Improving Football Betting Predictions with Machine Learning

Group 34

1. Introduction

We chose to attempt to build a prediction model that will be based on the Premier League games. The problem we are tackling is allowing users to make accurate predictions while making bets on football matches. The dataset will be used to predict which team is most likely to win based on the previous encounters with the particular club and the wins they have had recently and in the past. Using this will hopefully help users to improve their chances of making correct bets. We found this an interesting problem to take on because we want to see if using a prediction model like this can increase the success rate of betting. Our model will take data from previous Premier league fixtures as input. We will then use a number of classifier algorithms (Deep Neural Network, Deep Neural Network with Principal Component Analysis and Random Guess) to make predictions. Our desired outcome is to create a model that predicts results with a high degree of accuracy.

2. Dataset and Features

We made our dataset with the help of premier league match results scraped from football-data.co.uk. We used data from season 2000-2022 onwards as this data had the most consistency in features throughout.

On initial review, the data had a variety of features that document the events of each game, such as the amount of goals scored by the away team (FTAG). The table below explains the meaning of each feature.

Date	Match Date (dd/mm/yy)
HomeTeam	Home Team
AwayTeam	Away Team
FTHG and HG	Full Time Home Team Goals
FTAG and AG	Full Time Away Team Goals
FTR and Res	Full Time Result (H=Home Win, D=Draw, A=Away Win)
HTHG	Half Time Home Team Goals

HTAG	Half Time Away Team Goals
HTR	Half Time Result (H=Home Win, D=Draw, A=Away Win)
Referee	Referee
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
AR	Away Team Red Cards

Combining all the seasons together we have 8506 matches from all the seasons.
We had to transform the raw data into a format suitable for predicting results. We created a database of games, keyed on season and team, and season-long statistics that can be used as fixed team attributes. Then we created a table of features and targets by querying the built-up database, with parameters to select the number of games used. This will allow us to be flexible in our parameter choice.

Creating the Database

We first take the raw data and load into multiple pandas dataframes keyed by the season. Then, we build the subset tables for matches. For each season, all of the games (both home and away) for each team will be noted, with the data concerning the games stored as a seasonal subset. We noticed there was a clear difference in the stats for home and away games.For example, it was seen that on average home teams had more shots, whereas the away team had more fouls. This was apparent across all of the data. As a result, we split our data into home team and away team data. Additionally, the outcomes of the matches will be recorded as a feature, and one-hot encoded into Win, Lose, and Draw features.We can use the command at the end to query this database for each team and season, for example subset[21]['Chelsea'] returns all of Chelsea's results for the 2021 season.

Using this database, we can create another queryable database that stores season-long stats, such as Points, Goal Difference, Red Cards, etc. This database is designed to store

data for a particular season on all games, both home and away, so that the statistics for each can be accessed separately.

Once the databases are ready, we created the proper feature/target list by querying the databases. We evaluated a substantial amount of possible features. These features were divided into the related data for the home team's past results and the away team's. The tables underneath display this for the home side (HS). The same features were extracted for the away side (AS).

Previous HS game	**Previous X HS games (5 by default)**	**Previous X HS-home games (5 by default)**	**HS Previous Season Data**	**Previous Season Home Record**	**HS vs Away Team Record**
Games Played			Position	home Draws	Win
		Average Corners	Draws	home Goals	Draw
	Average Corners	Average points	GD	home GoalsAgainst	Loss
Corners	Average points	Average YCards	Goals	home Losses	Goals
Win	Average Yellow Cards	Average RCards	GoalsAgainst	home RCards	GoalsConceded
Draw	Average Red Cards	Average Goals	Losses	home Team	BigChancesCreated
Loss	Average Goals	Average GoalsConceded	Points	home Wins	Average BigChancesCreated
Goals	Average GoalsConceded	# of Wins	RCards	home YCards	Average Corners
GoalsConceded	# of Wins	# of Losses	Team		Average points
Fouls	# of Losses	# of Draws	Wins	home Average Corners	Average YCards
Fouls Against	# of Draws	# of RCards	YCards	home Average CornersAgaints	Average RCards
Red Cards	# of RCards	# of YCards		home Average Fouls	Average Goals
Red Cards Against	# of YCards	# of YCards	Average Corners	home Average FoulsAgainst	Average GoalsConceded
Yellow Cards	# of YCards		Average CornersAgainst	home Average Goals	
Yellow Cards Against			Average Fouls	home Average GoalsAgainst	
Shots			Average FoulsAgainst	home Average Shots	
Shots Against			Average Goals	home Average ShotsAgainst	
Shots On Target			Average GoalsAgainst		
Shots Against On Target			Average Shots		

For each column of the above, we made specific code to query the database. The get_targets will produce the game results as the target column and also convert the data into a one-hot encoding format of ['Win','Lose','Draw'].

We went through every match of the following seasons, noting the match number, date, home side and away side. We then ran queries with the given parameters against the database, compiling the results and feeding them into the relevant tools to access the relevant features.

We have a large number of features at 156, especially since we only have 6025 samples to train with. This could lead to the curse of dimensionality, which is when the feature space increases quickly as new dimensions are added. This means that more samples would be necessary to create models with accuracy. To avoid this, we use feature selection using Principal Component Analysis discussed in the section below.

3. Methods

The group had initially planned to use some of the methods learned in class to model the problem however the methods learned in class were not effective in doing this. As a result the group had to research other methods for modelling this problem. From reading the current literature on the topic, it seemed that an Artificial Neural Network would be the most effective solution. We had seen Convolutional Neural Networks in class, however we knew that they were usually applied to visual data like images, and used kernels to create convolution layers. Our data was numerical, which led us to Deep Neural Networks. A deep neural network has layers similar to the convolution neural network that we worked with in class, however instead of multiplying kernels across images it uses fully connected layers to output a probability given a set of inputs. As the number of features was high in comparison to the input size, we used Principal Component Analysis (PCA) to reduce the number of features. Another DNN was then trained on the most important features in order to try and improve performance. Finally, a random guess baseline classifier was used to get a baseline for comparison.

Deep Neural Network

A deep neural network (DNN) is a neural network that has some degree of complexity, often at least two layers. Deep neural networks use advanced maths modelling to analyse input in complicated ways. The initial stage is the Contrastive Divergence algorithm, where the model learns a layer of features from visible units. After that, the model learns features of features and considers the activations of previously learnt features as visible units. When the learning for the last hidden layer is accomplished, the whole network is

trained. Recurrent neural networks provide sequential and parallel computing. as the human brain does (large feedback network of connected neurons). They can recall crucial details from the feedback they got, which helps them to be more exact.

Principal Component Analysis

Principal component analysis (PCA) has a wide range of uses, including exploratory data analysis, dimensionality reduction, information compression, data de-noising, and many more. Finding these principal components, which may characterise the data points with a collection of principal components, is the primary goal of PCA. Vectors make up the primary components, but they are not picked at random. To account for the most variation in the original characteristics, the first main component is calculated. After the first principal component, the second component, which is orthogonal to the first, explains the most variation. Feature vectors may be used to represent the original data. Using PCA, we may describe the data as linear combinations of the principal components, taking this one step further. PCA has the benefit of being simple to calculate.It is based on linear algebra, which is easily solved by computers computationally. It also has the benefit of accelerating other machine learning methods. It also offers the benefit of addressing the drawbacks of high-dimensional data.

Random Guess

The Random Guessing model makes the assumption that data behaves impartially and randomly. The general random guessing model predicts one of the possible outcomes randomly and impartially. This is useful in providing a baseline to compare another model's performance against.

4. Experiments/Results/Discussion

Our experiments consisted of first using a random guesser model to predict the target results. An accuracy score of 30% was observed. This was expected as there were three possible outcomes, win,lose or draw. This would be used as our baseline classifier, and would indicate if our model was beneficial. Our next model was a Deep Neural Network trained on a training dataset. The model did not undergo any feature reduction. The accuracy score for the model was 41% on average. This was a significant improvement over the baseline classifier. In order to improve performance, a second model was trained. This model used the same training dataset as the first, however PCA was performed in order to reduce the amount of features used. The optimal number of features determined by the model was 67. This PCA deep neural network had an accuracy score of 51% on average. This was a huge improvement over both of the previous models.

In order to increase performance further, more features could be added. Our model did not take into account individual players, or advanced metrics such as expected goals. However, from our research it did not seem like these features were worth the amount of time taken to implement them relative to their performance benefits. We were happy with our model's performance as its performance was approximately 20% more accurate in comparison to the random guesser baseline.

5. Summary

We wanted to improve the accuracy of sports prediction in order to optimise users chances when placing bets. Among the three models we tested, Deep Learning with Principal Component Analysis was able to make the most accurate predictions compared to the other two models. As we have used data from only the last 20 years, the Deep Learning model gave a good enough improvement in the prediction compared to the baseline model (Random Guess). Over the course of our research, we found out that professional sports bettors have an average accuracy of 65%. This level of performance was unachievable for our simple model. As the average user is not a professional sports bettor, we believe that our model has the possibility to optimise a user's bets as it gives them a more accurate prediction over randomly picking an outcome.

6. Contributions

Weekly meetings were held to discuss decisions, progress made and steps forward.

Paddy

- Deep Neural Network Model
- Principal Component Analysis Code.
- Sections 3 and 4 of the Report.
- FeatureBuilder function
- Part of Helper Function code.

John

- Collected Premier League from the year 2000.
- Cleaned and filtered collected data to match selected features.
- Code to build dataset and subsetsfrom individual season files.
- Dataset and Features section of the report.

Sankeerth

- Introduction, method, summary and references part of report.
- Implemented Random Guesser Model.

7. References

These are the references which we used for our project. We found our data sets on the “football-data” link which I have mentioned below. We as a group then read the article on medium.com which gave us an idea of how we are going to approach our solution.
https://www.football-data.co.uk/notes.txt
https://www.football-data.co.uk/englandm.php
https://medium.com/nerd-for-tech/premier-league-predictions-using-artificial-intelligence-7421dddc8778
https://towardsdatascience.com/modeling-expected-goals-a756baa2e1db
https://www.geeksforgeeks.org/introduction-deep-learning/#:~:text=Deep%20Neural%20Network%20%E2%80%93%20It%20is,is%20multi%2Dlayer%20belief%20networks.
https://www.keboola.com/blog/pca-machine-learning (website used)

Team Members

- **Patrick O Callaghan - 19334526**
- **John Kommala - 19303445**
- **Sankeerth Karpe - 18318175**

Github Repository

https://github.com/paddyocallaghan/Machine-Learning-34.git