace(r'%', 'pct', regex=True)
        df.columns = df.columns.str.replace(r'[^a-zA-Z0-9_]', '_', regex=True)

        return df

    except Exception as e:
        print(f"Error scraping {url}: {str(e)}")
        return None
```

A function named `scrape_table_with_selenium()` was built to:

- Navigate to each URL containing the statistical table
- Locate the table using its HTML ID
- Convert the HTML into a DataFrame using `pandas.read_html()`
- Clean column names (remove special characters, handle multi-index headers)

1.2.4 Step 4: Clean and Merge Data

- Normalize player and team names
- Merge tables using `Player` and `Squad` columns with `pd.merge()`
- Filter players with total minutes played > 90
- Drop duplicate or unnecessary columns

1.2.5 Step 5: Select and Rename Columns

- Keep only the important statistical columns specified in the assignment
- Rename unclear column names into readable and standardized forms such as `Goals`, `Assists`, `xG`, `Touches`, `Tackles`, etc.

1.2.6 Step 6: Handle Missing Data and Format Values

- Replace missing values with `"N/a"`
- Normalize the nationality column to keep only country names instead of flag icons

1.2.7 Step 7: Export the Data

- Final cleaned data is exported to a file named `results.csv` for further analysis

1.3 Tools and Data Processing Techniques Used

The following tools and techniques were used during post-processing:

- **Handling Missing Values**
  Tool: `pandas.fillna()`
  Purpose: Replace missing values with `"N/a"` for consistency and to avoid errors in later steps.

- **Column Name Normalization**
  Tool: `pandas.DataFrame.columns.str.replace()` with regular expressions (regex)
  Purpose: Remove special characters such as `%`, `(`, `)`, make column names easier to use and understand.

- **Merging Data from Multiple Tables**
  Tool: `pandas.merge()`
  Purpose: Combine statistical tables using `Player` and `Squad` to create a unified dataset.

- **Filtering Data Based on Conditions**
  Tool: `pandas.query()` or boolean filtering
  Purpose: Keep only players with more than 90 minutes of playing time as required.

- **Normalizing Nationality Column**
  Tool: `.str.split(' ').str[-1]`
  Purpose: Remove flag emojis and retain only the country names (e.g., `Spain` → `Spain`).

- **Final Sorting and Cleaning**
  Tool: `sort_values()`, `reset_index()`, `drop_duplicates()`
  Purpose: Sort by player names, remove duplicates, and finalize the cleaned dataset for analysis.

1.4 Results

The final dataset was successfully exported to `results.csv`.

1.5 Remarks

During the process of collecting and processing player statistics from `FBRef.com`, I noted the following:

1. **Data Complexity:**
   The data on FBRef is split into multiple tables with different structures and column counts. Some tables have unclear column names (e.g., `Unnamed: ...`) or use multi-index headers, which adds complexity to processing.

2. **Technical Challenges:**
   Since FBRef displays data using JavaScript, libraries like `requests` or `BeautifulSoup` alone are insufficient. Therefore, `Selenium` was used to simulate a real browser. Loading tables can sometimes be slow or unstable, so `time.sleep()` was added to ensure the table is fully loaded before scraping. Column name collisions during merging were handled with suffixes and by dropping unnecessary columns (e.g., columns with `_drop` suffix).

3. **Data Processing Techniques:**
   Column normalization is essential for easier manipulation, especially for statistical analysis or plotting. Filtering players based on playing time helps eliminate noise from low-appearance players, ensuring meaningful analysis.

4. **Lessons Learned:**
   I learned how to combine multiple Python libraries (`Selenium + pandas`) to solve a real-world data problem. I also gained experience in cleaning and organizing messy data from various sources into a single dataset. Furthermore, I practiced dealing with missing data and building a robust data processing pipeline.

2 Statistical Analysis and Data Visualization

2.1 Objective

Conduct descriptive statistical analysis and data visualization to explore trends, distributions, and performance of players and teams in the 2024–2025 Premier League season. Calculate mean, median, and standard deviation for technical indicators. At the same time, identify the most outstanding team based on average performance.

2.2 Methodology

The analysis is carried out through the following specific steps:

- **Step 1: Find top 3 players for each statistic**
  Use a loop to iterate through each statistical column, find the top 3 and bottom 3 players with the highest and lowest values, along with team, nationality, and position information. The results are saved to the file `top_3.txt` to support individual performance evaluation.

- **Step 2: Descriptive statistics by team**
  Calculate the mean, median, and standard deviation for each indicator, applied to each team and the entire league. The results are saved in the file `results2.csv` for inter-team comparison.

- **Step 3: Visualize data using Histogram charts**
  Plot histograms for several attacking (Goals, Assists, Goals per Shot) and defensive (Tackles, Interceptions, Blocks) indicators. Charts are displayed for the entire league and for each team to assess distribution and playing characteristics.

- **Step 4: Identify leading team for each indicator**
  Compute the average value of each statistic for every team and identify the team with the highest value. Results are saved to the file `best_teams_per_statistic.csv`, allowing identification of top-performing teams in each aspect.

- **Step 5: Determine the best overall team based on positive metrics**
  Identify a group of positive indicators (e.g., goals, assists, accurate passes...), then count how many times each team leads these indicators. The team with the most positive statistics is considered the most effective overall.

2.3 Tools and data processing techniques used

- **Top 3 player analysis:** using `pandas` and `numpy` to identify the best and worst performing players for each statistic.
- **Descriptive statistics calculation:** using `pandas.groupby()`, `mean()`, `median()`, and `std()` to analyze average performance and variation by team.
- **Data visualization:** using `matplotlib.pyplot.hist()` to show the distribution of metrics through charts, enabling easy comparison among players and teams.
- **Best team analysis:** using logical filtering techniques in `pandas` to determine teams that excel in multiple key performance metrics.

2.4 Results

The analysis results are exported to the following files:

- `top_3.txt`: List of top 3 and bottom 3 players for each indicator.
- `results2.csv`: Descriptive statistics table (mean, median, std) by team and for the entire league.
- `best_teams_per_statistic.csv`: List of teams leading each technical indicator.
- `team_rankings_by_positive_stats.csv`: Ranking table of teams with the most positive indicators.

2.5 Remarks

The analysis reveals clear differences among teams in scoring ability, defense, and ball control. Histogram charts help detect skewed distributions, outliers, or clustering of statistics in a small group of players. Identifying teams that lead in multiple positive indicators provides an objective and comprehensive view of overall performance across clubs.

3 Player Clustering using the K-means Algorithm

3.1 Objective

This section aims to categorize players in the 2024–2025 Premier League into groups with similar technical characteristics and playing styles. Such clustering facilitates tactical analysis, player recruitment, and performance evaluation. It also enables a deeper understanding of player distribution and the emergence of specific archetypes such as defensive specialists, attackers, or ball controllers.

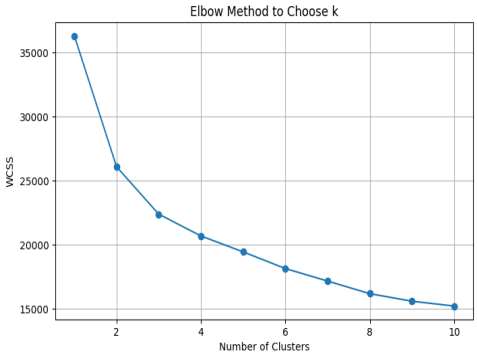
3.2 Methodology

3.2.1 Step 1: Data Preprocessing

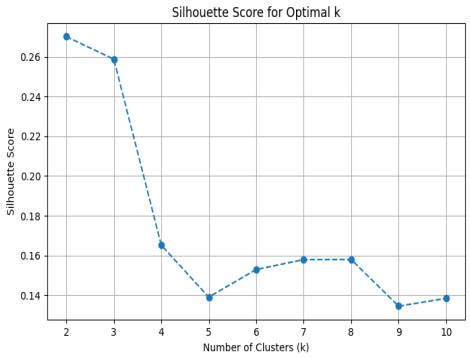
- Convert the `Age` column to float type for numerical processing.
- Remove descriptive columns including `Player`, `Nation`, `Squad`, and `Position`.
- Handle missing values using `SimpleImputer` with the mean strategy.
- Standardize all numerical features using `StandardScaler`.

3.2.2 Step 2: Determining the Optimal Number of Clusters

- **Elbow Method:** Plot the Within-Cluster Sum of Squares (WCSS) to identify the "elbow point."



- **Silhouette Score:** Evaluate the quality of cluster separation for k ranging from 2 to 10.



- Both methods indicate that $k = 8$ is the most appropriate number of clusters.

3.2.3 Step 3: Applying the K-means Algorithm

The `KMeans` algorithm is applied with $k = 8$ to segment the player dataset into 8 distinct clusters. Each player is assigned a corresponding cluster label, which is appended to the result dataset.

3.2.4 Step 4: Cluster Profiling

For each cluster, the mean of all technical indicators is computed to identify defining characteristics. Some example cluster archetypes include:

- High goal-scoring cluster
- Defensive tackling cluster
- Ball control cluster

3.2.5 Step 5: Visualizing Clustering Results with PCA

- Apply Principal Component Analysis (PCA) to reduce data dimensionality to 2D.
- Plot a scatter diagram with color-coded clusters.
- The visualization illustrates a clear separation between clusters in two-dimensional space.

3.3 Tools and Libraries

- **scikit-learn:** Modules used include `KMeans`, `PCA`, `StandardScaler`, `SimpleImputer`, and `silhouette_score`.
- **pandas:** For data manipulation and preparation.
- **matplotlib** and **seaborn:** For visual representation of clustering results.

3.4 Results

Clustering with $k = 8$ partitions players into eight distinct groups with unique technical profiles. The clusters reflect clearly different playing styles such as:

- High goal scorers
- Creative playmakers
- Defensive disruptors
- Possession-oriented players

The PCA-based scatter plot confirms a well-defined separation of clusters in the 2D projection space.

3.5 Discussion

Clustering reveals common player archetypes within the league and supports practical applications such as transfer targeting, tactical squad building, and evaluating positional balance. However, the reliability of clustering results depends heavily on the selected features, and further validation using real-world match data is recommended.

4 Player Value Prediction

4.1 Objective

The objective of this section is to combine players' technical statistics with their estimated transfer values (ETV) to create a clean, complete dataset suitable for training machine learning models. This integration requires accurate player name matching, numerical data cleaning, and appropriate feature preparation for modeling.

4.2 Methodology

4.2.1 Stage 1: Collecting and Normalizing ETV Data

- The `Selenium` library was used to automatically access 22 web pages from `footballtransfers.com`, each containing a list of Premier League players and their corresponding transfer values.
- HTML tables were located using Selenium methods to extract player names and ETVs.
- Player names were normalized by removing special characters, converting to lowercase, handling specific name cases (e.g., "Son Heung-min" to "Heung Min Son"), and sorting name tokens for accurate matching.
- For players not found on the website or whose names could not be matched due to format differences, ETV values were manually assigned using a predefined mapping dictionary.

4.2.2 Stage 2: Merging Technical Statistics with ETV Data

- Technical performance data was read from the `results.csv` file, containing detailed statistics such as goals, assists, tackles, etc.
- Data was merged using `pandas.merge()` based on normalized player names.
- An `inner` join strategy was applied to retain only players with both technical and ETV data.
- The merged dataset was saved as `results_with_etv.csv`.

4.2.3 Stage 3: Cleaning and Preparing Training Data

- The `ETV` column (e.g., "66.5M") was converted to float by removing special characters.

```
df['ETV'] = df['ETV'].str.replace(r'[€M]', '', regex=True).astype(float)
df['ETV'] = np.clip(df['ETV'], df['ETV'].quantile(0.05), df['ETV'].quantile(0.95))
df['ETV'] = np.log1p(df['ETV'])
```

- Clipping (5% and 95%) and log transformation (`log1p`) were applied to reduce the impact of outliers.
- The `Age` column was normalized from "years-days" format to a single float.

```
def convert_age(age):
    try:
        if isinstance(age, (int, float)):
            return float(age)
        if '-' in str(age):
            y, d = map(int, str(age).split('-'))
            return y + d / 365
        return float(age)
    except:
        return np.nan

df['Age'] = df['Age'].apply(convert_age)
df['Age'] = df['Age'].fillna(df['Age'].median()).round(1)
```

- Remaining numerical features were cast to float and missing values were filled with zero.
- Irrelevant columns such as `Player`, `Nation`, `Squad`, and `Position` were removed.

```
def engineer_features(df):
    X = df.drop(columns=['ETV', 'Player', 'Nation', 'Squad'])
    y = df['ETV']
    if 'Position' in X.columns:
        X = X.drop(columns=['Position'])
    return X, y

X, y = engineer_features(df)
```

- The dataset was split into training (80%) and testing (20%) sets using `train_test_split` from `scikit-learn`.

4.2.4 Modeling and Evaluation

Several machine learning models were trained to predict the log-transformed ETV values, including:

- **Linear Regression**
- **Random Forest Regressor**
- **Gradient Boosting Regressor**

Among them, Gradient Boosting Regressor achieved the best results on the test set:

- R^2: 0.6958
- RMSE: 0.4314
- MAE: 0.3451
- Spearman Rank Correlation: 0.7955

Feature importance was also extracted to understand the most influential attributes in player valuation.

4.3 Tools and Libraries Used

- **Selenium**: For web scraping ETV data.
- **pandas**: For data processing, cleaning, and merging.
- **numpy**: For numerical transformations.
- **fuzzywuzzy**: For accurate player name matching.
- **regex**: For string formatting and cleaning.
- **scikit-learn**: For data preprocessing, splitting, and modeling support.

4.4 Results

The `results_with_etv.csv` dataset was successfully created, containing over 200 Premier League players with full information on nationality, age, club, position, and more than 50 technical performance indicators. ETV values were integrated and the dataset is ready for machine learning tasks.

4.5 Discussion

The integration phase was crucial since improper name normalization could result in mismatches or missing data. The log transformation of ETV values helped stabilize the data distribution and reduce the risk of model overfitting.

5 Conclusions and Development Directions

5.1 Summary of the Implementation Process

The project successfully developed a system for analyzing and estimating the value of Premier League players based on technical performance data and estimated transfer value (ETV). The implementation process included the following main stages:

- Collected player technical statistics from FBRef.com using Selenium.
- Cleaned and integrated multiple statistical tables into a single aggregated dataset.
- Collected and standardized ETV data from footballtransfers.com.
- Merged the technical and ETV data to form a complete training dataset.
- Built machine learning models for player value prediction using algorithms such as Linear Regression, Random Forest, and Gradient Boosting.

5.2 Model Efficiency and Achieved Results

Model	R^2	RMSE	MAE	Spearman
Linear Regression	0.594	0.4948	0.4137	0.7846
Random Forest	0.6661	0.4520	0.3512	0.7497
Gradient Boosting	**0.6958**	**0.4314**	**0.3451**	**0.7955**

Among the models implemented, the Gradient Boosting Regressor yielded the best performance on the test set with the following metrics:

- $R^2 = 0.6958$
- RMSE = 0.4314
- MAE = 0.3451
- Spearman correlation = 0.7955

These results indicate that the model can predict player values with relatively high accuracy, especially in ranking players by their market value.

5.3 Highlights in the Implementation Process

- Successfully integrated data from multiple heterogeneous sources.
- Standardized player names to handle formatting inconsistencies during data merging.
- Applied preprocessing techniques such as outlier handling, log transformation, and normalization.
- Deployed a complete machine learning pipeline with automatic feature selection using Recursive Feature Elimination (RFE) to improve predictive performance.

5.4 Limitations

- ETV data is time-sensitive but was collected at a single point in time.
- Some players were manually assigned ETV values due to missing data, which may cause minor inconsistencies.
- Certain potentially useful features such as tactical roles, recent performance trends, or team-related parameters were not deeply exploited.

5.5 Future Development Directions

- Expand the model across multiple seasons to improve generalizability.
- Add more contextual features from the transfer market, such as player age, rarity of nationality, or contract duration.
- Deploy a web application or interactive dashboard that allows users to input player data and receive real-time ETV predictions.
- Experiment with advanced models such as XGBoost, LightGBM, or deep learning approaches.

Appendix

A. Data Sources

- Player statistics: `https://fbref.com/en/comps/9/Premier-League-Stats`
- Estimated Transfer Values (ETV): `https://www.footballtransfers.com`

B. Key Technical Features Used

- Offensive: Goals, Assists, Expected Goals (xG), Shots on Target, xG per 90 minutes
- Defensive: Tackles, Interceptions, Blocks, Clearances
- Possession: Touches, Carries, Progressive Carries, Passing Accuracy
- Others: Minutes Played, Age (normalized), Positions (encoded)

C. Model Performance Summary

Model	R^2	RMSE	MAE	Spearman
Linear Regression	0.5940	0.4984	0.4137	0.7846
Random Forest	0.6661	0.4520	0.3512	0.7497
Gradient Boosting	**0.6958**	**0.4314**	**0.3451**	**0.7955**

D. Tools and Libraries

- **Selenium** – Web scraping from dynamic websites
- **pandas, numpy** – Data manipulation and numerical computation
- **scikit-learn** – Machine learning models, preprocessing, evaluation
- **fuzzywuzzy** – Fuzzy string matching for player name alignment
- **matplotlib, seaborn** – Data visualization

References

- FBRef. Premier League Player Stats. https://fbref.com/en/comps/9/Premier-League-Stats
- FootballTransfers. Estimated Transfer Values. https://www.footballtransfers.com
- scikit-learn: Machine Learning in Python. https://scikit-learn.org/stable/
- pandas Documentation. https://pandas.pydata.org/
- Selenium Python Docs. https://selenium-python.readthedocs.io/
- matplotlib Documentation. https://matplotlib.org/
- seaborn Documentation. https://seaborn.pydata.org/