Faculty of Engineering & Applied Science

SOFE 4620U- Machine learning & Data Mining

Soccer Match Predictor

Final Report

Date: April 2nd, 2023

Group Number: 15

Course instructor: Khalid Abdel Hafeez

Github link:

https://github.com/Waleed20210/Final-Project-Machine-Learning-and-Data-Mining

Student Name	Student Id
Waleed El Alawi	100764573
Muhammad Zaeem Khalid	100746801
Carson McClelland	100725653
Gutu Shiferaw	100767090

Abstract

The soccer match predictor system is an advanced machine learning model designed to predict the outcomes of soccer matches based on historical data and team performance. The system uses various algorithms and techniques such as logistic regression, decision tree, and SVM to analyze team performance and to predict accurate results. The accuracy of the system has been evaluated using older datasets from different sessions and years. Furthermore, this report focuses on explaining the methodologies and design decisions used for the system, including its features and dataset used. By providing a comprehensive overview of the system this report aims to demonstrate its effectiveness in predicting soccer match outcomes.

Table of Contents

Introduction and Background

The idea of this project is to design a machine learning algorithm that can predict the results between different teams and choose the best team. We will be trying multiple frameworks/libraries such as Sklearn and TensorFlow before choosing which one will fit and work with our model. Our dataset will include information about players performance, player rating, team performance, and team rating. This project will allow fans and the general public to predict who will most likely win the next soccer game. Our project will be applicable and beneficial to the sports betting industry, sports commentary and analysis industry, and general debates between fans of soccer teams.

The inspiration for this project is from a time thousands of years ago when humanity had started to conduct forms of entertainment to unify the common populace. Many forms of entertainment were created ranging from basic theater performances to violent fights. A very famous entertainment was the chariot racing in the colosseum in the Roman empire, many people wanted to earn money off placing bets and people were just curious about who would win as well [1]. The early forms of entertainment analysis started by analyzing the win rate, physique, skill level, and experience of participants; smart individuals were able to make educated guesses about which fighter would win. The modern form of entertainment that rivals this ancient entertainment in terms of popularity is football(or soccer in North America), which is why we wanted to create software that can predict and analyze the matches in one of the most popular sports leagues in the world, the premier league.

Project Objective

Our goal is to create a machine-learning algorithm that can predict who will win in a game between two soccer teams. This algorithm will take into account various different situations based on the dataset that is created. These situations include where the game is taking place (ie. Team A has home-field advantage), which team won the past games, and which team has higher statistics like shots on target. This will allow the algorithm to predict with high accuracy which team will win. We designed this application for different use cases such as sports betting and sports analytics.

Project Methodology

To solve our project of predicting the result of a soccer match, we adopted a methodology that involved implementing a range of machine learning algorithms and data processing techniques. By using multiple algorithms and techniques, we were able to investigate and analyze the various methods available and compare their effectiveness. This approach allowed us to gain a deep understanding of the different methods and their potential applications.

To predict the outcome of a Premier League match, our team decided to implement and analyze five different algorithms: Decision Tree, Support Vector Machine, Logistic Regression, Random Forest, and Gradient Boosting. To start, we created a skeleton code that formed a base framework for importing the dataset and extracting the desired features. Then utilized Premier League datasets from previous seasons, which were stored in CSV files with more than 50 unique features. We then selected a subset of features that would be used to train the model and predict team ratings based on match attributes. The selected features included HomeTeam, AwayTeam, FTHG (Full Time Home Team Goals), FTAG (Full Time Away Team Goals), HS (Home Team Shots), AS (Away Team Shots), HST (Home Team Shots on Target), AST (Away Team Shots on Target), HF (Home Team Fouls Committed), AF (Away Team Fouls Committed), HC (Home Team Corners), AC (Away Team Corners), HY (Home Team Yellow Cards), AY (Away Team Yellow Cards), HR (Home Team Red Cards), and AR (Away Team Red Cards).

After creating the initial framework for cleaning, transforming, and organizing the dataset, our team implemented unique data processing techniques for each of the five machine learning algorithms. The data processing techniques we implemented were encoding categorical data, normalizing the data, feature scaling, feature engineering, and splitting the data into training and test sets. To ensure accurate predictions, we encoded each team's name for both the home and away. This allowed the model to give a rating to each team based on the other attributes. Next, the team normalized and scaled the data in a manner specific to each algorithm. These techniques ensured that the data was transformed appropriately for the algorithms to accurately learn from it. The final step was to split the dataset into training and test sets. This step was critical to avoid overfitting or underfitting the resulting model. The team followed the standard training-to-test ratios of 80% and 20% or 70% and 30%, depending on the algorithm used.

The final step was to train and evaluate our models. We used popular libraries such as TensorFlow and Scikit-Learn. We then evaluated each model's performance based on the metrics of accuracy, precision, recall, and F1-score, allowing us to compare each algorithm independently and fine-tune the models by adjusting features, bias, and variance. Our goal was to create the highest-performing models, and ultimately, compare the five algorithms to select the best one as our final model.

Detailed Design

Framework/tools

We used python as the main language in our project. Python allows us to have the options, flexibility, and scalability needed to accomplish our goals. The main machine learning framework we utilize is TensorFlow and scikit-learn. TensorFlow is an open-source library that allows for building deep learning models while remaining simplistic and organized. There are lots of data, algorithms, and resources available about soccer. This allows for the datasets to be quickly populated and modified to best conform to our desired results. We plan on testing multiple algorithms to ensure our final product produces the best results, a few of the algorithms we plan on implementing are; Logistical regression, K-nearest neighbor classifier, Linear Regression, and Random Forest. Finally, we need a user interface for the use of our algorithm and one of the options for implementing a user interface is through the use of the python tkinter library for creating and designing UI.

Dataset

We got our dataset from the Kaggle website. It is a platform for data scientists and machine learning practitioners to find and share various datasets. This dataset is very useful for analyzing trends and performance over time. It can also be used to build a predictive model to forecast future outcomes. The dataset we used includes CSV files containing data from previous league matches, teams, and players as well as the number of wins, draws, losses, and goals scored and conceded. Some of the key variables included in the dataset “Data”, “HomeTeam”, “AwayTeam”, “FTHG”, “FTAG”, “FTR”, “HS”, “AS”, “HF”, “AF”, “HC”, “AC”. It also includes some extra variables about the total number of shots, fouls, and corners for each match. Overall, this dataset provided us with the required information to train and test our models and to predict future outcomes.

Results

In this section, we will explore the results of comparing each algorithm against the test set, validation set, and each other. We implemented and analyzed five different machine learning algorithms: Decision Tree, Support Vector Machine, Logistic Regression, Random Forest, and Gradient Boosting. After training and evaluating each model's performance based on the metrics of accuracy, precision, recall, and F1-score, our team was able to compare each algorithm independently and against each other. The results and corresponding scores can all be found in the appendix.

When comparing the performance of each algorithm against the test set, and validation set Gradient Boosting was found to have produced the best-performing model. This result aligns with previous studies that have shown Gradient Boosting to be highly effective for predicting outcomes in sports matches [4]. Gradient Boosting uses boosting techniques, where it iteratively adds weak learners to the model, each focusing on improving the errors of the previous one. This allows Gradient Boosting to build an effective, and accurate model that can handle complex relationships between all the features.

The Random Forest algorithm was found to be overfitting, it performed extremely well on the test data but when tested against the validation test set it performed poorly. Even when we modified the algorithm with early stopping, regularization, and fewer features it still did not perform adequately.

Regarding the other three algorithms, Decision Tree, Support Vector Machine, and Logistic Regression, the performance results were very similar when comparing between test set, validation set, and each other. It is possible that these three algorithms were too simple to capture all the attributes and distinctions of the Premier League match dataset, resulting in lower accuracy compared to more complex models like Random Forest, Gradient Boosting, and Neural Network. Moreover, these models may suffer from underfitting, they may not be complex enough to fit the data and therefore fails to capture the relationships between the input features and the output variable.

Conclusion and Recommendations

Our goal was to create a software model that can predict which teams will win when playing against each other in the English Premier league. We selected 5 different algorithms to create models from which were Decision Tree, Support Vector Machine, Logistic Regression, Random Forest, and Gradient Boosting. Based on the accuracy and factoring in overfitting and underfitting we found that Gradient boosting was the most effective model for our problem. We used python, Tensorflow and Sklearn to create these models and used these frameworks for our overall software. Furthermore, we created a small user interface to input the game matchups using the Tkinter library. Overall we can predict which team would win with a fairly high degree of accuracy.

Some limitation of our approach is the limited amount of features that our merged dataset has, this can be fixed by including different datasets with player data, environment data, and sports politics data. Another limitation is the number of new variables introduced frequently, this can be rectified by retraining our program model with the newest data as it changes. Making the system cloud-based and distributing each individual part as microservices could invoke loose coupling and high cohesion.

In the future, our approach can be expanded to include individual player data and statistics, through this we can determine which team has a better chance of winning, and even predict how a team will do when new players are being drafted or transferred into the team. Furthermore, this expansion can help fantasy football team creators, general managers who are looking to create or improve their teams, and fans who wish to understand which players would be able to bring their teams to new heights. Another important aspect that can be expanded upon is a dataset, this can be expanded to include other soccer leagues from around the world, furthermore with slight modifications the software would even be able to be implemented in different sports leagues like the MLB and NBA.

Contribution Matrix

Table 1: Contribution Matrix

Task	Waleed El-Alawi	Muhammad Zaeem Khalid	Carson McClelland	Gutu Shiferaw
Abstract	85%	5%	5%	5%
Introduction and background	5%	45%	5%	45%
Project Objective	5%	5%	5%	85%
Project Methodology	40%	10%	40%	10%
Results	25%	5%	45%	25%
Conclusion and Recommendation	5%	85%	5%	5%
Appendices	5%	5%	85%	5%
Total	25%	25%	25%	25%

Table 2: Overall Contribution Matrix

Task	Waleed El-Alawi	Muhammad Zaeem Khalid	Carson McClelland	Gutu Shiferaw
Phase 1: Project Planning & Requirements Gathering	25%	25%	25%	25%
Project Development and Testing	25%	25%	25%	25%
Final Report	25%	25%	25%	25%
Project Presentation	25%	25%	25%	25%
Total	25%	25%	25%	25%

References

[1]P. J. Kiger, “Chariot Racing: Ancient Rome’s Most Popular, Most Dangerous Sport,” *HISTORY*, Mar. 28, 2022. https://www.history.com/news/chariot-racing-ancient-rome (Accessed 2 Apr. 2023).

[2]“English Premier League,” *www.kaggle.com*. https://www.kaggle.com/datasets/saife245/english-premier-league?select=final_dataset.csv (accessed Feb. 05, 2023).

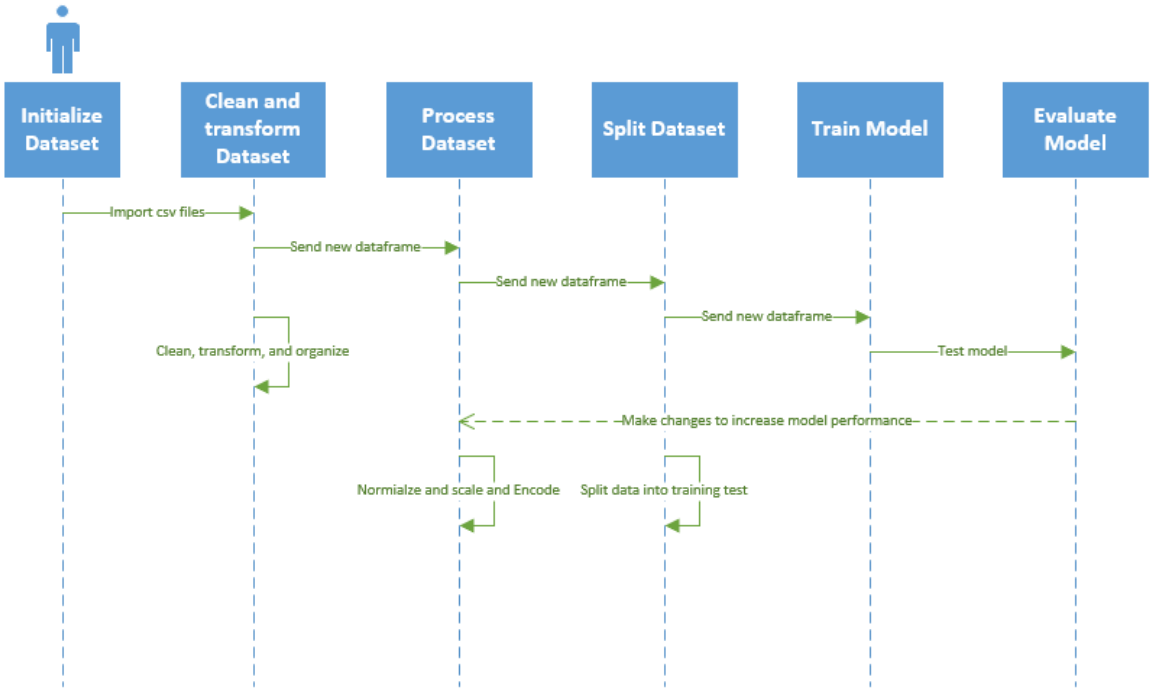
[3]A. Caldas, “Beating soccer odds using Machine Learning — Project Walkthrough,” *Analytics Vidhya*, Jun. 30, 2021. https://medium.com/analytics-vidhya/beating-soccer-odds-using-machine-learning-project-walkthrough-a1c3445b285a (accessed Feb. 05, 2023).

[4]Duarte, Denio, and Jefferson Alexandre Coppini. "MACHINE LEARNING APPROACHES TO PREDICT THE MATCH RESULT: BRAZILIAN FUTSAL LEAGUE CASE." *Revista Brasileira de Futsal e Futebol*, vol. 13, no. 53, May-Aug. 2021, pp. 275+. *Gale OneFile: Informe Académico*, link.gale.com/apps/doc/A683425987/IFME?u=anon~e938f84d&sid=googleScholar&xid=ca9b77cc. (Accessed 2 Apr. 2023).

[5]Simplilearn, “Gradient boosting algorithm in python with scikit-learn: Simplilearn,” *Simplilearn.com*, 24-Feb-2023. [Online]. Available: https://www.simplilearn.com/gradient-boosting-algorithm-in-python-article. [Accessed: 02-Apr-2023].

Appendices

UML sequence diagram depicting model creation process



Train vs test results of the five algorithms

```
Accuracy (Decision Tree): 0.6197368421052631
 Precision (Decision Tree): 0.6411960132890365
 Recall (Descision Tree): 0.516042780748663
 F1-score (Descision Tree): 0.5718518518518518
```

```
Accuracy (Logistic Reg): 0.6789473684210526
 Precision (Logistic Reg): 0.7044025157232704
 Recall (Logistic Reg): 0.5989304812834224
 F1-score (Logistic Reg): 0.6473988439306357
```

```
Accuracy (SVM): 0.6421052631578947
 Precision (SVM): 0.656441717791411
 Recall (SVM): 0.5721925133689839
 F1-score (SVM): 0.6114285714285714
```

```
Random Forest Classifier:
Accuracy: 0.98285536159601
Precision: 0.9937673130193906
Recall: 0.9689399054692776
F1-score: 0.9811965811965813
```

```
Gradient Boosting Classifier:
Accuracy: 0.9978179551122195
Precision: 0.9952956989247311
Recall: 1.0
F1-score: 0.9976423038059953
```