Prediction and Analysis Based on English Premier League

Yuan-Hsi Lai
Dept. of Electrical Engineering
Columbia University
yl4305@columbia.edu

Jin Huang
Dept. of Electrical Engineering
Columbia University
jh4137@columbia.edu

Qiaoyu Gu
Dept. of Electrical Engineering
Columbia University
qg2172@columbia.edu

Abstract

In this work, we get our hands on the data of English Premier League, which is the top level of the English football league system. The data is consisted of matches from past four year with detailed team and player statistics of each match. We focused on four aspects in our work: Player Playstyle Clustering, Player Influence Analysis, Team Performance Prediction, and Match Result Prediction. The methods we used include K-means, Long short-term memory (LSTM), and several classification tools including Random forest, XGBoost, and Multilayer perceptron (MLP). The final proposed match prediction model is the combination of LSTM team performance model and a random forest classifier. We also visualized the results on webpage based on D3js and JavaScript.

1. Introduction

The English Premier is the top level of the English football league system. Contested by 20 clubs, it is the world's most lucrative football league (Manchester United have hit a record revenue of £627 million in 2018-2019 season). It is also the world's most-watched sports league, broadcast in 212 territories to 4.7 billion audience, which is extremely valuable and profitable. As a result, we aim to design a prediction and analysis system that has wide use for multiple characters in the English Premier League System. For example, for club managers, the system will be useful to them since they will need some player data collection or analysis for them to develop strategies of transferring and buying players; for audiences, they may be curious about the match results or player information query; and for media, they would like to know the match prediction or team performance so they can write more news.

We noticed that there are lots of websites that show match information, team and player query, for example the official website [1] of English Premier League, ESPN [2]. And there are also some prediction websites such as Foot Ball Predictions [12]. However, those websites are either only functional of statistics querying, or their predictions are based on human prediction and are not reliable. As a result, the goal of our work is to provide a complete, reliable, and precise system that has the function of Player Play-style Clustering, Player Influence Analysis, Team Performance Prediction, and Match Result Prediction.

For the player playstyle clustering, we employed K-means to the average statistic including touch, save, goal, pass… etc. of each player and clustered them into 10 styles. For the Player Influence Analysis, we first designed an algorithm for team performance of each match which is weight summed by goal, passing accuracy, contest success rate, shots conversion and possession rate. With this performance score per match, we can calculate the player's influence value by the performance of team that the player attended. For Team Performance Prediction, we developed a two-layer LSTM model that reads the performance score of past five games and output the predicted performance score. In the last part Match Result Prediction, we propose a combined model that first uses LSTM model to predict the performance based on past five games and by the performance of the two team, we used random forest to predict the match result. The result of a match consists of three categories: win, draw, and lose, and the method we proposed reached 65% of accuracy.

2. Related Work

There are quite a lot of match result prediction methods, L.M. Hvattum and H. Arntzenin in [4] used ELO ratings for match result prediction, F. Owramipur in [5] used Bayesian Network for Spanich League football match prediction, and in [13]. In [4], the work simply calculated the two teams' ELO difference and used a regression model to predict the result. However, due to the changing environment of football league nowadays, this method could fail by not considering a team's recent performance. In [5], it used several reasonable factors including weather, psychological state, player age, injured player… etc., to predict the match results. The factors used are pretty good but however, factors like weather, injured players are very detailed data that was hard to collect. Most importantly, the 92% accuracy that it reached was trained and tested without splitting the data and was only based on a single season. For

[5], it claimed that it reached an average of 75% accuracy in predicting three different season's matches. However, the attributes that it used was team shots, team fouls, yellow cards, and red cards, which are useless inputs for an unplayed upcoming match since the only data we know about an unplayed match are only the teams and the players.

3. Dataset

The dataset we use is downloaded from Kaggle [6]. It records specific data of the English Premier League from Season 2014-2015 to Season 2017-2018. In each season's data, there are two json files. One of them describes in-game match stats for every match of the past 4 seasons (current season included) like player touches, passes, shots, yellow cards, saves etc. Some of the stats are available as aggregate stats for the entire team and some of them are player specific. Second file describes general match outcomes like the full time and half-time score etc.

This dataset contains several dimensions of matches, including more than 1,000 players, 1,500 matches and 1M records. Thus, it is valuable resource to support many works, such as evaluating player influence, commenting on team performance and predicting future match results, etc. Since the data is deep layers of json data (Figure 1), we still need to do seriously pre-processing in order to get the features we want. We first extract the player data of each match and the team data of each match to two csv files. Based on the two files, we were able to do query and data extraction on them.


```
"Player_stats":{
    "Jonas Lössl":{
        "player_details":{
            "player_id":"131171",
            "player_name":"Jonas Lössl",
            "player_position_value":"1",
            "player_position_info":"GK",
            "player_rating":"6.61"
        },
        "Match_stats":{
            "good_high_claim":"2",
            "touches":"39",
            "saves":"4",
            "total_pass":"28",
```


Figure 1: json file of our data

4. Methods

We implemented several methods in clustering, classification, regression and neural network.

4.1. K-means

K-means clustering is a method of vector quantization for cluster analysis in data processing. In this model, Euclidean distance is used as a metric and variance is used as a measure of cluster scatter. It identifies k number of centroids, and then allocates every data point to the nearest cluster, while keeping the centroids as small as possible, by minimizing within-cluster variance.

It is considered as one of the most used clustering algorithms due to its simplicity. In our work, we implement it to classify players based on their playstyles. Such classification can offer reference to managers of each club to make transfer strategy through searching players with similar styles.

4.2. Long short-term memory (LSTM)

Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. It is developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.

LSTM networks are well-suited to classifying, processing and making predictions based on time series data, since there can be lags of unknown duration between important events in a time series. LSTMs were developed to deal with the exploding and vanishing gradient problems that can be encountered when training traditional RNNs.

4.3. XGBoost

XGBoost model evolves from decision tree. It is a decision-tree-based ensemble machine learning algorithm that uses a gradient boosting framework. It is a scalable and accurate implementation of gradient boosting machines and it has proven to push the limits of computing power for boosted trees algorithms, as it was built and developed for the sole purpose of model performance and computational speed.

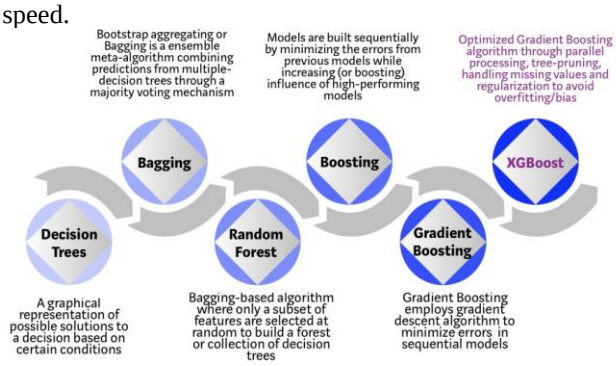

Figure 2: Evolution of XGBoost Algorithms [7]

4.4. Random Forest

Random forest is an ensemble learning method for classification, regression and other tasks that operates by

constructing a multitude of decision trees at training time and outputting the class that is the mode of the classes or mean prediction of the individual trees.

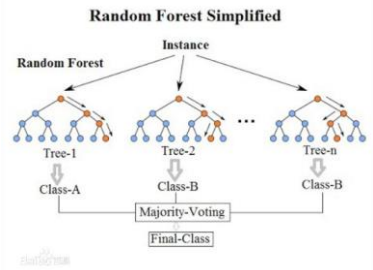

Figure 3: Architecture of random forest [8]

This algorithm has several advantages for us to utilize in the project:

- It provides reliable feature importance estimate.
- It offers efficient estimates of the test error without incurring the cost of repeated model training associated with cross-validation
- It supports large amount of input variables.

4.5. Multi-Layer Perceptron

Multi-layer perceptron is a logistic regression where instead of feeding the input to the logistic regression, we insert an intermediate layer, called the hidden layer that has a nonlinear activation function, which can figure data that is not linearly separable out.

Besides hidden layer, there are another two layers called input layer and output layer. It can map multiple input data sets to a single output data set through three layers. This model also implements a supervised training policy called backpropagation.

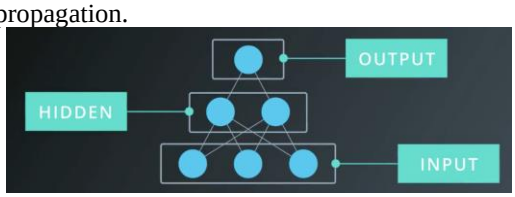

Figure 4: Three layers of multi-layer perceptron model [9]

One of the main advantages of multi-layer perceptron is its capability of generalization. In other word, it is able to effectively classify an unknown pattern with other known patterns that share the same distinguishing features.

4.6. Team performance algorithm

After each match, it is common that all players and each team will be rated according to their performance in the game. However, most of such rating depends on raters' subjective wills, thus could not be directly used to predict future results.

In our project, we designed a team performance evaluation algorithm, which can obtain relatively objective rating for both team and each player in one match, by taking several in-game attributes into calculation. Figure 5 shows how the team performance is calculated by goal, passing accuracy, contest success rate, shots conversion and possession rate. Each attribute counts different weights according to its influence on the whole game.

```
score = GOAL/5*24 +\
        ACC_PASS/TOTAL_PASS*13 +\
        GOAL/TOTAL_SCORING_ATT*13 +\
        AERIAL_WON/(AERIAL_WON+AERIAL_LOSS)*13 +\
        POSSESION_PERCENTAGE/100*20
```

Figure 5: The algorithm of team performance

5. Experiment

In this section, we will explain how we utilize the methods above to achieve our goals.

5.1. Player Playstyle Clustering

In this section, we would like to cluster the players into different types of play style. The input data we decided to use is the average player statistics, which is achieved by aggregating stats of each player in each match and do average. As mentioned above, K-means is an effective method of clustering. However, one of the drawbacks of K-means is that the number of clusters has to be determined by users. There's a method so called elbow method [4], which the idea is to run k-means clustering on the dataset for a range of values of, and for each value of k calculate the sum of squared errors (SSE). Then after plotting a line chart of the SSE for each value of k, if the line chart looks like an arm, then the "elbow" on the arm is the value of k that is the best.

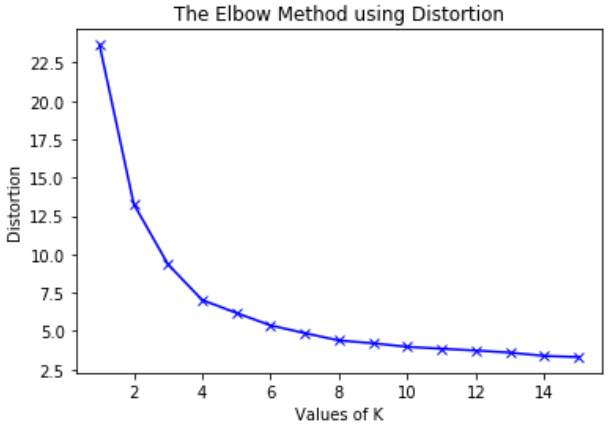

Figure 6: SSE plot of Elbow method

As shown in Figure 6, the elbow shows at k = 4. However, as the number of positions in football being four, which are forward, mid field, back guard, and goalkeeper, we believe four clusters can't effectively cluster the players into different playstyles. Therefore, we decided to cluster

the players into 10 different styles. Figure 7 is the result of our clustering after applied the T-SNE [5] method for visualization.

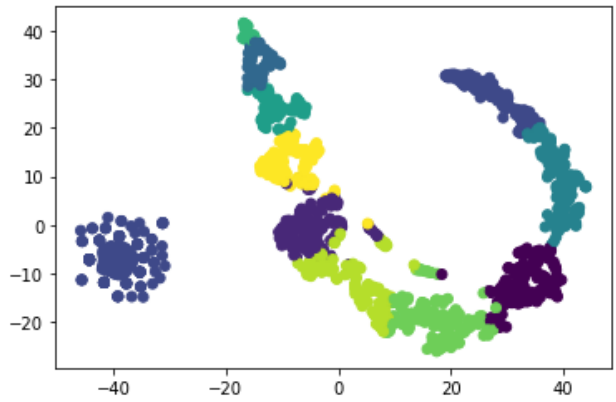

Figure 7: Cluster result of player playstyle

5.2. Player Influence Analysis

A player's influence value is an important indication of his ability, how much salary he should get, and whether a team should keep him or not. To calculate a player's influence value, an intuitive way is to calculate by the player's statistics. However, we know that there are several different positions in football, and players at different position will have different statistics. Therefore, it is hard for us to judge whether a player is influential only based on his statistics.

We came up with an algorithm that is to calculate the team's performance when the player is attending, which will be a better indication of the player's influence, because if every time this player plays and team performs well, we can infer that he has great influence to the team. Thus, we utilize the Team Performance Algorithm mentioned earlier to calculate each team's performance each game and transform into a player's influence. Specifically, a player's influence is calculated by:

$$\frac{\sum T_P}{25 \times \sqrt{n}}$$

Where T_P is the team's performance score when the player is attending. The original thought was to calculate the average team performance of the player playing, and 25 is some constant to scale the influence value to some range. However, we notice that there are some players that only played little games, and coincidentally, the team performances are all great. It turns out that there are some times that teams are winning a lot, they will send some substitute players to play for fun so the substitute players get good performance scores. Therefore, we add a square root of n so that it counts on the total games that the player has played, with more games played the influence value will be higher. Below are the sample results of most influential players in team Arsenal.

name	value
Petr Cech	19.2
Nacho Monreal	18.8
Alexis Sánchez	18.6
Héctor Bellerín	18.6
David Ospina	17.9
Olivier Giroud	17.9

Figure 8: Top valued players in Arsenal

5.3. Team Performance Prediction

For team performance prediction, we used the same algorithm we designed every team's history performance. What we want to do here is to base on a team's past performance, we predict the performance of next match. This is a kind of sequence prediction, and we know that for sequence prediction, it is best to use recurrent neural networks (RNN) since our data is based on time sequences. More specifically, we used an alternative version of RNN which is Long short-term Memory (LSTM). The advantage of LSTM is that it will preserve some feature from the previous input, so that it will have some short-term memory, and thus improve the sequence prediction.

In order to train these kinds of prediction model, it is very important to do standard scaling to our input. Therefore, we did standard scaling and after several tuning, we used a model with two hidden layer of LSTM cells and each with 32 cells in it. We originally set the input to be the past 10 games' performance of the team. However, the results showed that using past 10 games, we couldn't get effective prediction results. Since there are probably no pattern for the model to learn, it will end up prediction very close to the average of the input (Figure 9).

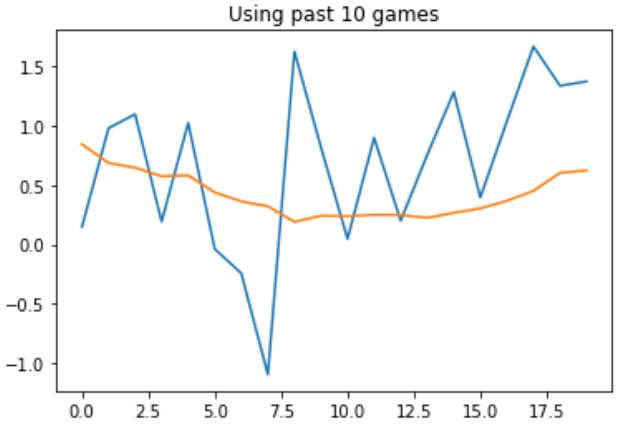

Figure 9: True score (blue) v.s predicted score (orange)

Therefore, after a bit of trial and error, we end up setting

the input to be the performance of past 5 games of the team. Figure 10 shows the mean square error (MSE) during our training. As shown, the mse only reduced to around 0.7, which is not a very small number. However, considering the team's performance is always changing a lot and it is in fact very hard to find a pattern, we did our best to minimize the MSE.

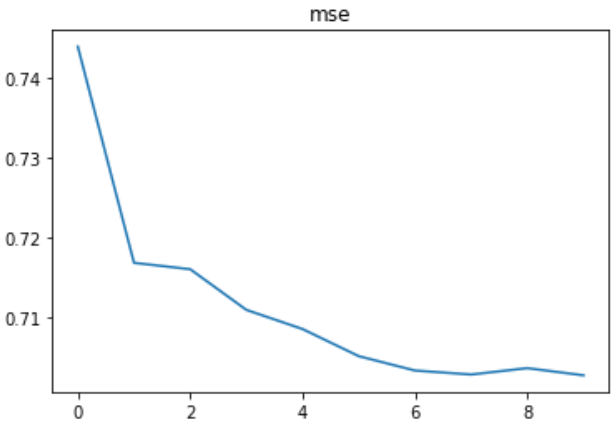

Figure 10: MSE of our LSTM model using 5 past games

We can also observe through Figure 11, which is the whole view of predicted data that after several tuning, we achieved a decent prediction model.

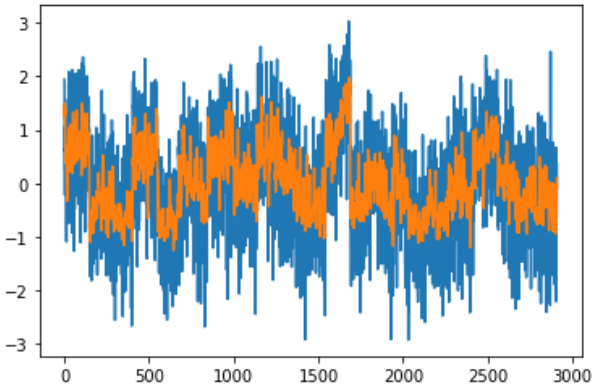

Figure 11: True score (blue) v.s predicted score (orange) using past 5 games performance after tuning

5.4. Match Result Prediction

For this section, we want to predict the results of matches. The first naïve approach is to use the team ids that are playing each other as the input and then classify them into three categories: win, draw and lose. The second approach is to use the player ids that are playing in the match to better represent a team's strength since the true determination of it are by players rather than team names. We used the most powerful classification models recently which are Random Forest, XGBoost, and MLP to classify them. However, as shown in Figure 12, the results show that even if we use these powerful models, if we split the data into training and testing, no matter using player id as input or team id as input, we can only get the accuracy of around 45%.

	Random Forest	XGBoost	MLP
Team (Split)	0.46	0.51	0.46
Player (Split)	0.44	0.47	0.38
Team	0.7	0.57	0.49
Player	0.98	0.81	0.92

Figure 12: Prediction accuracy of various classification methods with different input features

What's interesting is that we found that if we include all data to the training process, for using player's data we can get very high accuracies. We believe this happened because of several reasons: First, the result of the match simply based on team's or player's data doesn't really have observable indicators (see Figure 13). If there are really patterns that is predictable, it will be too easy to bet with money. Second, the data we use only consists of 4 seasons of data, which means for any two teams, they could only possibly play each other at most 8 times in the whole data, not to mention there are always 3 changing teams each season. Which means that if we split the data, we only have 6 games to train and 2 games to predict. If the results don't change a lot, it would be quite easy to predict. However, in English Premier League, the higher standing team always has a change to lose to lower standing teams, which means the data is too less considering this dynamically changing environment. Lastly, since each game there are 16 players playing, the player's id as a feature is actually a high dimensional data. Given this less data (at most 8 games each team), the input feature will be very sparse, which would easily cause overfitting when using these powerful models.

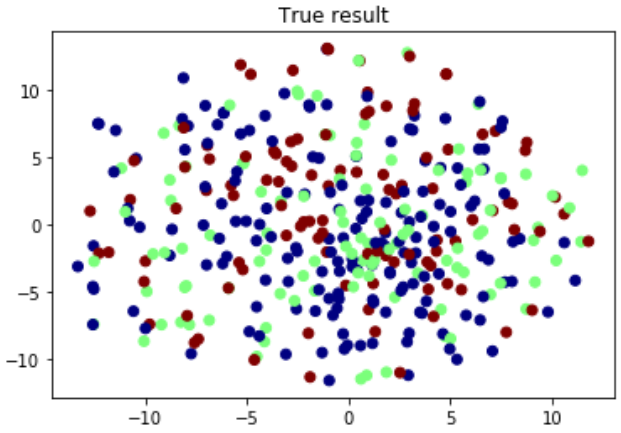

Figure 13: This figure shows that the match result data are extremly patternless

As a result, we propose a method which combines our previous team performance prediction work and the classification models we tested to solve the problem we encountered. The model is the combination of the LSTM

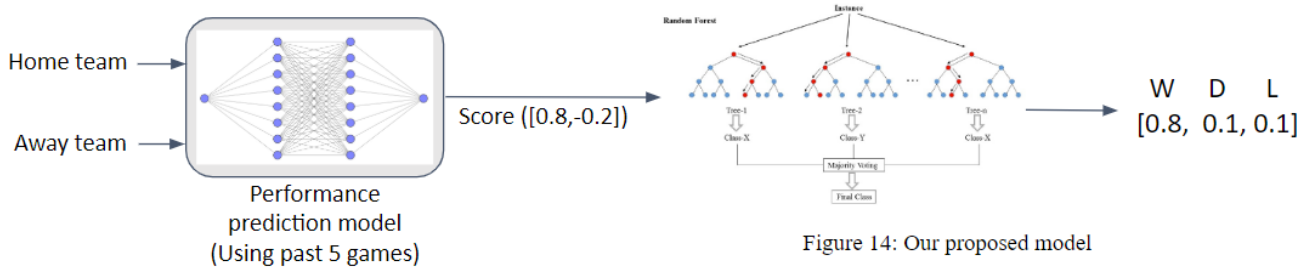

Figure 14: Our proposed model

model and Random Forest Classifier. The reason why we choose Random Forest is because the average accuracy achieved by the previous result (Figure 18 shows comparison).

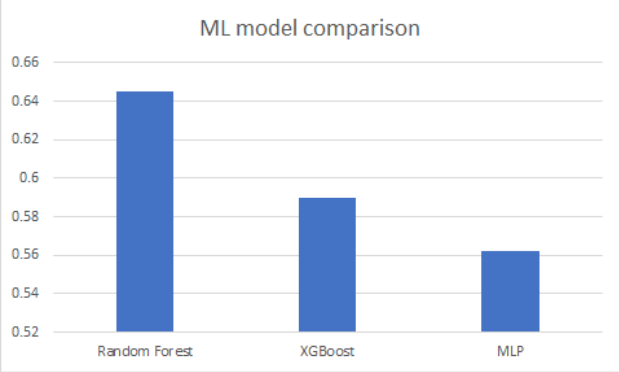

Figure 18: Model comparison

The main benefits of this model are: First, by predicting the team's performance first, we took consider of a team's recent performance. It is known that when a weaker team has been performing well, there's a chance that the team will carry this momentum to the next game and beat some stronger teams. Second, by using this combined model for classification, we solved the problem with lack of training data. As mentioned before, the data consists of four seasons which are totally 1520 games, and each team only met each other 8 times and by using team id as input, there will be 8 same data that have likely different labels. However, using this performance prediction output as the input of our random forest model gives us 2900 inputs of performance score and are more normally distributed data. The accuracy that our proposed model achieved was 65% (with splitting data).

5.5. Webpage Visualization

The data in BigQuery is not Interpretable to group managers. Thus, the data visualization is a must for our product. Our visualization web page process includes four parts: Django, Backend, Frontend, and Cloud server. The total system of our web application can be described as Figure 15 below.

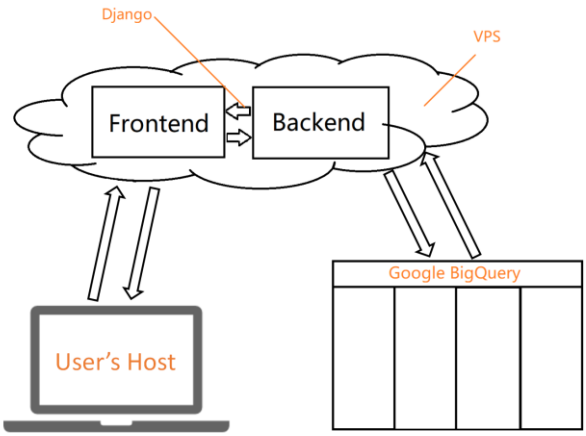

Figure 15: Web Application

Django [10] is an open source web application framework written in python. The framework mode of MTV is adopted, i.e. model M, view v and template t. It is easy to deploy in any hosts and servers. When using Google's BigQuery, no local database needed to place a web application. The Django framework can help us build our web easily. We can run the core python file "manage.py" to make our site active. Our remote server is a VPS from bandwagon host provider with an IP address 104.128.81.222. People can get access to our work via http://104.128.81.222:8000/dashboard.

Backend mainly contains 4 python files: setting.py, urls.py, view.py and wsgi.py. Some of them defines the query function to BigQuery and run the SQL commands in GCP. The backend scripts can bring the data from database to the frontend. The backend scripts focus on heavy computational tasks and solving logical problems such as queries.

Frontend is some JavaScript and html files. Our web application contains two pages: dashboard.html (Figure 16) and cluster.html (Figure 17). In dashboard.html, people can query any teams participated in the Premier league from 2014-2018, and we can show our prediction of the current season via a pop-up. If the pop-up shows "D", It is more likely for two teams to draw. The letter "W" means that the home team has a greater possibility to win over the away team. The letter L means lose.

We also gave out our precise prediction of Win, Draw and Lose by presenting it in SVG. People can clearly see the exact output of our prediction algorithm. To help have a

better understanding of the data, we also use the Query function of BigQuery to choose random 5 historical matches in 3 years.

It is easy to test our accuracy and the efficiency of our algorithm by using the current data of 2019's ranking. In 2019 premier league, the Manchester city took first place, the Liverpool took the second. So how did our prediction perform in the final? We can clearly see that the Manchester City is more likely to overtake the Liverpool if it is the home team (Win:60%, Draw:15%, Lose:24%). And it is hard to say if the Manchester City is the away team (Win:42%, Draw:16%, Lose:42%). That proved that our algorithm and web application does make sense.

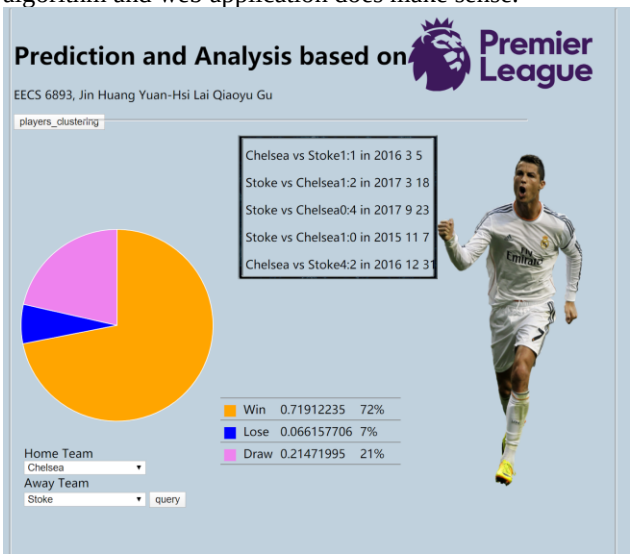

Figure 16: Visualizing Webpage for prediction

In clustering page, we clustered more than 1000 players into 10 clusters. Each cluster of players has a similar playstyle. It is easy to determine the label for each cluster by someone that is in favor of soccer, but the machine sometimes work better. The larger circles reflect a more influential player. When you move the mouse over the clustering graph, you can find the player's name, and you can then find the players with the same playstyle. Those circles are based on SVG elements as well.

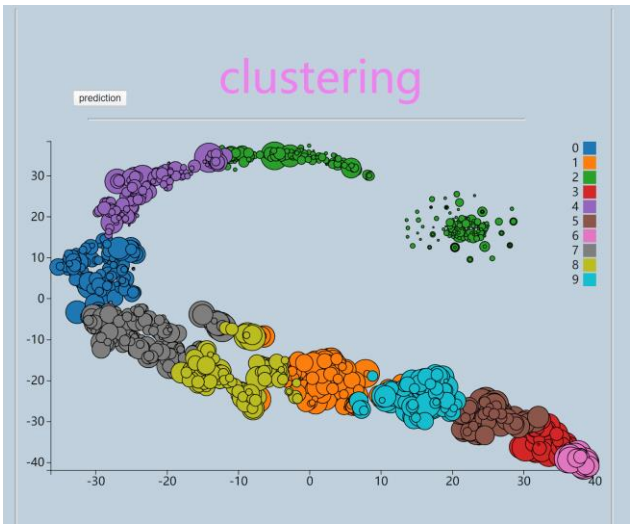

Figure 17: Visualizing Webpage for player clustering and value

6. System Overview

Figure 19 is what the overview of our system looks like. We use Google Cloud Platform (GCP) to store raw data and the preprocessed data. We also used Pyspark to process the csv data into RDD and do mapping and filtering for out feature extraction. For the prediction and classification models, we used several tools: Keras, Scikit-learn, XGboost. We used Keras to build the LSTM model for performance prediction, and also the MLP model for match result classification. Scikit-learn was used to do several data preprocessing including one hot encoding, testing data splitting, and standard scaler. We also utilized the Random Forest classifier in Scikit-learn. XGboost is also another classification model that we used and is included in the XGBoost Library.

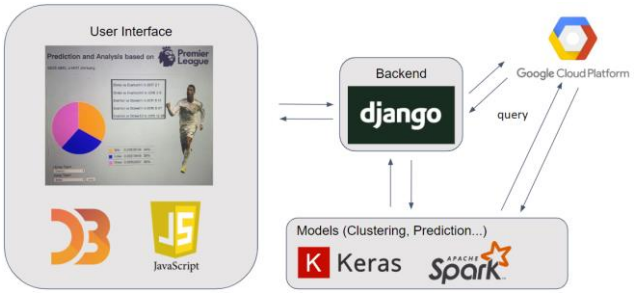

Figure 19: Overview of our system

For the webpage side, we used Django for backend. We wrote a Python program to do query from the Big Query in GCP and transfer the data through Django for webpage visualization. In the frontend part, D3js was used for big data visualization and we also used JavaScript for some webpage function.

7. Conclusion

In this paper, we designed several models to analyze the English Premier League data. Observing the results, we could see that there are still room for improvement. The main difficulty of our work is that the change in recent football league are huge. Every season, there are team leaving and joining, player leaving and joining, and good teams don't always win games. Therefore, the possible improvement of our work relies on more specific details in the data or possibly more useful data. Though we have only four seasons of data and we could have collected more, it is useless for data too long ago since the players and teams will be completely different. We tried our best on the current dataset and we realized that it is still hard to design a model that effectively predict the result for today's matches.

References

[1] Premier League Football News, Fixtures, Scores & Results *https://www.premierleague.com*

[2] English Premier League News, Stats, Scores – ESPN *https://www.espn.com/soccer/league/_/name/eng.1*

[3] Lars Magnus Hvattum, Halvard Arntzen (2010). Using ELO ratings for match result prediction in association football

[4] Farzin Owramipur, Parinaz Eskandarian, and Faezeh Sadat Mozneb (2013). Football Result Prediction with Bayesian Network in Spanish League-Barcelona Team

[5] L.J.P. van der Maaten and G.E. Hinton (2008). Visualizing High-Dimensional Data Using t-SNE

[6] English Premier League in-game match data. *https://www.kaggle.com/shubhmamp/english-premier-league-match-data*

[7] Vishal Morde, Venkat Anurag Setty (2019). XGBoost Algorithm: Long May She Reign!

[8] Will Koehrsen (2017). Random Forest Simple Explanation

[9] Multilayer Perceptron *https://xiaozhuanlan.com/topic/3750926184*

[10] Holovaty A, Kaplan-Moss J. (2009). The definitive guide to Django: Web development done right

[11] Ferraiolo J, Jun F, Jackson D. (2000). Scalable vector graphics (SVG) 1.0 specification

[12] Football Predictions. *FootballPredictions.com*

[13] Nazim Razali, Aida Mustapha, Faiz Ahmad Yatim, Ruhaya Ab Aziz (2017). Predicting Football Matches Results using Bayesian Networks for English Premier League (EPL)

Team Contribution

	Works	Percentage
Yuan-Hsi Lai (yl4305)	Model designing, Model training	40%
Jin Huang (jh4137)	Data processing, Webpage designing	30%
Qiaoyu Gu (qg2172)	Algorithm Designing, Video editing	30%