Report on the Python Script for Premier League Player Data Analysis

1. Introduction

This report details a Python script designed to scrape football statistics for the 2024-2025 Premier League season from the FBref.com website, process this data, identify top and bottom performing players for various metrics, perform player clustering using K-means, and visualize the clusters using Principal Component Analysis (PCA).

2. Libraries Used

The script utilizes several key Python libraries, including:

- **pandas:** For data manipulation and analysis.
- **re:** For regular expressions, used for cleaning player names.
- **numpy:** For numerical operations.
- **matplotlib.pyplot:** For plotting graphs.
- **selenium:** For web scraping dynamic content.
- **beautifulsoup:** For parsing HTML content.
- **sklearn.cluster.KMeans:** For performing K-means clustering.
- **sklearn.preprocessing.StandardScaler:** For standardizing data.
- **sklearn.decomposition.PCA:** For dimensionality reduction.
- **sklearn.metrics.silhouette_score:** For evaluating clustering performance.
- **warnings:** To handle warnings.
- **time:** For introducing delays.
- **os:** (Although not explicitly used in the provided snippet) is typically used for interacting with the operating system.

3. Key Functions (Problems 4.1)

The script defines several important functions:

3.1. clean_player_name(name)

- **Purpose:** This function takes a player's name as input and cleans it by removing any non-alphanumeric characters (except spaces) and stripping leading/trailing whitespace.
- **Input:** name (string): The player's name to be cleaned.
- **Output:** (string): The cleaned player name. If the input is not a string, it returns an empty string.
- **Details:** It uses the re.sub() function with the pattern r'[^\w\s]' to replace any character that is not a word character (\w) or whitespace (\s) with an empty string. The .strip() method removes any leading or trailing whitespace.

3.2. extract_first_name(name)

- **Purpose:** This function extracts the first name of a player from their full name.
- **Input:** name (string): The full name of the player.
- **Output:** (string): The first name of the player. If the name is empty, it returns the original empty name.
- **Details:** It splits the input name string into a list of words using spaces as delimiters. If the list of words is not empty, it returns the first element (the first name).

3.3. scrape_table(url, table_id, retries=7)

- **Purpose:** This function scrapes a specific HTML table from a given URL using Selenium and BeautifulSoup. It includes error handling and retry mechanisms.
- **Inputs:**
    - url (string): The URL of the webpage containing the table.

    - table_id (string): The ID attribute of the HTML table to be scraped.
    - retries (integer, default=7): The number of times to retry scraping in case of errors.
- **Output:** (pandas.DataFrame or None): A Pandas DataFrame containing the scraped table data, or None if scraping fails after all retries.
- **Details:**
    - It initializes a headless Chrome browser using Selenium to handle potentially dynamic content on the webpage.
    - It navigates to the specified url and waits (up to 15 seconds) for the element with the given table_id to be present.
    - It retrieves the outer HTML of the table element.
    - It uses BeautifulSoup to parse the HTML.
    - It iterates through the table rows (identified by data-row attributes) and extracts the text content of each cell (td or th). Missing cell values are replaced with "N/a".
    - It extracts the table headers from the thead row or the first row if thead is not found. It handles potential inconsistencies in the number of header and data columns by padding or truncating the header list. It also ensures unique column names by appending a counter to duplicate names.
    - It creates a Pandas DataFrame from the extracted data and headers.
    - It removes rows where the 'Player' column is NaN, 'Player', or empty.
    - It handles duplicate player entries by grouping by 'Player' and taking the first occurrence, while also printing details of the duplicates found.

- It includes a retry loop with a delay to handle potential network issues or temporary website problems. If scraping still fails after the specified number of retries, it returns None.

3.4. compute_top_bottom_players(df)

- **Purpose:** This function computes the top 3 and bottom 3 players for each numeric statistic in the DataFrame, including players with zero scores, and saves the results to a file named top_3.txt.
- **Input:** df (pandas.DataFrame): DataFrame containing the player data.
- **Output:** No explicit return value. The results are saved to the top_3.txt file.
- **Details:**
  - It identifies the numeric columns in the DataFrame (excluding 'Player', 'Team', 'Position', 'Nation', 'First_Name').
  - For each numeric column, it sorts the DataFrame by that column in descending order (to find the top 3 players) and ascending order (to find the bottom 3 players).
  - It handles goalkeeper-specific columns ('GA90', 'Save%', 'CS%', 'PK Save%') separately by filtering the DataFrame to include only players whose position contains 'GK'. Similarly, for other columns, it excludes goalkeepers.
  - It drops rows with NaN values in the current column before sorting.
  - It writes the name, team, and value of the top 3 and bottom 3 players for each statistic to the top_3.txt file.

3.5. perform_clustering_and_pca(df)

- **Purpose:** This function applies K-means clustering, determines the optimal number of clusters, applies PCA to reduce the data dimensionality to 2 components, and plots the 2D clusters.

- **Input:** df (pandas.DataFrame): DataFrame containing the player data.
- **Output:** No explicit return value. The elbow plot, silhouette score plot, and 2D cluster plot are saved as PNG files (elbow_plot.png, silhouette_plot.png, clusters_2d.png), and the clustering results are saved to a CSV file (clusters.csv).
- **Details:**
    - It identifies the numeric columns in the DataFrame to be used for clustering.
    - It prepares the data for clustering by selecting the numeric columns and dropping columns with all NaN values.
    - It imputes any remaining NaN values with the mean of the respective column.
    - It standardizes the data using StandardScaler.
    - It determines the optimal number of clusters (k) by using the Elbow Method and Silhouette Score for a range of cluster numbers (2 to 10). Plots for both methods are saved.
    - It applies the K-means algorithm with the optimal number of clusters.
    - It adds the cluster labels to the original DataFrame.
    - It saves the clustering results (Player Name, Team, Position, Cluster) to a clusters.csv file.
    - It analyzes the characteristics of each cluster by printing the number of players, average values for some key statistics (Goals, Assists, Tackles, Recoveries), position distribution, and a few sample players within each cluster.
    - It applies PCA to reduce the standardized data to 2 principal components.
    - It plots the 2D clusters, with each point representing a player and the color indicating the cluster they belong to. A few

representative players (e.g., the top scorer in each cluster) are labeled on the plot. The plot is saved as clusters_2d.png.

3.6. main()

- **Purpose:** This function is the main function that executes the entire process.
- **Input:** No explicit input.
- **Output:** No explicit return value. It orchestrates the data scraping, processing, analysis, and saving of results.
- **Details:**
    - It defines a list of URLs and corresponding table IDs to scrape data from FBref.com.
    - It validates the format of the tables list.
    - It scrapes the initial standard statistics table to get basic player information.
    - It cleans player names and filters out players who have played less than 90 minutes.
    - It creates an initial result DataFrame with player names and first names.
    - It maps the required statistic display names to their corresponding FBref column names and adds this data to the result DataFrame.
    - It iterates through the remaining URLs and table IDs, scrapes the data, and merges it into the result DataFrame based on player names.
    - It keeps track of any unmapped columns.
    - It saves the final result DataFrame to a results.csv file.
    - It calls the compute_top_bottom_players() and perform_clustering_and_pca() functions to perform further analysis.

- Finally, it prints a summary of the number of 'N/a' values in each column of the results.csv file.

3. Key Functions (Problems 4.2)

Key functions used:

parse_value()

python

Copy

Download

```
def parse_value(value):
    if pd.isna(value) or value == "N/A":
        return np.nan
    if isinstance(value, (int, float)):
        return float(value)
    if isinstance(value, str):
        value = value.replace("€", "").replace("£", "").strip()
        if "M" in value:
            return float(value.replace("M", "")) * 1e6
        elif "K" in value:
            return float(value.replace("K", "")) * 1e3
        return float(value)
    return np.nan
```

Purpose:

Converts string values (e.g., "€50M" → 50,000,000)
Handles missing values

Label Encoding

python

Copy

Download

```
label_encoders = {}for col in ["Nation", "Team", "Position"]:
    encoder = LabelEncoder()
    data[col] = encoder.fit_transform(data[col])
```

Purpose:

Converts categorical data (Team, Position) to numerical values

Model Training

Train-Test Split

python

Copy

Download

```
X_train, X_test, y_train, y_test = train_test_split(
    features, target, test_size=0.2, random_state=42)
```

Purpose:

Splits data into 80% training and 20% testing sets

Random Forest Regressor

python

Copy

Download

```
rf_model = RandomForestRegressor(
    n_estimators=100,
    max_depth=15,
    min_samples_split=5,
    min_samples_leaf=3,
    random_state=42)
```

rf_model.fit(X_train, y_train)

Parameters:

`n_estimators=100`: 100 decision trees
`max_depth=15`: Limits tree depth to prevent overfitting

Evaluation

Metrics Calculation

python

Copy

Download

```
mse = mean_squared_error(y_test, predictions)
r2 = r2_score(y_test, predictions)
```

Output:

MSE: 2.34e12 (lower is better)
R^2 Score: 0.87 (closer to 1 is better)

Visualization

Actual vs Predicted Values Plot

Shows model accuracy (ideal: points on red dashed line)
Saved as `actual_vs_predicted.png`

Feature Importance

Ranks features by impact on predictions
Saved as `feature_importance.png`

4. Conclusion

This Python script provides a comprehensive pipeline for collecting, processing, and analyzing statistical data of Premier League players. By scraping data from FBref.com, cleaning and merging it, the script enables the identification of top and underperforming players across various metrics. Furthermore, the

application of K-means clustering and PCA facilitates the discovery of groups of similar players based on their statistical profiles and visualizes these relationships intuitively. The results are stored in text, CSV, and PNG image files for further analysis and reference.