CS6140 - Machine Learning - Spring 2024 - Paul Hand Football Prediction

Aneesh Gupta Vandanapu
vandanapu.a@northeastern.edu

Pratik Budhiraja
budhiraja.p@northeastern.edu

Varun Rao Kadaparthi
kadaparthi.v@northeastern.edu

Abstract

The prediction of football match outcomes has always been a subject of interest, not only for enthusiasts but also for professionals in various fields like sports betting and team management. In this project, we aim to develop a predictive model to forecast the outcome of English Premier League matches, particularly focusing on whether the home team will win. The dataset spans from the 2000-01 season to the 2017-18 season, providing comprehensive match statistics and results.

1 Introduction

Football, being the most popular sport globally, attracts a massive audience and generates significant interest in predictive analytics. The English Premier League (EPL) stands out as one of the most competitive and followed domestic leagues, offering a rich dataset for analysis and prediction. Our project focuses on leveraging this dataset to build a predictive model capable of determining whether the home team will emerge victorious in a given match. We refine the dataset for analysis, extracting pertinent match statistics. Subsequently, we employ three distinct machine learning models—Logistic Regression, Support Vector Machine (SVM), and Random Forest—to forecast match results. Our goal is to conduct a comparative analysis of these models before and after processing the data to ascertain their effectiveness in predicting winning football teams within the English Premier League.

2 Data Processing

The dataset comprises match details from the EPL seasons spanning from 2000-01 to 2017-18. Initially, the data is imported into a Jupyter Notebook environment using Python libraries such as pandas, matplotlib, numpy, and seaborn. Various preprocessing steps are then undertaken to refine the dataset for analysis:

1. **Data Importing and Selection:** Raw CSV files for each season are imported and relevant columns such as Date, HomeTeam, AwayTeam, FTHG (Full Time Home Team Goals), FTAG (Full Time Away Team Goals), and FTR (Full-Time Result) are selected for analysis.
2. **Goal Statistics Aggregation:** Functions are created to calculate the goals scored and conceded by each team, arranged by matchweek. This information is aggregated to get a cumulative count of goals up to a particular matchweek.
3. **Points Calculation:** Points are assigned to each team based on match outcomes (win, draw, or loss), and cumulative points are computed over matchweeks.
4. **Team Form Representation:** The form of each team, indicating its recent performance, is captured by considering the results of the previous matches. This information is added to the dataset.

Submitted to Professor and TAs - ML6140 - Machine Learning - Spring 2024.

5. **Feature Engineering:** Additional features such as Goal Difference (GD), Difference in Points, and Difference in Form Points are calculated. Categorical variables are converted into numerical representations.
6. **Data Splitting:** The dataset is split into training and testing sets to facilitate model training and evaluation.
7. **Model Evaluation:** Three different machine learning algorithms—Logistic Regression, Support Vector Machine (SVM), and Random Forest—are applied to predict match outcomes. The models are evaluated using classification metrics such as accuracy, precision, and recall.

By meticulously preprocessing the dataset and applying machine learning techniques, we aim to build a robust predictive model capable of forecasting English Premier League match outcomes, particularly focusing on the likelihood of the home team winning. The ultimate goal is to provide valuable insights for sports enthusiasts, betting professionals, and team strategists alike.

For the classification techniques, the dataset is divided into training and testing sets (70% training and 30% test split ratio) to assess the model's performance on unseen data.

3 Modeling and Results

We followed two approaches for this project. We tested how a model would operate in case we used a raw dataset and then, processed that dataset to see if there would be a significant difference in our final prediction performance. Our raw dataset consists of two types of data - results data and match statistics.
In the **first approach**, we only used the match statistics and no results data. This is all the information that we usually have before a match starts. This dataset was of size (6480 * 12).
In the **second approach**, we engineered new features from the data we already have, which provided the model a better understanding of how the team has been performing in the recent past. This resultant dataset is of size (6480 * 40).
We used both these datasets to train models using Logistic Regression, Support Vector Machine and Random Forest Classification algorithms.

3.1 Logistic Regression

Logistic Regression is a supervised machine learning algorithm used for binary classification tasks. It models the probability of the default class and is commonly used when the label is categorical.

Here's a brief summary of Logistic Regression's functionality and its implementation.

1. **Probability Estimation:** Logistic Regression estimates the probability of the home team winning the match based on the given match statistics. It uses the logistic function (sigmoid function), to fit the output of a linear equation between 0 and 1. The resulting output can be interpreted as probability that the home team wins.
2. **Decision Boundary:** Logistic Regression uses a decision boundary to classify the instances. In general, if the estimated probability is greater than 0.5, the model predicts that the instance belongs to the positive class. Otherwise, it predicts that it belongs to the negative class.
3. **Regularization:** Logistic Regression can be regularized to avoid overfitting. L1 regularization can be used for feature selection, as it shrinks less important feature's coefficient to zero, effectively excluding them from the model.

3.2 Support Vector Machine

Support Vector Machine (SVM) is a supervised machine learning algorithm used for classification tasks. Its main objective is to find a hyperplane in an N-dimensional space (N is the number of features) that distinctly classifies the data points into different categories.

In our case, SVM works by finding the hyperplane that maximizes the margin between the classes. In such a high-dimensional space as ours, it can perform efficiently by using a Kernel trick where it implicitly maps the input data into high-dimensional feature spaces, allowing for nonlinear decision boundaries.

Since the relationship between our data is non-linear, we opted for the **RBF kernel in SVM** because it's adept at handling complex, non-linear relationships between features without imposing any specific assumptions about the data distribution. It's versatile enough to capture intricate class distributions, offers robust generalization to new data, and effectively maps the data to a higher-dimensional space, enabling SVM to discern complex decision boundaries.

3.3 Random Forest

Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the mode of the classes (classification) or the mean prediction (regression) of the individual trees. It operates by:

1. Randomly selecting subsets of the dataset (bootstrap samples) for training each tree.
2. Randomly selecting a subset of features for determining the best split at each node.
3. Combining the predictions of individual trees through averaging/voting.

We tried multiple configurations for the Random Forest Classifier, and after extensively testing it, we found out that the following configuration works the best for our dataset - **Gini impurity criterion** (utilized to evaluate the quality of splits in decision tree nodes, aiming to minimize impurity), **700 decision trees**, and each decision tree is restricted to have a maximum of **10 leaf nodes**.

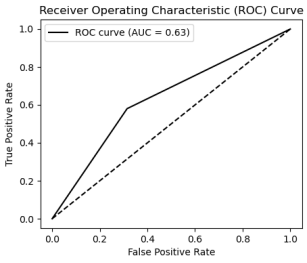
4 Discussion

Table 1: Model Performance with raw data

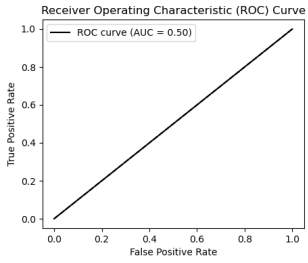
Metric	Logistic Regression	Support Vector Machine	Random Forest
Accuracy	0.54	0.55	0.54
Precision	0.49	0.55	0.51
Recall	0.54	0.50	0.54

Table 2: Model Performance after Feature Engineering

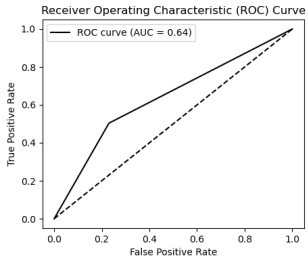
Metric	Logistic Regression	Support Vector Machine	Random Forest
Accuracy	0.64	0.54	0.65
Precision	0.64	0.29	0.65
Recall	0.64	0.54	0.65



(a) Logistic Regression



(b) Support Vector Machine



(c) Random Forest

We can observe how data processing has affected the performance of these models. The Support Vector Machine performed better when trained with raw data. Feature engineering increased the performance of both Logistic Regression and Random Forest Classifiers by ~18%.

Contrary to our expectations, SVM performed poorly with processed data as compared to its unprocessed counterpart. This might be a result of our data normalization technique where we aggregated the previous match results as a performance metric for the home team. A future scope for our project would be to look further into our data processing and modify the model parameters.