Football Predictor Project

*[Python, matplotlib, seaborn, pandas]*

**Table Of content:**

* [Aim and Objective](#AO)
* [Dataset](#dataset)
* [Data Cleaning & Preparation.](#dc_p)
* [Exploratory Data Analysis](#eda)
* [Our base model](#basemodel)
* [Steps to improve our model (Rolling Average)](#improve)
* [Model selection & training.](#modelsel)
* [Evaluation](#evaluation)
* [Further work and Improvements](#fimpro)

**Aim and Objective**

The aim of this project was to build a model that could accurately predict the outcome of future premier league football matches. Success was judged using the following objectives:

* Achieve a test accuracy of greater than 60% in the base model, with a stretch target between 60%-70%.
* Also, I wanted to demonstrate machine learning skills learned in my coursework. Topics such as data cleaning, data manipulation, understanding correlation, modeling, model improvement strategies, and evaluation.

**Dataset**

Note: the dataset for this project was initially gathered by scraping the website “<https://fbref.com/en/comps/9/Premier-League-Stats>” using the BeautifulSoup library but eventually the scraping code is outdated as the website got updated. The code will be launched soon and will be added to the repository.

For the project, the data was captured from the website for individual teams and seasons. The attached file here has the data for teams' scores and fixtures and shooting variables from 2019 to 2023. All the files collected were combined in a single file using a python code available here. But if you need the final combined file, you can find it here.

The dataset consists of the following attributes which are gathered from both the scores and fixtures(matches) and shooting data files.

1. Date - Datetime of the match
2. Time - Time listed in local to the match
3. Comp - Competition name
4. Round - Round or Phase of Competition
5. Day - Day of the week
6. Venue - Matched played at home or Away.
7. GF - Goals For
8. GA - Goals Against
9. Opponent - The opponent team
10. Poss. - Possession - calculated as the number of passes attempted.
11. Formation - No. of players in each row from defenders to forwards, not including the goalkeeper.
12. Referee - the referee of the match
13. team - stats respective to the team
14. Gls - Goals scored.
15. Sh - Shots total
16. SoT- Shots on target, does not include penalty kicks
17. SoT percentage - shots on the target percentage value
18. G/Sh - Goals per shot
19. G/SoT - Goals per shot on target
20. Dist - Average Shot distance
21. FK - Shot from free kicks.
22. PK - Penalty kicks made.
23. PKatt - Penalty kicks attempted.

**Data Cleaning & Preparation**

Performed the below steps to prepare and clean data to feed the machine learning algorithm.

* Converted data time values to Date Time using pandas datetime function.
* mapped missing values in one dataset (matches) from another dataset (shooting)
* performed average fill for missing values wherever necessary.
* Removed missing data (<5%).
* Added category codes for categorical variables.

**Exploratory Data Analysis**

Performed exploratory data analysis to find answers to a few high-level questions using matplotlib and seaborn libraries. Some of the questions are:

1. What are the home vs Away goals per team

2. Distribution of Home Win, Away Win, and Draw.

3. Totals goals by team

4. Total goals against

5. Average Possession per team.

6. showcasing best shots on target per team.

7. Best average distance shot per team.

8. Number of Free kicks

9. Total penalty kicks/attempted ratio.

**Our base model**

Created a simple Random Forest classifier to predict the results of the game on our train data and check its accuracy on test data. Also, checking the precision score evaluated from the model. The accuracy came out to be **60.15%.** which is a decent score for the first iteration of our base model. However, the predictors used in this algorithm were only ['Venue', 'Opp\_code', 'hour', 'day\_code']. From this base model, we have a threshold, and we now have to try working on our model to improve our accuracy.

**Steps to improve our model (Rolling Average)**

To improve our accuracy and precision, we can evaluate the rolling averages of a few of our predictors such as ["GF", "GA", "Sh", "SoT", "Dist", "FK", "PK", "PKatt"]. For this I have created a function that can calculate the rolling averages of the above 3 values for the above-mentioned predictors grouping by the teams and then removing the NaN values obtained.

**Model selection & training**.

A range of algorithms was selected and tested from the library: sci-kit-learn. All models were optimized using a grid search with a 3-fold cross-validation accuracy metric. Created a pipeline in the parameter grid so that we can change the preprocessor and classifier as well as their hyperparameters. This helped us to run multiple algorithms at once and see the best scores evaluated from all the algorithms.

* Support Vector Classifier
* XGBoost Classifier
* Random Forest Classifier
* Logistic Regression
* Gradient Boosting Classifier.

**Evaluation**

# We got the best accuracy score from RandomForestClassifier of 62.38% with the hyperparameters max\_depth=5, max\_features='sqrt', n\_estimators=200. As performing rolling averages did not increase the accuracy by many levels, but still, this is quite a decent score for our initial iterations. to improve the results, we might increase the dataset for more seasons and include other features

**Further work and Improvements**

* Collect additional data: the [website](https://fbref.com/en/comps/9/Premier-League-Stats) can supply numerous seasons of data prior to that collected in this project. Additional data collected will result in better clustering, especially those fixtures counted as a draw.
* Incorporate additional unused stats: the project only consisted of scores & fixtures with shooting stats per team, however, stats for goalkeeping, passing defensive actions, possession, and other miscellaneous should be incorporated into model building and compared with the existing model.
* Test alternative approaches to feature engineering. Currently, to reduce the number of features, game data over the past 10 fixtures are averaged. Dimensionality reduction techniques (PCA, LDA) should be tested on this raw data. These results may then be compared with the existing model.