Predicting Football Results with Fixed-Odds Betting using Machine Learning Techniques

Yi Rong, Sichao Yang, Yan Wang

CS760 Course Project Report

University of Wisconsin – Madison

Department of Computer Science

Abstract

Predicting results of football matches is particularly challenging since the match itself has lots of uncertainty and that it is very difficult to encode detailed data about a match into a single model to run the prediction. In this paper, we present a method using neural network to predict the results (win, lose or tie) given the values of the pre-match fixed-odds betting as well as past match data. We also compare it with a simple decision tree as our baseline model, and a deeper neural networks classifier.

Introduction

Champions League is probably the most well-known club football competition and is being watched by over one billion football fans every year. Due to its popularity and prestige in the football world, we think it is an interesting challenge to try to predict the result and further using the result to bet on those matches.

For every individual match, there are three possible outcomes: win (the home team defeats the away team), tie, and lose (the away team defeats the home team). Before the match starts, various sports betting companies made their odds disclose for people to bet on. It is very difficult to predict the result of a match beforehand, since how those two teams perform depends largely on the situation at that specific time of the match, such as team members' selection, offensive and defensive skills discussed in their team meetings, impact of fans, etc. We would like to put as much of those knowledges into our model as possible, and we chose to use pre-match betting odds and past match data as our primary features.

In this paper, we will develop a machine learning model that can predict the match results of Champions League with some numeric confidence, and a software system that assists odds buyers on how much they should put into different betting options. We will measure the performance by both the accuracy of our outcome prediction, and the final value of how much one can earn (or lose) if he or she bet on every match with suggested value from our model.

Yi Rong: yrong9@wisc.edu
Sichao Yang: syang456@wisc.edu
Yan Wang: wang2264@wisc.edu

Related Articles

There are several papers related with betting prediction of football matches' result provide us some potential approaches to construct the models of prediction. One paper contributed by Goddard and Asimakopoulos (2004) utilized an ordered probit regression model on 10 years' data to forecast English league football match results. The regression in this paper shows, the significance of the match for end-of-season league outcomes, the involvement of the teams in cup competition and the geographical distance between the two team's home towns all contribute to the predicting model's performance. This result reminds us to emphases several factors including the performance of one team in home court and guest court and the past outcomes in the end-of -season and cup competition are very important in the procession of feature engineering. Also, this paper suggests a strategy of selecting end of season bets with a favorable expected return according to the model appears capable of generating a positive return. This result hints us to focus on more end-of-season bets when construct our own model to obtain the highest returns from match betting. Another conference paper contributed by Hubacek, Sourek, Zelezny (2018) discusses a new approach embedding method for predictive sport analytics. They compared the performance of the lifted relational neural network on team embedding (LRNN) with other several popular methods like RDN boost, State-of-the-art model. The RPS metric shows LRNN has better performance than other methods. This result provides us another potential approach of embedding learning.

There are several popular algorithms used in previous published research. Arabzad, Araghi and Ghofrani (2014) utilized the Artificial Neural Networks (ANNs) to predict the outcomes of the Iran Pro League 2013-2014 in one-week period. They designed a two-hidden layered neural net to predict the outcomes of fifteen teams. The P value of their result is very significant (<0.05). From this paper, we think ANN is one of proper algorithms for our study. Another paper written by Joseph, Fenton and Neil (2006) compared the performance of prediction from several common algorithms such as decision tree, Naive Bayes, Bayesian neural net, KNN based on the dataset of players' ability. From their results, we know the BNN has a relative higher accuracy compared with other algorithms. Therefore, the Bayesian neural net is also a vital algorithm in our study. At last, one master thesis from Pettersson and Nyquist (2017) provides us one approach of model to train the dataset. They chose Recurrent Neural Network (RNN) to train their data and predict the outcomes of football matches. By using proposed LSTM architectures in RNN, they got an acceptable accuracy of pre-match outcome's prediction, which is 0.4396, and then a high accuracy of football outcome in full time (game finished) for many-to-one strategy (98.63%) and many-to-many strategy (88.68%) in the full data sequence. We might take RNN as one of potential methods in the procession of model selection if there is enough time.

Data Collection

We used the dataset provided by Football-Data.co.uk. It is a public dataset that presents the historical results of most top soccer leagues in Europe and betting odds from different companies for each match. Although soccer experts will evaluate matches based on more detailed knowledge about the capability, style, and form of the 22 startup players in each team's line-up, we want to explore whether the overall statistics and the betting odds set up by online betting sites can capture the essential information for predicting the result of a match.

The data on this website is available for download in the form of individual CSV forms for each league and each year, for example E1.csv contains match data for English Football League (EFL) Championship in season 2018/2019.

We wrote a small script to download all the data and merged them into one single CSV file for the ease of reading it into our model. However, data in the recent decade from the English leagues are also singled out to be as they are both big enough and completer than average, and thus thought to be more predictable according to the existing data.

Feature Selection

Different csv file downloaded from the source above contains a different number of columns, due to difference in the data collection conventions and the involvement of betting companies in different leagues. Below is the rough arrangement in the original file:

Division	Date	Home Team	Away Team
Various Match Statistics	…	Betting Odds Data	…

Match Statistics contains the full-time results (result, home goals, away goals) and half-time results. We want to frame the match prediction task as a classification task that tries to predict the full-time results based on every relevant event captured by the data in the dataset available before the actual match, including betting odds for this match and the statistics and betting odds for all relevant matches (played by the home and away teams prior to the date of the match).

For the betting odds data, we are using all odds data available before the match, including several different forms of odds.

Triplets of betting odds for the full-time result (home win, draw, away win):

- B365H = Bet365 home win odds (as well as: tie, away win)
- BSH = Blue Square home win odds (as well as: tie, away win)
- BWH = Bet&Win home win odds (as well as: tie, away win)
- GBH = Gamebookers home win odds (as well as: tie, away win)
- IWH = Interwetten home win odds (as well as: tie, away win)
- LBH = Ladbrokes home win odds (as well as: tie, away win)
- PSH and PH = Pinnacle home win odds (as well as: tie, away win)
- SOH = Sporting Odds home win odds (as well as: tie, away win)
- SBH = Sportingbet home win odds (as well as: tie, away win)
- SJH = Stan James home win odds (as well as: tie, away win)
- SYH = Stanleybet home win odds (as well as: tie, away win)
- VCH = VC Bet home win odds (as well as: tie, away win)
- WHH = William Hill home win odds (as well as: tie, away win)

The total goals betting odds:

- BbMx>2.5 = Betbrain maximum over 2.5 goals (as well as: under 2.5 goals)

- BbAv>2.5 = Betbrain average over 2.5 goals (as well as: under 2.5 goals)
- GB>2.5 = Gamebookers over 2.5 goals (as well as: under 2.5 goals)
- B365>2.5 = Bet365 over 2.5 goals (as well as: under 2.5 goals)

The Asian handicap betting odds:

- BbMxAHH = Betbrain maximum Asian handicap home team odds (as well as:away team odds)
- BbAvAHH = Betbrain average Asian handicap home team odds (as well as:away team odds)
- GBAHH = Gamebookers Asian handicap home team odds (as well as:away team odds)
- LBAHH = Ladbrokes Asian handicap home team odds (as well as:away team odds)
- B365AHH = Bet365 Asian handicap home team odds (as well as:away team odds)

Closing odds (last odds before match starts):

- PSCH = Pinnacle closing home win odds (as well as: tie, away win)

We think the odds are a great indication for the final match results, due to the fact that the betting companies are essentially encouraging buyers to bet on one or more options which they later on will potentially give back more than what those buyers put in. In other words, in order for those betting companies not to lose money, they will try to calculate the betting odds ratio as carefully and risk-free as possible, which requires them to have a comprehensive understanding of the match before it takes place to figure out the best odds for them. This indicates that the domain knowledge of the match, which we are not able to acquire just through the match result, is very likely already encoded in the betting odds ratio they gave before the match.

However, betting odds are calculated by the experts to maximize the profit of companies rather than the prediction accuracy of match results. The preference of the buyers can cause a deviation in the odds. Therefore, important information about the strength, weakness, form, and tactics should be dug out from the performance of the team in the current season as well as the historical match results.

Since a match can be selected from different time periods in a season, the match results and statistics are averaged to avoid the interference of the number of matches already played.

It is harder to design a fair measure for the historical results, as every year a fixed number of teams are promoted to and delegated from each league, resulting in a constant change of team composition of the leagues, and thus an unstable number of matches between different teams in the past. For example, The Wolves was promoted to the Premier League in 2018, so it did not have a chance to play a league match with Man United in the 2017/18 season, but Liverpool played Man United in the same year. Also, the performance of the Wolves in the Championship (one level lower than the Premier League) cannot be directly compared with the performance of Man United in the Premier League.

After elaborating on this problem, for each match between team A and team B, we decide to calculate a weighted average of the numerical statistics and betting odds from six kinds of matches in each year of interest:

- A vs B: previous matches between A and B where A is the home team (as well as B vs A).
- A vs C: previous matches A played at home against all teams A and B both competed with in the same year (as well as C vs A, B vs C, C vs B).

Multiple results from the same year are averaged. The average results from different years we are interested in then averaged with an exponential decay according to the years. For example, if we think results from n_{year} ago matters for a match in year y, but a year should be weighted *(1-decay)* times higher than the year before, then the accumulated result vector r in terms of result r_t in year t should be

$$\sum_{t=0}^{n_{year}} r_{y-t} * (1 - decay)^t / Z$$

$$where\ Z = \sum_{t=0}^{n_{year}} (1 - decay)^t$$

The averaged results are then truncated into the complete vector of 897 features:

Betting Odds	In-Season Home Stats	In-Season Away Stats	A vs B	B vs A	

Machine Learning Techniques

In order to achieve better prediction performance, we tried many learning algorithms to determine which gives the overall most accurate predictions. This task is a very typical classification task, so we can just reuse lots of existing approaches to quickly evaluate the performance. We also took into consideration previous attempts in the Related Articles section, and chose the following algorithms:

- Simple Decision Tree based purely on pre-match odds
- Small Artificial Neural Networks
- Deep Neural Network Classifier

Decision Tree on Betting Odds

We use a naïve approach as our baseline model. As stated above, before every match starts, we get to know all the pre-match betting odds values. We do a weighted average on odds from different betting company and get a triple of final odds, which for convenience we will call it final win odds O_w, final tie odds O_t, and final lose odds O_l. This

model can be viewed as an ensemble of single decision trees that tries to predict the result solely on one company's odds value. We are in some way using all values available to prevent potential mistakes of a special betting company. Our algorithm basically predicts which ever final odds is the lowest. So, for example if we have a triple of odds (1.2, 1.1, 3.4), then our simple decision tree would give us "tie".

3-Layer Densely Connected Neural Network

```
model = Sequential()
model.add(BatchNormalization())

# hidden layer 1
model.add(Dense(n1, input_dim=691, kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Activation('relu'))
model.add(Dropout(keep_prob))

# hidden layer 2
model.add(Dense(n2, kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Activation('relu'))
# model.add(Dropout(keep_prob))

# # hidden layer 3
model.add(Dense(n3, kernel_initializer='he_normal'))
model.add(BatchNormalization())
model.add(Activation('relu'))
# model.add(Dropout(keep_prob))

# output layer
model.add(Dense(3, activation='softmax', kernel_initializer='glorot_normal'))
model.compile(optimizer=optimizer,
              loss='categorical_crossentropy',
              metrics=['accuracy'])
```

A three layer densely connected neural network is used to classify the match results.

Since the features include different match statistics and different calculation methods of betting odds, the input features have very diverse norms, which are normalized at the beginning.

Then the normalized input is fed into three consecutive dense layers with the RELU activation function, He initialization, and batch normalization before activation. The technique of dropout is applied to early layers to reduce overfitting. Without regularization, the model can induce a variance as high as abut 10%, while setting the hyperparameter *keep_prob* to 0.8 reduced the variance to less than 1%.

A softmax function transforms the output of the last dense layer into the final output. Categorical crossentropy is used as the lost function. Xavier initialization is used for the weights between the third and output layer. The model worked the best used the Adam optimizer with a learning rate of 0.01 and no decay.

DNN Auto classifier

We use this model mainly to try out whether we could get slightly higher accuracy by adding more layers to the network.

This model is set up to be a three-class classifier that contains 10 layers of hidden units with Proximal Adagrad as its optimizer and L1 regularization method. It uses the exact same feature input as our 3-layer model.

```
for mode in ['home', 'away']:
    feature_columns = feature_columns + [
        tf.feature_column.numeric_column(key='{}-wins'.format(mode)),
        tf.feature_column.numeric_column(key='{}-draws'.format(mode)),
        tf.feature_column.numeric_column(key='{}-losses'.format(mode)),
        tf.feature_column.numeric_column(key='{}-goals'.format(mode)),
        tf.feature_column.numeric_column(key='{}-opposition-goals'.format(mode)),
        tf.feature_column.numeric_column(key='{}-shots'.format(mode)),
        tf.feature_column.numeric_column(key='{}-shots-on-target'.format(mode)),
        tf.feature_column.numeric_column(key='{}-opposition-shots'.format(mode)),
        tf.feature_column.numeric_column(key='{}-opposition-shots-on-target'.format(mode)),
    ]
model = tf.estimator.DNNClassifier(
    model_dir='model/',
    hidden_units=[10],
    feature_columns=feature_columns,
    n_classes=3,
    label_vocabulary=['H', 'D', 'A'],
    optimizer=tf.train.ProximalAdagradOptimizer(
        learning_rate=0.1,
        l1_regularization_strength=0.001
    ))
with open('training-log.csv', 'w') as stream:
    csvwriter = csv.writer(stream)

    for i in range(0, 400):
        model.train(input_fn=train_input_fn, steps=100)
        evaluation_result = model.evaluate(input_fn=test_input_fn)
        predictions = list(model.predict(input_fn=test_input_fn))
        prediction_result = betting.test_betting_stategy(predictions, test_features, test_labels)
        csvwriter.writerow([(i + 1) * 100, evaluation_result['accuracy'],
    evaluation_result['average_loss'], prediction_result['performance']])
```

Evaluation

We use two metrics, accuracy and principal left after simulated betting, to evaluate our models. We define the accuracy as percentage of predicted matches that the model correctly classifies between win, tie and lose.

$$\text{Accuracy} = \frac{\#\ of\ instance\ correctly\ predicted}{\#\ of\ total\ test\ instances}$$

For our second measure, we assume that a virtual user of our software system starts with a principal of x dollars. Throughout the test set, our model will make prediction for every test instance and guide that virtual user to bet according to our confidence. And we measure the overall performance of a model by comparing the final amount of principal left, or hopefully how much additional money the model could help that user make just through pre-match betting.

Results & Analysis

For our 3-layer ANNs, we tried to tune the hyperparameters n_{year} and *decay over the* English Football League dataset. Each setting is tested for ten times. The average testing accuracies are as follows

n_{year} \ *decay*	.8	.7	.5	.0
1	52.90%	52.97%	52.32%	52.32%
2	51.55%	50.90%	52.84%	51.77%
3	51.80%	50.91%	51.81%	51.16%
4	51.55%	52.84%	50.91%	52.07%

we ran the model on one of our laptops, and after training over the entire English Football League dataset, we managed to achieve an accuracy of 52.9%, as opposed to the theoretical 33% with random guessing. This result is on par with the best result we've seen on other papers doing similar prediction for football results. For the TensorFlow DNN classifier, we ran the model with the same training

environment and data. Although we had a much deeper neural nets, we ended up with an accuracy of 51%, merely better than tossing a coin, while the training set accuracy was over 84%. Our baseline ensemble of decision trees gives us the final accuracy of 49%

As for the final amount of money left, we had only 40% of the original principal left for our simple ANNs classifier, and had a negative value for both the DNN classifier and baseline decision tree ensemble, which shouldn't be considered given our assumption.

It is obvious that a 10-layers fully connected DNN classifier would overfit a dataset this small, and that's why we had much higher accuracy for training than actual testing. For a simpler 3-layer neural nets, we can achieve what most of the current approaches do.

Conclusion

In the future, instead of training a model to predict the match results, we can modify the loss function and output of the model to let it choose to spend money on which result or simply keep the money. In this way, it may make better use of the betting odds and predict the expected reward ratio of betting on each game. Still, the less predictive general match statistics should be replaced by more detailed lineup statistics including player values, abilities, injuries, recent ratings, etc.

References

Arabzad, S.M., Araghi, M.E., Sadi-Nezhad, S., & Ghofrani, N. 2014. Football Match Results Prediction Using Artificial Neural Networks; The Case of Iran Pro League. *Journal of Applied Research on Industrial Engineering*. Vol.1, No.3 (2014) 159-179

Goddard, J; and Asimakopoulos, I. 2004. Forecasting Football Results and the Efficiency of Fixed-odds Betting. *Journal of Forecasting* 23, 51-66 (2004).

Hubacek, O; Sourek, G; and Zelezny, F. 2018. Lifted Relational Team Embeddings for Predictive Sports Analytics. *ILP Up-and-Coming / Short Papers*. Czech Technical University, Prague, Czech Republic.

Joseph, A.; Fenton, N.E; and Neil, M. 2006. Predicting Football Results using Bayesian Nets and Other Machine Learning Techniques. *Knowledge-Based Systems 19 (2006) 544-553.*

Pettersson, D.; and Nyquist, R. 2017. Football Match Prediction using Deep Learning. Master's Thesis in Computer Science- algorithms, languages and logic, Department of Computer Science, Chalmers University of Technology, Gothenburg, Sweden.