Boğaziçi University

Sports Forecasting

Use of Alternative Information Sources for Predicting Scores Outcome

Emre ATILGAN

Mert SARIKAYA

IE 492 - Project

Asst. Prof. Mustafa Gökçe BAYDOĞAN

04.01.2019

Abstract

Sports forecasting is important for the sports fans, the bookmakers, team managers, the media and the growing number of the bettors who bet on online platforms. In this project, soccer will be the primary focus due to its market size. Betting odds are one of the main sources of information in terms of predicting the soccer match outcome. These odds are inherently projecting probabilistic beliefs of bookmakers. In order to remove the bookmakers' margin and calculate probabilities from these odds, two types of normalization methods are utilized, namely basic normalization and Shin's method. It will be shown that Shin's method works better than basic normalization in terms of prediction accuracy. Ranked Probability Score will be used for measuring the goodness of predictions. It is concluded by using Friedman and post-hoc Nemenyi tests that bookmakers differ from each other in terms of predicting match outcomes. This project proposes an ordered random forest model for predicting match results by using Shin's normalized probabilities, mean change rates of odds and proportion of insider trading. Finally, it will be observed that this model performs as well as the best bookmakers' class and simple strategies like betting on only draw can make money in the long run.

Özet

Spor tahminlemesi, spor hayranları, bahisçiler, takım yöneticileri, medya ve çevrimiçi platformlarda bahis yapan artan sayılardaki bahisçiler için önemlidir. Bu projede futbol, pazar büyüklüğü nedeniyle ana odak noktası olacaktır. Bahis oranları, futbol maçı sonucunu öngörme açısından ana bilgi kaynaklarından biridir. Bu bahis oranları doğal olarak bahisçilerin olasılıksal inançlarını yansıtıyor. Bahisçilerin marjını ortadan kaldırmak ve olasılıkları bu oranlardan hesaplamak için, temel normalleştirme ve Shin'in yöntemi olmak üzere iki tür normalleştirme yöntemi kullanılır. Shin'in yönteminin, tahmin doğruluğu açısından temel normalleştirmeden daha iyi çalıştığı gösterilecektir. Sıralı Olasılık Skoru, tahminlerin iyiliğini ölçmek için kullanılacaktır. Friedman ve post-hoc Nemenyi testleri kullanılarak bahisçilerin maç sonuçlarını tahmin etme açısından birbirlerinden farklı olduğu sonucuna varıldı. Bu proje, Shin'in normalize edilmiş olasılıklarını, ortalama oran değişim hızlarını ve içerden öğrenenlerin dağılımını kullanarak maç sonuçlarını tahmin etmek için sıralı Random Forest modeli önermektedir. Son olarak, bu modelin en iyi bahisçiler sınıfı kadar iyi performans gösterdiği ve sadece beraberliğe bahis yapmak gibi basit stratejilerin uzun vadede para kazanabileceği gözlemlenecektir.

Table of Contents

1. Introduction

    a. A summary of the IE problem handled

Betting is the action of gambling money on the outcome of a race, game, or other unpredictable event. In 2015, the online gambling market had a volume of almost 38 billion U.S. dollars, and the largest portion – 48 percent – was held by sports betting. When it comes to type of the sports, football betting has the largest volume of sport betting. There are mainly two players in this industry namely bookmakers and bettors. Bookmakers are the organizations that running the whole operation of betting such as announcing the odds, collecting the bets, and paying offs to the winners. Second critical players in this market are bettors who are trying to guess the outcome of the match and betting accordingly to get financial gains.

There are many different possibilities when it comes to football that one can bet on. In this project, type of "singles" bet is the focus area. A single bet in the football field is win, draw or loss (1; X; 2) from a home team perspective. A general single bet from any bookmaker can be similar to (1.67; 5.00; 3.50) that means bettors can have back 1.67 times of the money they bet on home win and so on. These odds are reflecting the probabilistic beliefs of the bookmakers on match results. Society does not know surely how a bookmaker determines the odds, but it is commonly believed that they have some "expertise" in this area and some heuristic models working at the backstage. By the definition of odd in the probability theory, one can easily calculate that bookmakers give 1/1.67 = 0.59 chance for home team win and similarly 0.20, 0.28 chances to draw and away win, respectively. In this point, it is assumed that the sum of probabilities should be equal to 1. However, in the above betting odds, the sum is 1.07 which indicates that there is a 7% margin that bookmakers have added. This margin, called booksum, is the extra amount of chance added by bookmakers to increase their expected gains no matter what the actual result of game will be. There is no common rule of thumb that how much a bookmaker can or should put the margin, but it is a trade-off that company has to decide. Lower

margin may result in the more expected liabilities while keeping the margin too high may affect that bettors may choose to play the same bet on another bookmaker. It can be said that there is a competition between bookmakers to gain more bettors by decreasing the margins. On the other hand, reducing margin in betting is quite risky for bookmakers which do not know the real probabilities and use only the estimates. The most important way to get rid of this trade-off is improvement of forecasting accuracy of matches beforehand. By doing so, bookmakers can add relatively safer margins without losing bettors to other bookmakers. In this project, the main problem is that how a bookmaker's predictions can be improved by using statistical models and alternative sources of information. Also, it is investigated to see whether bookmakers are efficient enough forecasting the probabilities and whether there is any statistically significant difference between them in terms of accuracy of forecasts.

b. A short overview of identification, analysis and solution methodologies

According to some empirical research, using betting odds of the main sources of information is most efficient way in terms of predicting the soccer match outcome (Štrumbelj, 2014). However, putting the odds itself in models can affect the results of predictions since the odds contains the margins of bookmakers. First step to getting a more reliable source of information from betting odds is to eliminate the effects of margin by using some normalization techniques.

The first normalization approach is the simple basic normalization. In basic normalization, the probabilities get by taking inverses of odds are simply be divided by their sum. For example, from the odds (1.67; 5.00; 3.50), it can be easily getting that probabilities are (0.59, 0.20, 0.28) that reflects the probabilistic belief of bookmaker for a given match. When these numbers are divided by their sum of 1.07, the probabilities are captured as 0.55, 0.19, and

0.26 respectively. Last figures are the normalized probabilities from betting odds by basic normalization.

Second way to get normalized probabilities is the Shin's method (Shin, 1993). It is the game-theoretical approach to get the final probabilities. Basically, Shin's method is based on assumption that bookmakers are considering the ratio of insider traders while announcing their odds. Insiders are the ones assumed to know the result of match before it is actually played. In football, some match-fixing can be source of such superior information. Shin's method is considering these insider traders ratio and does reverse-engineering and fixed-point iterations to get the probabilistic beliefs of bookmaker as it is done in basic normalization.

After getting the more reliable input variable in our model by using basic and Shin normalization methods, it is needed to get proper metric to evaluate different predictions. In our problem, our response - the match result of home win, draw, and away win - is the ordered categorical variable since the nature of football match. In any football match, the draw outcome is in the middle of the other outcomes since it is not possible to move home-win state to away-win state by moving the draw state first. Therefore, while making predictions about the match it is important to distribute the remaining probabilities correctly. To consider these effects, Ranked Probability Score (RPS) (Epstein, 1969) is used as a metric. This metric gives the difference between the cumulative probabilities and the cumulative actual outcome to reflect the ordinality of the variable measured.

At the beginning of the model development process of the project, it is needed to see whether there is a significantly difference between the use of basic or Shin normalization and if the bookmakers are different from each other in terms of success of predictions. In this aim, it is calculated the RPS scores of all the matches for each bookmaker by using Basic and Shin normalizations. Since RPS values violates the normality assumption, a non-parametric comparison approach (Friedman and post-hoc Nemenyi tests) is used. After applied the tests it

can be said that (I) Shin normalizations are superior basic normalization for almost all bookmakers and (II) there are some significance groups of bookmakers that are different that others in terms of prediction accuracy. The results of these comparisons are used as base of our statistical models. As a first feature set of input variable, it is used shin normalizations of better bookmakers.

For further feature engineering, it is calculated the insider trader ratio of a bookmaker for a given match as a second feature set. It inherently reflects the probabilistic belief and companies' extra precautions for a risky match. Moreover, bookmakers alter their odds from their first announcement to just before start of match. Reasons of these changes may be an injury of an important player or some balance adjustments for profitability. In our data set, these changes of betting odds are captured and, as final feature set, it is calculated the mean change rates of odds for all bookmakers and for all odd types of a given match.

In the model development process, four fundamental models are tested - Decision Tree, Generalized Linear Models, Gradient Boosting, and Random Forest. It is considered the power of feature selection properties of these model during the selection of candidate models since our feature sets are large. Moreover, because our response variable is categorical and ordered, the ordered versions of these models (if any) are compared to get the final model.

c. Improvements

The most improvement of this project is to gain a comparative analysis of different solution methods of prediction problem of ordinal categorical response variable. Four main classification model are compared by their versions of handling ordinality in the terms of using 3 different scenarios of feature sets. The accuracy of prediction of different models are different and the strongest models are the ordinal forest random forest with the use of shin probabilities,

insider ratios, and mean change rate of odds and decision tree model with only a feature set of shin probabilities.

Another improvement of the project is that ordinal random forest is the third best model among the 25 bookmakers in the 2018-2019 season of English Premier League. In terms of statistical differences, the model of this project is in the top group of more accurate bookmakers and statistically better than 5 bookmakers.

Finally, with the simple betting strategy of betting the draw outcome accordingly the ordinal random forest model, the model is gained 140 Turkish liras when it is started with 400 liras at the beginning of 2017-2018 season (about 35% of return).

d. Brief summary of conclusions

Forecasting the football match results is challenging area of statistical learning world. The companies of in this industry are highly competitive and this lead to need of better statistical tools of predictions.

In this project, by comparing the different classification models and using the strongest one with the simple betting strategy it can be concluded that there are lots of improvements and market is still inefficient. Ordinal random forest model with the use of Shin probabilities, insider traders' ratio and mean change rates of odds is the promising model for the future works. It can be improved with use of other bet types of odds or some domain-specific knowledge.

e. Contents of the report

In this report, sport forecasting - use of alternative information sources for predicting scores outcome project will be presented. It will be started with the problem definition, requirements, and limitations section and followed by analysis for solution methodology. After presented the detailed solution approach, the alternative solution approaches and their

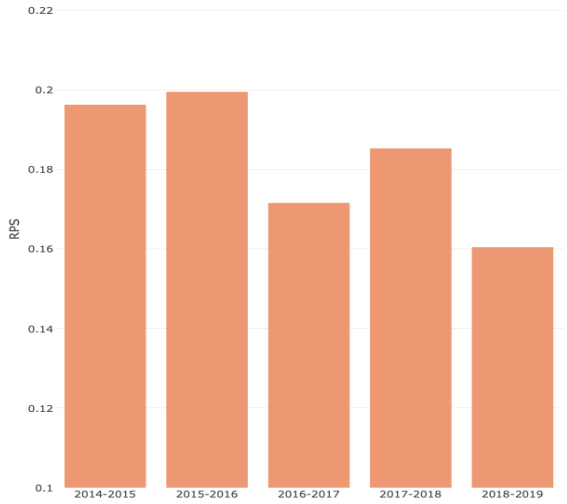
comparisons are will be discussed. Finally, suggestions for successful implementation of the solution and conclusions and discussions of the project will be presented.

2. Problem Definition

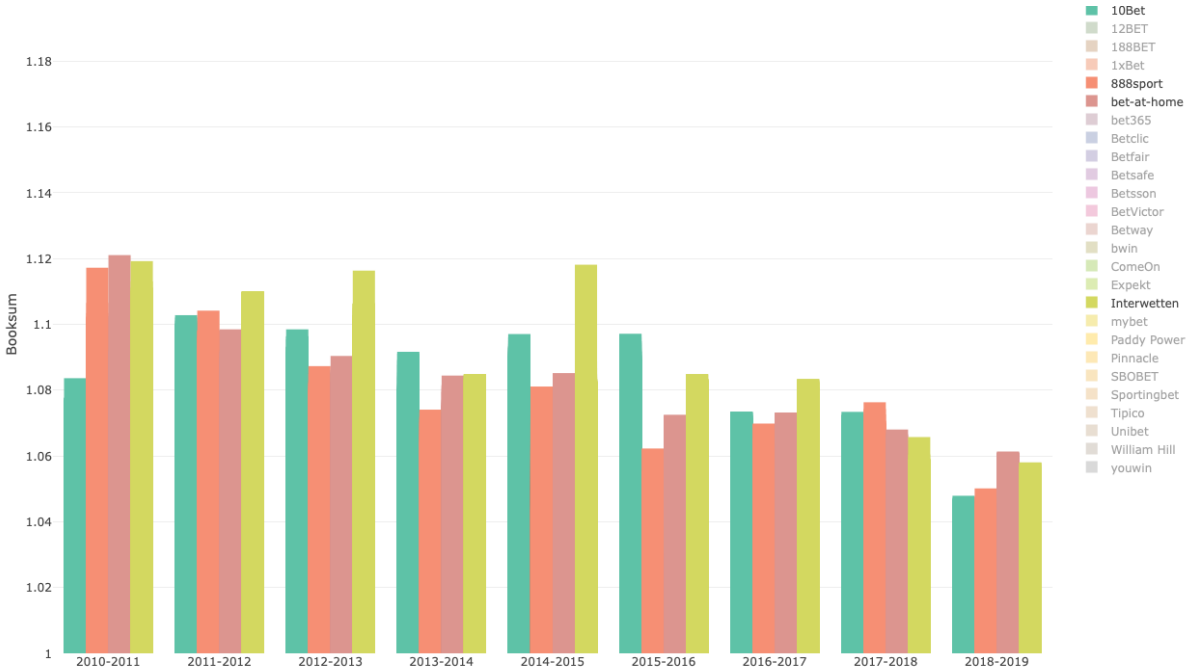
a. What seems to be the problem?

Bookmakers add their margin while announcing their odds. This margin, called booksum, is the extra amount of chance added by bookmakers to increase their expected gains no matter what the actual result of game will be. There is no common rule of thumb that how much a bookmaker can or should put the margin, but it is a trade-off that company has to decide. Lower margin may result in the more expected liabilities while keeping the margin too high may affect that bettors may choose to play the same bet on another bookmaker. It can be seen that average margins throughout a season are decreasing seasonally due to high competition in the sector. In order to decrease the amount of the margin, bookmakers need to become better in predicting.

Bookmakers improve their goodness of predictions by time, their RPS values are getting lower consistently for each season. They might invest in their algorithms or tipster might have learnt their business better. In order to exist in this competitive environment, they need to keep getting better. This is one of the reasons why each bookmaker has a motivation to improve its model.



Average RPS scores for bookmaker 1xBet



Average booksums for some bookmakers for each season

Whereas they are not good enough for predicting some parts of the matches. For example, they are insufficient for predicting draw result. After analysing data, it can be seen that when home and away team has similar winning probability, actual draw ratio is more than what it was predicted to be. So, there is surely a room for improvement in this area.

b. What is done to understand the causes of the problem?

Bookmakers are not only trying to predict the result of the game accurately, but they are also trying to optimize their revenues game theoretically, regardless of the result of the matches. After analysing the data by grouping matches into the bins according to differences between home and away teams' implied probabilities, it can be observed that bookmakers are not predicting. To be more clear, when considering all the matches whose home winning implied probabilities minus away team winning implied probabilities is between -0.02 and 0.02, it can be seen that bookmakers assign approximately 30 percent probability on average, whereas these matches were resulted in draw approximately 36 percent. This 6 percent difference is where improvement needs to be done. In other point of view, if this area (where home and away team's winning probabilities are similar) can be utilised our model works better than the bookmaker's predictions.

c. Needs and requirements of the system

Tests are applied on previous data but in order to forecast the upcoming matches, constantly updating the data is a must. In order to be consonant with our model, missing data must be as small as possible and each change in odds can be observed in our data too. The only thing that matters is having valid and up to date data.

In order to test our model in real life, one must be eligible for using bookmaker's betting system, legal requirements must be satisfied. In this paper, it is assumed that betting on single match is allowed. These constraints must be fulfilled before testing the model on the real life.

d. Limitations and constraints

It is not easy to reduce the uncontrollable and unpredictable factors such as chance, the weather, the ground, bad referee decisions in the game. And since goals are relatively rare and

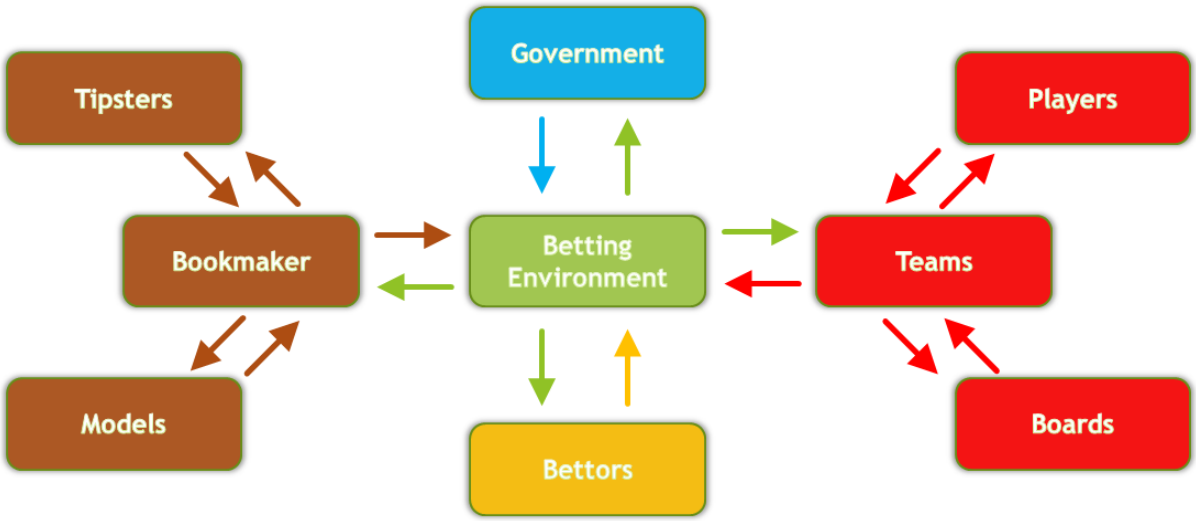
the margin of victory for the winning team is relatively low for most matches, this game-changing circumstances makes prediction harder. Poor referee calls, penalties, injuries or red cards may occur at any time and the they are not occurring as a result of superior or inferior play by one team, but are due to difficult-to-capture events. So, predicting the football outcomes is never going to be easy.

The data is basically in the form of odds and their changes by the time and actual result of the matches. Whereas the starting eleven, strengths of the players in each team might be important for the outcome. It might be safely assumed to see the effects of this subsequent information hidden in the odd values of the dataset, but it's still a limitation.

e. Data gathered and used in the identification phase

There are two different data sets, first one is matches and the other is odd details. Matches file has necessary information directly related with matches, like home and away team, final score and match time. Whereas odd details data set has the specific odds announced by bookmakers for given time. This file has matchId, one can join these 2 data set on this matchId column. And odd details has bettype, odd type, bookmaker, date, totalHandicap and odd. In our project only "1x2" bettype was utilized, remaining are removed from the data. odd type stands for home draw or away odd. There are 27 bookmakers and 8 seasons, these seasons can be calculated from date column. There are more than 6 million of rows in odd details table and more than 3100 matches. These numbers are increasing each week also.

f. Context diagram of the handled design problem



g. Performance criteria and potential improvements

Main performance criteria will be the average Ranked Probability Score. Final models, or actual bookmakers, can be measured in terms of their profitability but, Constantinou claims that making money in the long run does not necessarily means that predicting better (Constantinou, 2015). One can predict worse but with some different strategies he/she can make money even in the long run, and one can predict better than the bookmakers but without making portfolio optimization or without deciding how to bet on he/she can lose money in the long run. For this reason, ranked probability score will be the conclusive metric.

After comparing our final model with current bookmakers, it can be concluded that we are in the best class of bookmakers in terms of ranked probability score. If all bookmakers were analysed with respect to their RPS scores and post-hoc Nemenyi test was applied, one can observe there are some groups of bookmakers who are in the best class, which means one cannot say that a bookmaker is significantly better than any other bookmakers in that specific group, and who are in the worst class, meaning each bookmaker in that group is statistically worse than the bookmakers in the other classes. Even now our final model is in the best class of

bookmakers and we can make money in the long run with some betting strategies, our final model can be improved, and it becomes significantly better than all current bookmakers.

In order to do that we can utilize additional datasets in order to decrease the level of the randomness, to control what is controllable in a sense. Weather forecasts, matches in different leagues, referee and fan information can be added to our model as future improvement.

3. Analysis for Solution Methodology

a. Literature overview

There are a lot of studies on soccer forecasting area, due to market size. The European football market alone is projected to exceed € 25 billion in 2016/17 season. (Deloitte, 2016) and global sports gambling market is estimated to worth up to $ 3 trillion, with football betting representing 65% of this figure (Daily Mail, 2015). Some of these papers that want to predict single league by using all the leagues available to analyse. In Dolores, Past relevant academic studies typically focus on a single league or tournament, with predictions derived using various predictive modelling techniques.

Erik Štrumbelj, in his paper "on Determining Probability Forecasts from Betting Odds", claims that "the empirical evidence suggests that betting odds are the most accurate source of sports forecasts and odds-based probability forecasts have been shown to be better than, or at least as good as, statistical models using sports-related input variables, expert tipsters and (aggregated) lay predictions" (Erik Štrumbelj).

Bookmakers' underlying probabilistic beliefs might be indicated by inverse odds. But when these inverse odds are summed up, they exceed 1 so these excess percent must be accounted. Some papers utilize basic normalization, but Shin proposed a different normalization approach for this kind of problems. In his paper, Shin concludes that the activity of insider traders influences the outcome in the betting market in a significant way. (Shin, 1993)

His model was based on the assumption that bookmakers quote odds which maximize their expected profit in the presence of uninformed bettors and a known proportion of insider traders. And he proposed a simple formula to approximate the insider proportion, and then one can simply find normalized probabilities by solving with fixed point iteration.

Ranked Probability Score is the most common metric in measuring soccer predictions, most recent papers use RPS as their metric. (see Štrumbelj, Berrar and Constantinou). The ranked probability score is the sum of the differences between the cumulative forecast probability and the cumulative outcome probability. In order to calculate this score, home draw and away must be ordered as 1,2,3 accordingly. This follows the natural order since a match cannot turn from home to away without draw.

b. Alternative solution approaches

It is believed that odds are at least as good as football related variables like previous matches between teams and score history of given week. But we can add different variables by using these odds. First one is insider proportion. Shin claimed that these insiders directly has effect on odds, so we believed that including this information in our model might positively affect our results. And changes in odds is something that is not well utilized in recent studies. Changes in odds happens when there is a vital news about match or bookmaker may simply revise odds in favour of its potential revenue. But these changes do not happen all the times, so their structures are different. So, there are many types of using these changes, in our project they are used as calculating change rate and taking the average of these rates. Therefore, we use 3 different input variables, namely Shin normalized probabilities, insider proportion and mean change rate of odds.

One of the main questions that we want to answer in this project is whether bookmakers differ in terms of prediction quality or not. In order to answer this question, bookmakers need

to be compared by their RPS values. After finding average values of RPS scores, since this metric violates the normality assumption one cannot simply apply hypothesis testing. After applying Friedman and post-hoc Nemenyi test, answer of this main question was found.

c. Assumptions

We are only dealing with the odd values. Main assumption in here is that these values inherently have the important information related with the teams. Any important information is somehow projected in announced odds. Even this cannot be proven since no one knows how bookmakers determine the odds, significant evidence can be reached concluding this is the case.

Another assumption is related with the real-world application. Our tests are applied on the data by assuming single match betting, which means one either win or lose for each match separately. But this is not always the case, some bookmakers require combining 2 or 3 matches at least, does not let betting on single matches. But combination complicates the problem, for the sake of easiness, it is assumed that single match betting is allowed.

d. Brief overview of the selected approaches

Our approaches can be divided basically from 2 different points, first creation of data set and second model type. By creation of data set, it is meant that which variables to be used and by selection of model type we meant choosing the appropriate machine learning model to our data.

Linear model, random forest, decision tree and gradient boosting are the models that we tried. The reason of choosing linear model is faster than most of the models, they are pretty simple and fast yet still gives adequate result. Decision trees are again simple and fast and the way they work (grouping and assigning their probabilistic beliefs within group accordingly) is very intuitive and first thing that comes to one's mind probably, so we added this type of model

to our basket. Random forests are being used by most of the studies and issues related with these concepts. They are very powerful, yet parameter selection and its running time makes the computation harder. Gradient boosting and random forest are good for feature extraction and selection. In order to do this feature selection in linear models, Lasso penalization introduced to general linear models.

Shin's normalized probabilities are base variables, they exist in every model. They project the probabilistic beliefs of bookmakers. But if we could find something that makes RPS score lower, that might work with our model too, that might correct implied probabilities. To be able to do that, insider proportions and changes in odds were introduced to our models. But they were not used directly, each model was tested on 3 different combination of these data sets, namely using only Shin probabilities, Shin probabilities with insider proportion, and 3 of them combined.

e. IE skills/tools/techniques/methods to be integrated to implement the proposed methodology

This project mainly utilises Statistics and Data Mining courses in IE curriculum. Data is very huge, analysing and creating a model from it requires some statistics background. While cleaning and preparing the data, and then pre-processing it we used some properties of normal distributions and basic statistics information. While testing the bookmaker difference we first tried to use hypothesis testing but since our metric violates one of its assumption, we learnt a new approach and used it in our project. We utilised finally different machine learning models that makes possible to train and test on our data on hand.

4. Development of Alternative Solutions

    a. Detailed technical description of the design process for generating meaningful alternatives

Initial steps of this project are cleaning and preparing the data sets. To prepare matches data sets, date column must be converted from UNIX to timestamp, anytime package provides this conversion. And there are some matches whose score is null since they were not played yet, these matches were separated to next_matches data set. Now, each row in matches set has score information. It is required to understand who's the winner or was it drawn from a string like "3:0". In order to do that, a parser function was implemented, it compares parts before colon and after colon, and assigns home and team scores and finally returns "odd1" if home team won, "oddX" if match was drawn and "odd2" if home team lost. For over / under, similar steps followed. If total score is greater than or equal to 3, it is considered as over, else it is under. So, if a match results in "3:2" it's both "odd1" and "over", whereas if match ends is "1:1" it's "oddX" and "under". Week and season information are necessary in order to calculate seasonal and / or weekly analysis. Odds are generally announced as weekly, so weekly analyses are required. This information can be calculated by manipulating date columns. Codes can be seen in the appendix.

To prepare odd details data set (from now on, it will be referenced as details) date column should be ordered at first. Since odds can be changed before the matches, we want to utilise these changes accordingly. So, when it changed and how much it changed are vital questions, therefore data set must be ordered with respect to date column. One of the bookmakers, Betfair Exchange, are announcing odds by using no algorithm or model but just creating a free trade environment. Everyone can sell odds whatever they like even if they can find a bettor who agrees on that odd. So, their values are ruining our model, therefore we remove

the rows of this bookmaker. Then its date column converted just like matches set. Since we are only focusing on 1x2 bettype our data was subsetted accordingly.

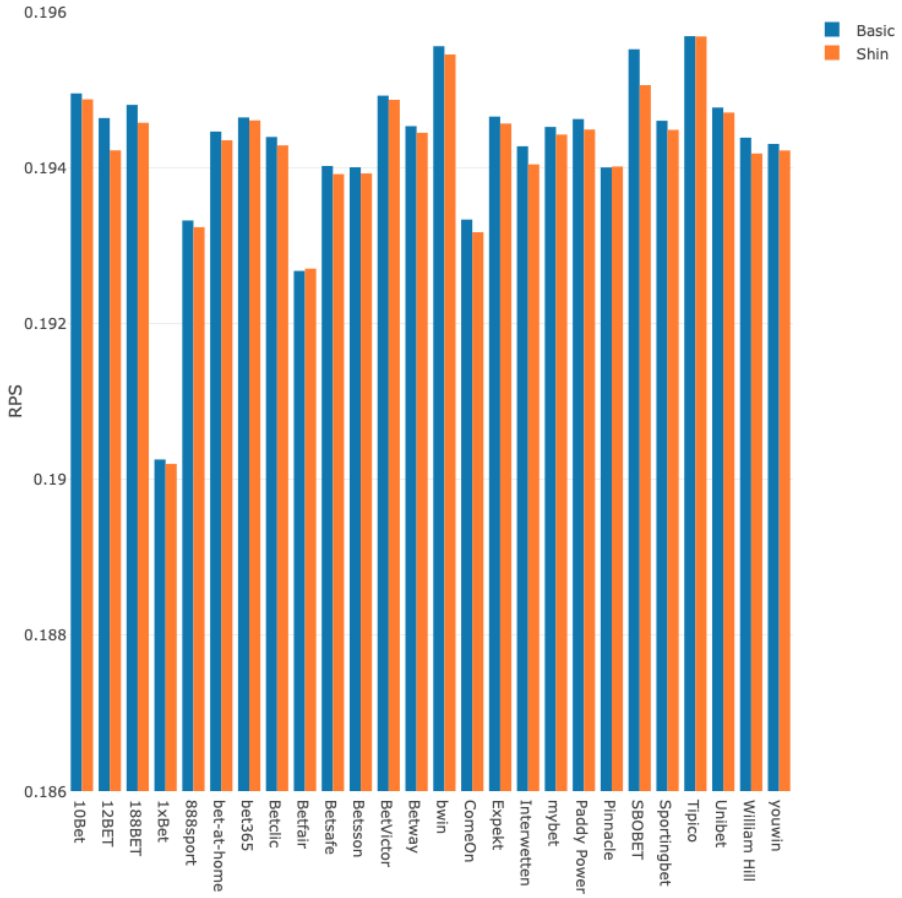
After creating a copy of details, observations were shifted and differences for each bookmakers and matches were calculated. This information will be used for calculating average difference rate. Rate is equal to change in odds divided by change in time (rounded to nearest hour). After calculating these rates, for each bookmaker, matchId, odd type tuple mean, sd and max values of these rates were calculated.

Changes were used, they can be implemented in our model, but we still did not find the probabilistic beliefs of bookmakers. So, from now on, different odds are not required, only the last entries of each bookmaker, matchId, odd type tuple will be used with the assumption of bookmakers correct their odds during the process, their last odds are most accurate projectors of their probabilistic beliefs. Last data set refers last entries for each bookmaker, matchId, odd type for details data set.

To convert last odds into the final probabilistic beliefs of bookmakers, one needs to take the inverses of odds first. With these inverse odds, we need to normalize these odds. One can simply make basic normalization, but it was shown that Shin was better than basic normalization, so we will only go for Shin's method. (Shin, 1993) To be able to apply Shin's method a function is defined, it can be investigated in detail in the appendix. Insider proportion is calculated also during this normalization process.

$$p_i = \frac{\sqrt{z^2+4(1-z)\frac{{\pi_i}^2}{\beta}}-z}{2(1-z)} \qquad z = \frac{\sum_{i=1}^{n}\sqrt{z^2+4(1-z)\frac{{\pi_i}^2}{\beta}}-2}{n-2}$$

For this calculation, p_i stands for normalized probabilities, z is insider proportion, π_is implied probabilities to be normalized, is booksum which is 1 + margin, and n is number of possible outcomes which is 3 in our 1x2 bet type case.



Basic and Shin's method normalization comparison with their average RPS

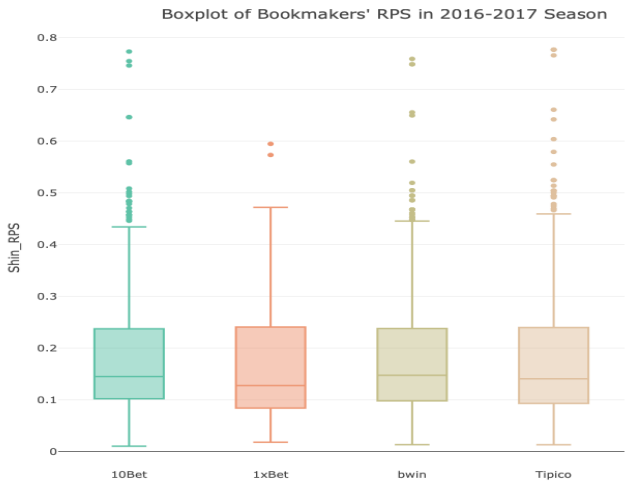
Up to this point, we have the required information of matches, and for each match, we have changes and last odds of each bookmaker and odd type tuple. And we also know Shin's normalized implied probabilities of each bookmakers for all the matches. And also results of the matches were known. Therefore, Ranked Probability Scores of each matches and bookmakers can be calculated.

Ranked Probability Score is a metric widely used in the area to calculate the goodness of a soccer prediction. It is widely used since it captures the ordinal relation that exists in the nature of football. It calculates the cumulative difference between predictions and observations. As can be seen in the chart below, smaller the RPS is the better predictions.

P(Home)	P(Draw)	P(Away)	Winner	Cum. Pred.	Cum. Obs.	RPS
1	0	0	Home	(1,1,1)	(1,1,1)	0
0.6	0.3	0.1	Home	(0.6,0.9,1)	(1,1,1)	0.0850
0.6	0.1	0.3	Home	(0.6,0.7,1)	(1,1,1)	0.1250
0.35	0.3	0.35	Draw	(0.35,0.65,1)	(0,1,1)	0.1225
0.6	0.3	0.1	Draw	(0.6,0.9,1)	(0,1,1)	0.1850

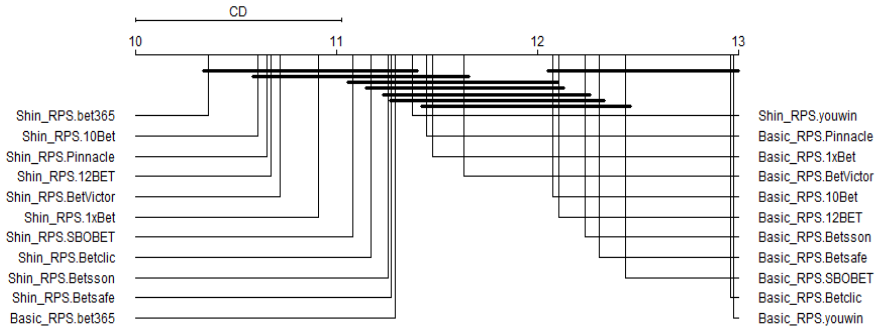
Ranked Probability Scores of Hypothetical Matches

To compare the bookmakers, first RPS values of each match needs to be calculated for all different bookmakers. After finding average RPS values overall or seasonally, the results are like the plot below. They have similar average RPS values, but they are all different from each other. In order to find whether this difference significant or not, we need to apply some statistical tests.



Average RPS Values of Selected Bookmakers

Since RPS metric violates nonlinearity assumption, we cannot apply hypothesis testing directly. Therefore, post-hoc Nemenyi tests are introduced. Results of Nemenyi's can be seen in plots below. It is proved that there exists a difference between bookmakers in terms of their prediction accuracy.



Post-Hoc Nemenyi Test

After creating data sets by merging previously calculated information like insider proportion and changes in odds, we can start creating our models. Our models will be applied on 3 different combinations, shin normalized probabilities only (we will call this A), shin normalized probabilities with insider proportion (B) and shin normalized probabilities with insider proportion and mean change in odds rate (C). Our candidate models were decision trees, random forest, gradient boosting, multinomial regression. Pros and cons of these models are as follows.

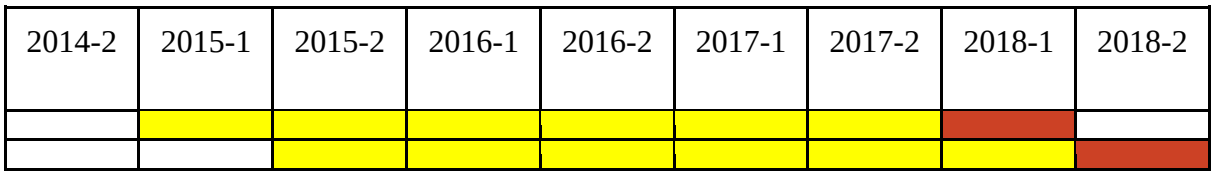
	PROs	CONs
Multinomial Logistic Regression	Feature selection via penalty parameter	Loss of information
Random Forest	Captures nonlinearity	Computational Complexity
Gradient Boosting	Better benchmark results	Many Tuning Parameters
Decision Tree	Simple and handles missing data	Ignores rest of the selection

Pros and Cons of Models

After introducing these models (for detailed information, see appendix), it is required to have fine tuning. In order to measure the goodness of cross validation, we introduced RPS as a cross validation metric. For random forest in caret package, one can only manipulate mtry parameter, so we tried to fine tune mtry. For lasso penalized multinomial logistic regression, lambda is the important parameter, so we played with that. For decision tree, caret allows only changing complexity parameter, but for gradient boosting it is required to play with 4 different parameters, namely interaction.depth, n.trees, shrinkage, n.minobsinnode. Selection of these parameters can be seen in source code in appendix.

But there are some limitations about these models and our data set. Some bookmakers did not announce odds for some matches, and most of our models cannot work with missing data. So, in order to decrease the missing value ratio, we kept the 15 bookmakers with the less missing value ratio. And since predictions of 3-4 years before was not good enough and there

was no high correlation between today's match and matches 4 years ago, we used sliding window approach. In order to test on first half of the 2018, train data should be started from the first half of 2015. This can be visually seen in the chart below.



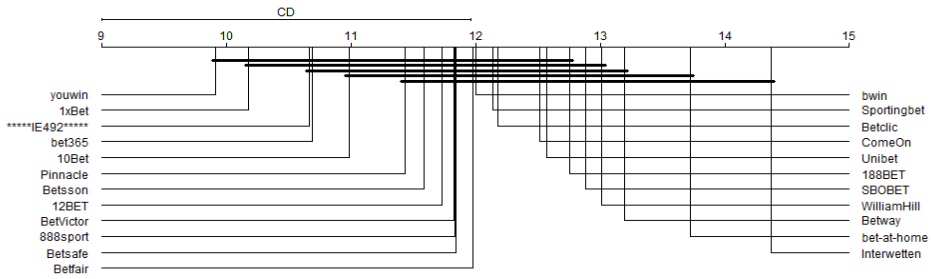
Sliding window (yellow represents train data, red represents test)

After applying all these models to 2018-2019 data by training our model from 2015 to 2017, we get the RPS results below. Since decision trees and linear models are hard to work on ordinal outputs, they are only utilised as non-ordered. From now on ordered random forest will be final decision since it gives the best result when all features are used in the model.

Average RPS	2018-2019					
	Decision Tree	Glmnet	Gradient Boosting		Random Forest	
Features			NON ORDERED	ORDERED	NON ORDERED	ORDERED
A	0.1704	0.1847	0.1721	0.1722	0.2281	0.1947
B	0.1704	0.2699	0.1765	0.1767	0.2172	0.1919
C	0.1851	0.2106	0.1724	0.1730	0.2051	0.1719

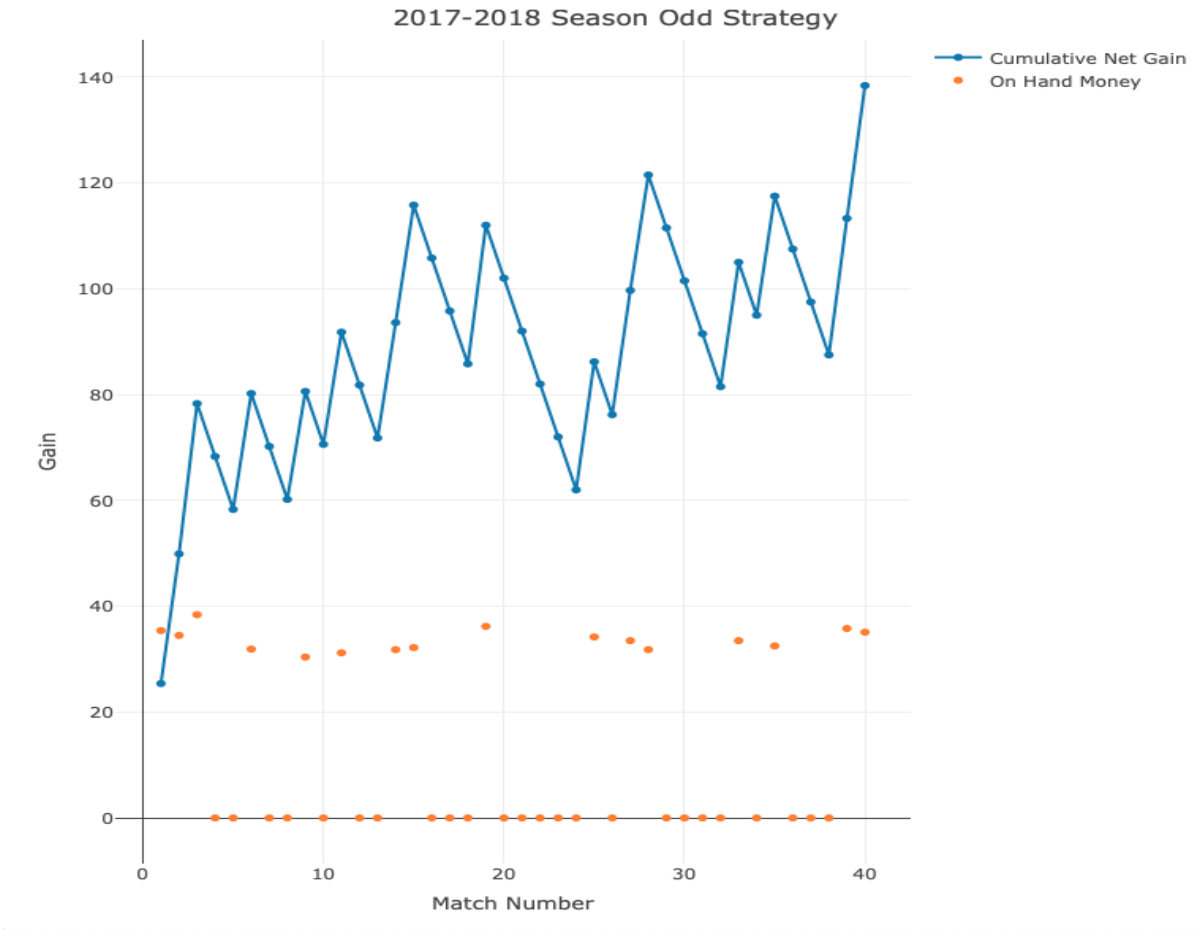
Average RPS values of models, tested on 2018-2019 season

If this ordered random forest model was applied in Nemenyi test, the result will be as follows. Our model is in the third place but there is no significant difference between our model and the best model, so we are in the best class of bookmakers.



Nemenyi test for 2018-2019 season, comparison of our model

After deciding on final model and completing hyper parameter selection, now our model is ready to be tested. As mentioned in literature (Mirza, 2016) bookmakers cannot predict draw probability good enough when winning probabilities of home and away team are similar. With this assumption, if we put $ 10 on the matches which we believe draw is most favourite outcome of the match, this plot shows the seasonally cumulative gain of our model. Odds are coming from the best bookmakers, youwin, by assuming this will be the gain for the worst-case scenario. Even if the matches we played on are not that much (40 for a season) this is because draw is harder to be favourite option in football, our model can make money in the long run. This simple strategy beats the bookmaker in 2017-2018 season.



Simple Betting Strategy for 2017-2018 Season

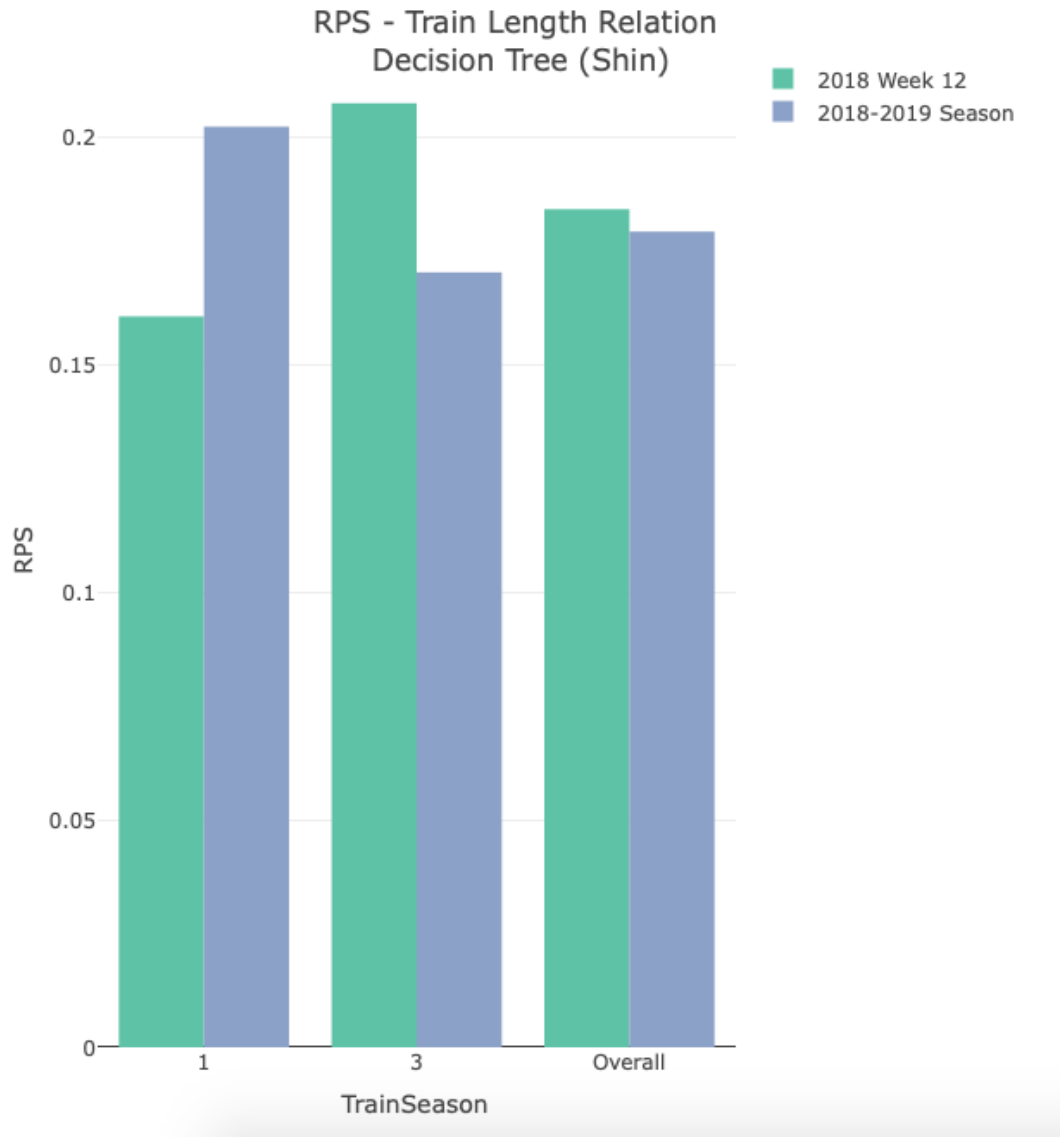
5. Comparison of Alternatives and Recommendation

   a. Numerical studies or evaluation procedure

In this project, four main model type and 3 feature set combinations were tried to predict the match outcomes. Four main models were linear model, decision tree, random forest, and gradient boosting. Among these models, the versions of them dealing with ordered categorical response variable were also computed and analysed.

Choosing the right train period were challenging due to some limitations in the data set and in the nature of football itself. Earlier seasons (before 2016-2017 season) have lots of missing values since some of bookmakers were not available in these years. When the nature of football is taking into consideration, using the whole available data has also negative effects

since teams and condition of league itself has changing significantly. To overcome these obstacles, sliding window approach has been implemented. To predict the result of matches of the next week and the next season, two train period has been tested - 3 years and 1 year before the start day of test period. It can be seen that using only data of last year has better RPS values when the aim is to predict the next week matches while using 3-years-long train period is better when predicting the next season matches.



Different RPS of Train Period Alternatives

After deciding to length of train period, the code has run for the 36 models for the matches of 2018-2019 seasons. Average RPS values of these models give us the result that ordinal random forest with combination of all feature sets is promising model to use. Another surprising result is that the decision tree algorithm with only Shin probabilities worked well even its simplicity.

b. Proposing a promising solution along with justification in terms of predefined criteria

After analysing different models and getting the average RPS values, it is decided to select the ordinal random forest with the three feature sets is the main promising solution among the alternatives. Even though decision tree algorithm worked well, it groups and assigns the same probabilities within group that will not very helpful in the future stages of portfolio optimization etc.

b.1. Limitations and Sensitivity of Solution

Even though the proposed solution is worked very well, it has some limitations to implement in the long run. Main limitation is that data has to be complete and up-to-date when it is wanted to predict the next week matches. Our mean change rates of odds feature set requires the changing data that has to be collected at least one time at a day.

After constructing the fit of the model, it may also require dynamic tuning approach to work accurately. Since it is the prediction model, the error of our predictions should be monitored and if there is a “out-of-control” situations model should be tuned again. When viewed from this aspect, the proposed solution is the sensitive to time.

c. Further assessment of the recommended solution

Due to nature of football, there are and there will always be some risky and surprise results. However, the proposed solution of ordinal random forest model is the competitive and promising solution in fulfilling the requirements of problem by statistical appropriateness. The model is consistent with the betting system requirements and limitations of data or football matches since it uses the betting odds as a base of feature sets.

Another advantage of the solution is that the market has a huge size of available data since the betting of football is most popular type of betting market as presented earlier. Therefore, the solution can be implemented and sustainable in terms of data requirements and data management.

Only disadvantage of the proposed solution is that data should be up-to-date almost all the time between matches. One of most powerful features is the one captures the odd changes that may reflect the football industry news and recent performances of the teams themselves. Therefore, to have a robust and accurate solution, the betting environment should be monitored closely, and data should be collected accordingly.

6. Suggestions for a Successful Implementation

a. How can the design be implemented?

The proposed solution should be implemented in the powerful computer system to reduce the run time of the code. The up-to-date data should be available to more accurate results and the errors of model should be monitored in case of recent need for tuning the parameters of model again.

b. Can the design be integrated with the overall system?

For the betting strategy, it is tried the simple betting to draw odds has been tested for 2017-2018 season. In this testing, it is assumed that one can play for only single game. In our country, it is not possible to play less than 3 matches, therefore, the proposed model can be used in the countries that allows to play single matches.

c. How often should any solution obtained be revised?

In the comparison of model alternatives, the model has been tuned (revised) at the beginning of the season and in the middle of the season. It is reasonable to keep track of the changes closely to have more accurate predictions. However, it is a trade-off to tune the model for every week since there are lots of tuning parameters. Therefore, when considered the breaks, transfer periods of teams and number of matches in one season, it is appropriate to tune the models twice a year.

7. Conclusions and Discussion

a. Summary of IE tools, techniques, methods used

In this project, the statistical background of the IE curriculum is highly used to get a proper solution. The tools learned in Data Mining classes and the advices of professor was the base sources of the project. Since the data is huge and the nature of problem is highly complex, the new methods and skills has been learned from the online sources of internet and from the literature itself. Regression techniques, statistical models, feature engineering processes has been implemented in the R Studio that has been learned in the core statistical classes of the IE curriculum.

b. Summary of the merits and significance of the solution

Predicting the result of football matches is highly important and popular one in the forecasting problems. since the market size of football betting is huge there are lots of companies (in our project number was 25) completing each other to have greater pie. In this crowded market, the prediction quality is the main determinant to which bookmakers will survive for long times. As discussed earlier, the market has still inefficiencies which leads to lots of recent academic researches and papers about this topic.

Getting a competitive and promising solution to predict the outcomes of matches is highly important for some reasons. First, the model can be improved with additional variety of data of betting types. Also, the prediction model of football matches can be implemented in the other sports types or even in the financial stock market prizes (Štrumbelj, 2014). Easiness of constructing and testing of a statistical prediction model in the football area may and will help the general forecasting problems in near future.

c. Economic, environmental, ethical and societal impacts of the solution

With the use of recommended solution, bookmakers will manage to correct their probabilistic beliefs about match. By doing so, they will gain competitive advantage over other bookmakers by reducing the effects of adjusting the margin. These will lead to profits on the long run for a company and also power to dominance the predictions market.

In the general point of view, this prediction project of ordered categorical response variable problems may result in advancement of similar type of classification projects such as weather forecast, stock market prizes etc. More accurate forecast in these type of real-world application problems may result in greater well-being of populations.

REFERENCES

1. Štrumbelj, E. (2014). On determining probability forecasts from betting odds. International Journal of Forecasting, 30, 934–943.
2. Brier, G. W. (1950). Verification of forecasts expressed in terms of probability. Monthly Weather Review, 75, 1–3.
3. Constantinou, A.C. (2018). Dolores: a model that predicts football match outcomes from all over the world. Machine Learning, Special Issue on Machine Learning for Soccer, 1-27.
4. Shin, H. S. (1993). Measuring the incidence of insider trading in a market for state-contingent claims. The Economic Journal, 103(420), 1141–1153.
5. Marek, P. (2018). Bookmakers' Efficiency in English Football Leagues. 36th International Conference Mathematical Methods in Economics, West Bohemia, Czech Republic, 2014, Researchgate Publishing
6. Deloitte. (2016). Annual Review of Football Finance 2016. Deloitte. Retrieved April 19, 2017 from https://www2.deloitte.com/uk/en/pages/sports-business-group/articles/annual-review-of-football-finance.html.

APPENDIX

R CODE FOR PROJECT

project.R

```
### clears the environment
rm(list = ls())

if (grepl("mert", toString(getwd()))){
  setwd("/Users/mertsarikaya/bitirme/")
}
if (grepl("Hp", toString(getwd()))) {
  setwd("C:/Users/Hp/Desktop/Bitirme/bitirme")
}

library(readr)
library(graphics)
library(data.table)
library(verification)
library(glmnet)
library(TunePareto)
library(anytime)
library(plotly)
library(stats)
library(PMCMR)
library(caret)
library(e1071)
library(rpart)
library(gbm)
library(plyr)
library(ordinalForest)
```

```
##### FUNCTIONS TO BE USED

################################################
# implementation of ranked probability score
# functions in this file:
# 1 - calculate_rps(home, draw, away, actual)
# 2 - calculate_rps2(over, under, actual)
################################################
source("rps.R")

################################################
# converting odd1, oddX, odd2 to 1,2,3 and viceversa
# 1 - convert(arr)
################################################
source("converter.R")

################################################
# read and prepare dataframes
# 1 - details (matchId, bookmaker, oddtype, odd)
# 2 - matches (matchId, score, home, away, date, over_under, winner, season)
# 3 - first (matchId, bookmaker, oddtype, odd)
# 4 - last (matchId, bookmaker, oddtype, odd)
# 5 - next_matches (matchId, score, home, away, date)
# 6 - details_change
# functions
# 1 - inverse
# 2 - over_under
# 3 - season_calc
# 4 - set_directory
# 5 - winner
################################################
source("get_dataframes.R")
```

```
source("comparison.R")

### converting odds to basic and shin probabilities, gives insiders
source("convert_odds.R")

### changing odds
source("changing_odds.R")


################  NOT FOR MODELS ##############
### calculate RPS for all matches using Basic and Shin probs
# lastrps <<<< required for models
source("calculate_rps.R")
### calculate average RPS for all bookmakers using Basic and Shin probs
#source("bookmaker_comparison.R")
### statistical tests
#source("statistical_tests.R")
#basic_vs_shin <- basic_vs_shin(lastrps)
#bookie_friedman <- bookmaker_comp_friedman(lastrps, bookiesToKeep)
#nemenyi_test_outputs <- bookmaker_comp_plot(lastrps,bookiesToKeep)
################################################


###################################
# Features:
# 1 - Shin Probabilities  df = last[matchId,bookmaker,oddtype,shin_prob])
# 2 - Changing in Odds    df = changes[....])
# 3 - Insider Traders     df = insider[matchId,bookmaker,z]
###################################

### reshaping features to create train_features
```

```
source("reshape.R")

### Creating training and test data together

shin <- lastrps[,-c("Shin_RPS")]
shin_insider <- merge(shin, insider, by = c("matchId", "bookmaker"))
shin_changes <- merge(last, details_change[,c("matchId", "bookmaker", "oddtype", "avg",
"winner")],by = c("matchId", "bookmaker", "oddtype"))
shin_changes <- reshape(shin_changes, idvar = c("matchId", "bookmaker", "winner"),
timevar = c("oddtype"), direction = "wide")
shin_changes_insider <- merge(shin_changes, insider, by = c("matchId", "bookmaker"))

#source("variable_importance.R")

source("train_models.R")

for (k in list(shin, shin_insider, shin_changes_insider)){
   for (i in c("random_forest", "decision_tree", "gradient_boosting", "glmnet")){
     model_new <- models(matches_df = matches[week = 42][season == '2018-2019'],
                details_df = k,
                model_type = i, is_ordered = FALSE)
   }
   # for (i in c("random_forest", "gradient_boosting", "vglm")){
   #   model_new <- models(matches_df = matches[week <= 42][season == '2018-2019'],
   #                       details_df = k,
   #                       model_type = i, is_ordered = TRUE)
   # }
   for (i in c("random_forest", "decision_tree", "gradient_boosting", "glmnet")){
     model_new <- models(matches_df = matches[week > 42][season == '2018-2019'],
                       details_df = k,
                       model_type = i, is_ordered = FALSE)
   }
   # for (i in c("random_forest", "gradient_boosting", "vglm")){
```

```
  #   model_new <- models(matches_df = matches[week > 42][season == '2018-2019'],
  #                       details_df = k,
  #                       model_type = i, is_ordered = TRUE)
  # }

 #2017-2018
  # for (i in c("random_forest", "decision_tree", "gradient_boosting", "glmnet")){
  #   model_new <- models(matches_df = matches[season == '2017-2018'],
  #               details_df = k,
  #               model_type = i, is_ordered = FALSE)
  # }
  # for (i in c("random_forest", "gradient_boosting")){
  #   model_new <- models(matches_df = matches[season == '2017-2018'],
  #                       details_df = k,
  #                       model_type = i, is_ordered = TRUE)
  # }
}

# for (k in list(shin, shin_insider, shin_changes_insider)){
#   for (n in noquote(unique(matches[season == "2018-2019"]$week))){
#     for (i in c("random_forest", "decision_tree", "gradient_boosting", "glmnet")){
#       model_new <- models(matches_df = matches[week == 48][week == n][season ==
'2018-2019'],
#                       details_df = k,
#                       model_type = i, is_ordered = FALSE)
#     }
#     for (i in c("random_forest", "gradient_boosting", "vglm")){
#       model_new <- models(matches_df = matches[week == 48][week == n][season ==
'2018-2019'],
#                       details_df = k,
#                       model_type = i, is_ordered = TRUE)
#     }
#   }
```

```
#   for (n in noquote(unique(matches[season == "2017-2018"]$week))){
#     for (i in c("random_forest", "decision_tree", "gradient_boosting", "glmnet")){
#       model_new <- models(matches_df = matches[week == n][season == '2017-2018'],
#                           details_df = k,
#                           model_type = i, is_ordered = FALSE)
#     }
#     for (i in c("random_forest", "gradient_boosting")){
#       model_new <- models(matches_df = matches[week == n][season == '2017-2018'],
#                           details_df = k,
#                           model_type = i, is_ordered = TRUE)
#     }
#   }
# }
```

get_dataframes.R

```
### sets directory easily
# 1 - set_directory(name)
# name can be "code", "data"
source("set_directory.R")

### implementation of converting match results from string to {over, under, 1, X, 2} types of outcome
# functions in this file:
# 1 - winner(score)
# 2 - over_under(score)
# 3 - inverse(odd)
source("match_scores.R")

### converting dates to seasons
# functions in this file:
# 1 - season_calc(date)
source("season_calculator.R")
```

```
set_directory("data")

#read raw data
matches <- read_rds("df9b1196-e3cf-4cc7-9159-f236fe738215_matches.rds")
details <- read_rds("df9b1196-e3cf-4cc7-9159-f236fe738215_odd_details.rds")

#prepare matches
matches <- data.table(matches)[, c("matchId", "score", "home", "away", "date"), with = FALSE]
matches <- unique(matches)
matches[,date:=anydate(date)]

next_matches <- matches[is.na(score)]
matches <- matches[!is.na(score)]

matches$over_under <- matches[, over_under(score), by = 1:nrow(matches)]$V1
matches$winner <- matches[, winner(score), by = 1:nrow(matches)]$V1
matches$week <- matches[, strftime(date-1, format = "%V"), by = 1:nrow(matches)]$V1
matches$season <- matches[, season_calc(date), by = 1:nrow(matches)]$V1

next_matches$week <- next_matches[, strftime(date-1, format = "%V"), by = 1:nrow(next_matches)]$V1
next_matches$season <- next_matches[, season_calc(date), by = 1:nrow(next_matches)]$V1

#prepare details and details_change
details <- details[order(date)]
details_change <- details
details <- data.table(details)[, c("matchId", "bookmaker", "date", "betType", "oddtype", "odd", "totalhandicap"), with = FALSE]
details <- details[bookmaker != 'Betfair Exchange']
details[,time:=anytime(date)]
```

```
#details_otherbets <- details[betType != "1x2"]

details <- details[betType == '1x2']

details[, c("totalhandicap" , "betType", "date", "time") := NULL]



#details[matchId == "zZ6f59Ue"][bookmaker == "bwin"][oddtype == "odd1"]

### nice example of changing data

details_change <- data.table(details_change)[, c("matchId", "bookmaker", "betType",
"oddtype", "date", "odd"), with = FALSE]

details_change <- details_change[bookmaker != 'Betfair Exchange']

details_change[,time:=anytime(date)]


details_change <- details_change[betType == '1x2']

details_change[, c("betType", "date") := NULL]


cols <- c("odd","time")

anscols <- paste("lead", cols, sep="_")

details_change[, (anscols) := shift(.SD, 1, "lead"), .SDcols=cols, by = c("matchId",
"bookmaker", "oddtype")]

details_change <- details_change[complete.cases(details_change)]

#details_change[matchId == "zZ6f59Ue"][bookmaker == "bwin"][oddtype == "odd1"]

details_change[, odd_diff := (odd - lead_odd)/odd , by = c("matchId", "bookmaker",
"oddtype")]

details_change[, time_diff := as.integer(difftime(anytime(time), anytime(lead_time),units =
"hours")), by = c("matchId", "bookmaker", "oddtype")]

details_change <- details_change[time_diff != 0]

#details_change <- details_change[odd_diff != 0]

details_change[, diff := odd_diff / time_diff, by = c("matchId", "bookmaker", "oddtype")]

details_change_next <- details_change[matchId %in% next_matches$matchId]

details_change <- merge(details_change, matches[,c("matchId", "winner")], by = "matchId")

details_change[, c("odd", "time", "lead_odd", "lead_time", "odd_diff", "time_diff") := NULL]

details_change_next[, c("odd", "time", "lead_odd", "lead_time", "odd_diff", "time_diff") :=
NULL]

#View(details_change[oddtype == "odd2"])
```

```
details_change <- details_change[, avg:= mean(diff), by = c("matchId", "bookmaker", "oddtype")]

details_change_next <- details_change_next[, avg:= mean(diff), by = c("matchId", "bookmaker", "oddtype")]


#details_change <- details_change[, sd:= sd(diff), by = c("matchId", "bookmaker", "oddtype")]

#details_change <- details_change[, max:= max(diff), by = c("matchId", "bookmaker", "oddtype")]


details_change <- unique(details_change)

details_change_next <- unique(details_change_next)



#prepare first & last

key(details) <- c("matchId", "bookmaker", "oddtype")

#first <- details[unique(details[,key(details), with = FALSE]), mult = 'first']

last <- details[unique(details[,key(details), with = FALSE]), mult = 'last']


set_directory("code")


RPS Calculation

calculate_rps <- function(a,b,c,d){
 if(is.na(a) || is.na(b) || is.na(c) || is.na(d)){
   as.double(NA)
 }
 else{
   if (d == "odd1") {d=1}
   if (d == "oddX") {d=2}
   if (d == "odd2") {d=3}
   pred = t(matrix(c(a, b, c)))
   output <- rps(obs = c(d), pred = pred)
   output$rps
```

```
  }
}
```

Shin Calculation

```
shin_prob_calculator <- function(list){
   a = list[1]
   b = list[2]
   c = list[3]

   if(is.na(a) || is.na(b) || is.na(c)){
     as.double(NA)
   }
   else{
     beta <- a + b + c
     z_current = 0
     z_new = sqrt(z_current^2+4*(1-z_current)*a*a/beta)+sqrt(z_current^2+4*(1-
z_current)*b*b/beta)+sqrt(z_current^2+4*(1-z_current)*c*c/beta)-2
     i = 1
     while (abs(z_new - z_current) > 0.001 && i < 21 ){
       z_current = z_new
       z_new = sqrt(z_current^2+4*(1-z_current)*a*a/beta)+sqrt(z_current^2+4*(1-
z_current)*b*b/beta)+sqrt(z_current^2+4*(1-z_current)*c*c/beta)-2
       i = i+1
     }
     z <- round((z_current+z_new)/2,3)
     (sqrt(z^2+4*(1-z)*list*list/beta)-z)/(2-2*z)

   }
}
```

train.model.R

```
rpsCaret<- function (data, lev = NULL, model = NULL) {
  require(verification)
```

```
  if (!all(levels(data[, "pred"]) == levels(data[, "obs"])))
    stop("levels of observed and predicted data do not match")
  rownames(data) <- NULL
  obs <- as.vector(as.numeric(data$obs))
  pred <- as.matrix(subset(data, select = levels(data$obs)))
  rpsObject <- verification::rps(obs, pred)
  out<- (-1) * rpsObject$rps
  names(out) <- c("rpsScore")
  out
}
environment(rpsCaret) <- asNamespace('caret')

# unique(matches[season == '2018-2019']$week)
# matches_df = next_matches[week == 51]
# matches_df = matches[season == '2018-2019']
# details_df = shin_changes_insider
models <- function(matches_df, details_df,
                   model_type = c("random_forest", "glmnet","gradient_boosting","decision_tree"),
                   is_ordered = FALSE, is_best_model = FALSE) {

  test_match_ids <- matches_df$matchId
  test_data <- details_df[matchId %in% test_match_ids]
  if (all(is.na(test_data$winner))){
    wide_test <- subsetBookies(bookiesToKeep, last[matchId %in% test_match_ids])
    wide_test <- reshape(wide_test, idvar = c("matchId", "bookmaker"), timevar =
c("oddtype"), direction = "wide")
    wide_test <- reshape(wide_test, idvar = c("matchId"), timevar = c("bookmaker"), direction
= "wide")
    wide_test$winner <- NA
  }
  if (!all(is.na(test_data$winner))){
    wide_test <- widening_withwinner(test_data, bookiesToKeep)
    wide_test <- wide_test[complete.cases(wide_test)]
```

```
}
min_date <- min(matches_df[matchId %in% test_match_ids]$date)
if(nrow(wide_test) > 40){prev_date <- 1000}
if(nrow(wide_test) <= 40){prev_date <- 365}

lower_date <- as.Date(min_date) - prev_date
train_match_ids <- matches[date < min_date][date > lower_date]$matchId
train_data <- details_df[matchId %in% train_match_ids]
wide_train <- widening_withwinner(train_data, bookiesToKeep)
wide_train <- wide_train[complete.cases(wide_train)]

train <- wide_train[,-c("matchId", "date", "week", "season")]
test <- wide_test[,-c("matchId", "winner", "date", "week", "season")]

if (is_ordered){train$winner <- ordered(train$winner, levels = c("odd1", "oddX", "odd2"))}
print(paste(model_type, "is_ordered", is_ordered))

if(is_best_model){
  fit <- best_model(train, test, wide_test, "gradient_boosting")
}

if(!is_best_model){
  if (model_type == "random_forest") {
    if (is_ordered){
      fit <- ord_rf(train, test, wide_test)
    }
    if(!is_ordered){
      fit <- random_forest(train, test, wide_test)
    }
  }
  if (model_type == "vglm") {
    if (is_ordered){
```

```
      fit <- vglmCumulative(train, test, wide_test)
    }
    if(!is_ordered){
      print("Vglm works with ordinal variables")
    }
  }
  if (model_type == "glmnet") {
    if (is_ordered) {
      print("Ordered GLMNET is not implemented yet.")
    }
    if (!is_ordered){
      fit <- train_glmnet(train, test, wide_test)
    }
  }
  if (model_type == "gradient_boosting") {
    fit <- gradient_boosting(train, test, wide_test)
  }
  if (model_type == "decision_tree") {
    fit <- decision_tree(train, test, wide_test)
  }
}

ourRPS <- mean(fit[[2]]$RPS)
preds <- fit[[2]]
fit <- fit[[1]]
print(preds[oddX > 0.3])

week_number <- unique(matches_df$week)
if (length(week_number) > 1) {week_number <- paste(length(week_number), "weeks")}
season_number <- unique(matches_df$season)
test_size <- nrow(matches_df)
current_time <- format(Sys.time(), "%Y-%m-%d %X")
```

```
  if(any(grepl('avg', colnames(details_df)))){
    if ("z" %in% colnames(details_df)){feature <- "A+B+C"}
    if (!"z" %in% colnames(details_df)){feature <- "A+C"}
  }
  if (!(any(grepl('avg', colnames(details_df))) & ("z" %in% colnames(details_df))) {feature <-
"A+B"}
  if (!(any(grepl('avg', colnames(details_df))) & !("z" %in% colnames(details_df))) {feature
<- "A"}
  str = ""
  for (i in 1:ncol(fit$bestTune)) {
    str <- paste(str, names(fit$bestTune[i]), fit$bestTune[i][1,])
  }

  df_summary <- read.csv("summary2.csv")

  df_new <- data.frame(ModelType = model_type,
                       Feature = feature,
                       Ordered = is_ordered,
                       Weeks = week_number,
                       Seasons = season_number,
                       TestSize = test_size,
                       TrainStart = toString(lower_date),
                       TestStart = toString(min_date),
                       OurRPS = ourRPS,
                       BestTune = str,
                       timestamp = current_time)

  df_summary <- rbind(df_summary, df_new)
  write.csv(df_summary, file = "summary2.csv", row.names = FALSE, quote = FALSE)
  return(list(fit, ourRPS, preds))
}
```

```
#train and test are necessary, requires defined lastrps (maybe use only last?)
#returns output_prob, testRPS
random_forest <- function(train, test, wide_test){
  set.seed(1234)
  control <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 3,
                          search = "grid",
                          classProbs = TRUE,
                          summaryFunction = rpsCaret)
  tunegrid <- expand.grid(.mtry= 4+(0:8)*0.5)

  fit <- train(winner~.,
               data=train,
               method="rf",
               tuneGrid=tunegrid,
               trControl=control,
               importance = T)

  output_prob <- predict(fit, test, "prob")
  colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  return(list(fit, output_prob))
}

ord_rf <- function(train, test, wide_test){
  set.seed(1234)
  # There are several hyperparameters, which do, however,
  # not have to be optimized by the user in general, because the default
  # values used for these hyperparameters were seen to be in a reasonable
```

```
  # range and the results seem to be quite robust with respect to the
  # choices of the hyperparameter values.
  fit <- ordfor(depvar="winner",
                data=train, nsets=1000, ntreeperdiv=400,
                ntreefinal=5000, perffunction = "equal")
  preds <- predict(fit, newdata=test, "prob")
  output_prob <- data.table(preds$classfreqtree)
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  return(list(fit, output_prob))
}

train_glmnet <- function(train, test, wide_test,
                  alpha=1,nlambda=50, tune_lambda=T,nofReplications=2,
                  nFolds=10,trace=F, max = F){
  set.seed(1234)
  train$winner <- convert(train$winner)
  train_class <- as.numeric(train$winner)
  train <- train[,-c("winner")]
  if (max) {
    glm_train_data$max <- do.call(pmax, glm_train_data)
    glm_test_data$max <- do.call(pmax, glm_test_data)
  }
  if(tune_lambda){
    # to set lambda parameter, cross-validation will be performed and lambda is selected based
on RPS performance
    cvindices <- generateCVRuns(train_class,nofReplications,nFolds,stratified=TRUE)

    # first get lambda sequence for all data
    glmnet_alldata <- glmnet(as.matrix(train), as.factor(train_class), family="multinomial",
alpha = alpha, nlambda=nlambda)
    lambda_sequence <- glmnet_alldata$lambda
```

```
  cvresult=vector('list',nofReplications*nFolds)
  iter=1
  for(i in 1:nofReplications) {
    thisReplication=cvindices[[i]]
    for(j in 1:nFolds){
      if(trace){
        cat(sprintf('Iteration %d: Fold %d of Replication %d\n',iter,j,i))
      }
      testindices <- order(thisReplication[[j]])

      cvtrain <- train[-testindices]
      cvtrainclass <- train_class[-testindices]
      cvtest <- train[testindices]
      cvtestclass <- train_class[testindices]

      inner_cv_glmnet_fit <-
glmnet(data.matrix(cvtrain),as.factor(cvtrainclass),family="multinomial", alpha =
alpha,lambda=lambda_sequence)
      valid_pred <- predict(inner_cv_glmnet_fit, data.matrix(cvtest), s = lambda_sequence,
type = "response")

      #check order of predictions
      order_of_class <- attr(valid_pred,'dimnames')[[2]]
      new_order <-
c(which(order_of_class=='1'),which(order_of_class=='2'),which(order_of_class=='3'))
      foldresult <- rbindlist(lapply(c(1:length(lambda_sequence)),function(x) {
data.table(repl=i,fold=j,lambda=lambda_sequence[x],valid_pred[,new_order,x],result=cvtestc
lass)}))
      cvresult[[iter]]=foldresult
      iter=iter+1
    }
  }
```

```
    cvresult <- rbindlist(cvresult)
    cvresult$result <- convert(cvresult$result)
    names(cvresult) <- c("repl", "fold", "lambda", "odd1", "oddX", "odd2", "result")

    # creating actual targets for rps calculations
    cvresult[,pred_id:=1:.N]
    outcome_for_rps <- data.table::dcast(cvresult,pred_id~result,value.var='pred_id')
    outcome_for_rps[,pred_id:=NULL]
    outcome_for_rps[is.na(outcome_for_rps)]=0
    outcome_for_rps[outcome_for_rps>0]=1
    setcolorder(outcome_for_rps, c("odd1", "oddX", "odd2"))

    # calculate RPS
    cvresult <- cvresult[, RPS := calculate_rps(odd1, oddX, odd2, result), by =
1:nrow(cvresult)]
    overall_results <- data.table(cvresult[,list(repl,fold,lambda,RPS)])

    # summarize performance for each lambda
    overall_results_summary <- overall_results[,list(RPS=mean(RPS)),list(repl,fold,lambda)]

    # find best lambdas as in glmnet based on RPS
    overall_results_summary <-
overall_results_summary[,list(meanRPS=mean(RPS),sdRPS=sd(RPS)),list(lambda)]
    overall_results_summary[,RPS_mean_lb := meanRPS - sdRPS]
    overall_results_summary[,RPS_mean_ub := meanRPS + sdRPS]

    cv_lambda_min <- overall_results_summary[which.min(meanRPS)]$lambda

    semin <- overall_results_summary[lambda==cv_lambda_min]$RPS_mean_ub
    cv_lambda.1se <- max(overall_results_summary[meanRPS<semin]$lambda)

    cvResultsSummary = list(lambda.min =cv_lambda_min, lambda.1se = cv_lambda.1se,
```

```
meanRPS_min=overall_results_summary[lambda==cv_lambda_min]$meanRPS,

meanRPS_1se=overall_results_summary[lambda==cv_lambda.1se]$meanRPS)
  }


  fit <- glmnet(as.matrix(train),as.factor(train_class),family="multinomial", alpha =
alpha,lambda=cvResultsSummary$lambda.min)
  predicted_probabilities <- predict(fit, as.matrix(test), type = "response")
  output_prob <- data.table(predicted_probabilities[,,])
  colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  fit$bestTune <- data.frame(lambda = fit$lambda)
  return(list(fit, output_prob))
}


train_glmnetcr <- function(train, test, wide_test,
                           alpha=1,nlambda=50,nofReplications=2,
                           nFolds=10){
  set.seed(1234)
  train_class <- train$winner
  train <- train[,-c("winner")]


  # to set lambda parameter, cross-validation will be performed and lambda is selected based
on RPS performance
  cvindices <- generateCVRuns(train_class,nofReplications,nFolds,stratified=TRUE)


  # first get lambda sequence for all data
  glmnet_alldata <- glmnetcr(as.matrix(train), as.factor(train_class), alpha = alpha,
nlambda=nlambda)
  lambda_sequence <- glmnet_alldata$lambda
```

```
 cvresult=vector('list',nofReplications*nFolds)
 iter=1
 for(i in 1:nofReplications) {
  thisReplication=cvindices[[i]]
  for(j in 1:nFolds){
    testindices <- order(thisReplication[[j]])

    cvtrain <- train[-testindices]
    cvtrainclass <- train_class[-testindices]
    cvtest <- train[testindices]
    cvtestclass <- train_class[testindices]

    inner_cv_glmnet_fit <- glmnetcr(data.matrix(cvtrain),as.factor(cvtrainclass), alpha =
alpha,lambda=lambda_sequence)
    valid_pred <- predict(inner_cv_glmnet_fit, data.matrix(cvtest), s = lambda_sequence, type
= "prob")

    #check order of predictions
    #order_of_class <- attr(valid_pred,'dimnames')[[2]]
    #new_order <-
c(which(order_of_class=='1'),which(order_of_class=='2'),which(order_of_class=='3'))
    foldresult <- rbindlist(lapply(c(1:length(lambda_sequence)),function(x) {

data.table(repl=i,fold=j,lambda=lambda_sequence[x],valid_pred$probs[,,x],result=cvtestclass
)
      }))
    cvresult[[iter]]=foldresult
    iter=iter+1
  }
 }

 cvresult <- rbindlist(cvresult)
 #cvresult$result <- convert(cvresult$result)
 #names(cvresult) <- c("repl", "fold", "lambda", "odd1", "oddX", "odd2", "result")
```

```
  # creating actual targets for rps calculations
  cvresult[,pred_id:=1:.N]
  outcome_for_rps <- data.table::dcast(cvresult,pred_id~result,value.var='pred_id')
  outcome_for_rps[,pred_id:=NULL]
  outcome_for_rps[is.na(outcome_for_rps)]=0
  outcome_for_rps[outcome_for_rps>0]=1
  setcolorder(outcome_for_rps, c("odd1", "oddX", "odd2"))

  # calculate RPS
  cvresult <- cvresult[, RPS := calculate_rps(odd1, oddX, odd2, result), by = 1:nrow(cvresult)]
  overall_results <- data.table(cvresult[,list(repl,fold,lambda,RPS)])

  # summarize performance for each lambda
  overall_results_summary <- overall_results[,list(RPS=mean(RPS)),list(repl,fold,lambda)]

  # find best lambdas as in glmnet based on RPS
  overall_results_summary <-
overall_results_summary[,list(meanRPS=mean(RPS),sdRPS=sd(RPS)),list(lambda)]
  overall_results_summary[,RPS_mean_lb := meanRPS - sdRPS]
  overall_results_summary[,RPS_mean_ub := meanRPS + sdRPS]

  cv_lambda_min <- overall_results_summary[which.min(meanRPS)]$lambda

  semin <- overall_results_summary[lambda==cv_lambda_min]$RPS_mean_ub
  cv_lambda.1se <- max(overall_results_summary[meanRPS<semin]$lambda)

  cvResultsSummary = list(lambda.min =cv_lambda_min, lambda.1se = cv_lambda.1se,
meanRPS_min=overall_results_summary[lambda==cv_lambda_min]$meanRPS,
meanRPS_1se=overall_results_summary[lambda==cv_lambda.1se]$meanRPS)
```

```
  fit <- glmnetcr(as.matrix(train),as.factor(train_class), alpha =
alpha,lambda=cvResultsSummary$lambda.min)
  predicted_probabilities <- predict(fit, data.matrix(test), type = "response")
  output_prob <- data.table(predicted_probabilities[,,])
  colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  fit$bestTune <- data.frame(lambda = fit$lambda)
  return(list(fit, output_prob))
}



gradient_boosting <- function(train, test, wide_test){
  fitControl <- trainControl(method = "repeatedcv",
                             number = 10,
                             repeats = 3,
                             classProbs = T,
                             summaryFunction = rpsCaret)
  tune_Grid <-  expand.grid(interaction.depth = c(1,3,5),
                            n.trees = (1:10)*50,
                            shrinkage = c(0.01, 0.05, 0.1),
                            n.minobsinnode = c(5,10))
  #plot(tune_Grid) #Error in plot.new() : figure margins too large
  #plot(gbmFit, plotType = "level")
  set.seed(1234)
  fit <- train(winner ~ .,
               data = train,
               method = "gbm",
               trControl = fitControl,
```

```
            verbose = FALSE,
            tuneGrid = tune_Grid)

  output_prob <- predict(fit, test, "prob")
  #colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  return(list(fit, output_prob))
}


decision_tree <- function(train, test, wide_test){
  set.seed(1234)
  fit <-  train(y = train$winner,
            x = train[,-c("winner")],
            method = "rpart",
            tuneGrid = expand.grid(.cp = c((1:7)*0.03)),
            trControl = trainControl(method = "repeatedcv",
                                     number = 10,
                                     repeats = 10,
                                     classProbs = T,
                                     summaryFunction = rpsCaret))

  output_prob <- predict(fit, test, "prob")
  #colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  return(list(fit, output_prob))
}
```

```
vglmCumulative <- function(train, test, wide_test){
  set.seed(1234)
  control <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 3,
                          classProbs = TRUE,
                          savePredictions = T)
  tunegrid <- expand.grid(parallel = TRUE, link = c("logit", "probit"))


  fit <- train(winner~.,
               data=train,
               method="vglmCumulative",
               trControl=control,
               importance = T)

  output_prob <- predict(fit, test, "prob")
  colnames(output_prob) <- c("odd1", "oddX", "odd2")
  output_prob$matchId <- wide_test$matchId
  setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
  output_prob <- comparison(output_prob, trace = T)
  return(list(fit, output_prob))
}

best_model <- function(train, test, wide_test, best_name){

  if(best_name == "gradient_boosting"){
    fitControl <- trainControl(method = "repeatedcv",
                               number = 10,
                               repeats = 3,
                               classProbs = T,
                               summaryFunction = rpsCaret)
```

```
  tune_Grid <-  expand.grid(interaction.depth = 1,
                            n.trees = c(200),
                            shrinkage = 0.01,
                            n.minobsinnode = 5)
  set.seed(1234)
  fit <- train(winner ~ .,
               data = train,
               method = "gbm",
               trControl = fitControl,
               verbose = FALSE,
               tuneGrid = tune_Grid)
}
if(best_name == "decision_tree"){
  set.seed(1234)
  fit <-  train(y = train$winner,
                x = train[,-c("winner")],
                method = "rpart",
                tuneGrid = expand.grid(.cp = c(0.06)),
                trControl = trainControl(method = "repeatedcv",
                                         number = 10,
                                         repeats = 10,
                                         classProbs = T,
                                         summaryFunction = rpsCaret))
}
if(best_name == "random_forest"){
  set.seed(1234)
  control <- trainControl(method = "repeatedcv",
                          number = 10,
                          repeats = 3,
                          search = "grid",
                          classProbs = TRUE,
                          summaryFunction = rpsCaret)
```

```
  tunegrid <- expand.grid(.mtry= 4.5)

  fit <- train(winner~.,
               data=train,
               method="rf",
               tuneGrid=tunegrid,
               trControl=control,
               importance = T)
 }

 output_prob <- predict(fit, test, "prob")
 #colnames(output_prob) <- c("odd1", "oddX", "odd2")
 output_prob$matchId <- wide_test$matchId
 setcolorder(output_prob, c("matchId", "odd1", "oddX", "odd2"))
 output_prob <- comparison(output_prob, trace = T)
 return(list(fit, output_prob))
}
odd_strategy <- function(output_prob){
 play <- output_prob[oddX > 0.30][odd2 < 0.315]
 table(play$winner)
 best_bookmaker <- "Pinnacle"
 portfolio_df <- lastrps[bookmaker == best_bookmaker][matchId %in%
play$matchId][,c(1,3,4,5)]
 #portfolio_df <- shin_changes[bookmaker == best_bookmaker][matchId %in%
play$matchId]
 odds <- details[bookmaker == "Pinnacle"][matchId %in% play$matchId]
 key(odds) <- c("matchId", "oddtype")
 last_odds <- odds[unique(odds[,key(odds), with = FALSE]), mult = 'all']
 last_oddsX <- last_odds[oddtype == "oddX"]
 reshape(last_odds, idvar = c("bookmaker", "matchId"), timevar = "oddtype", direction =
"wide" )
 View(merge(next_matches[,c(1,3,4)], play, by = "matchId")[,c(1:6)])
```

```
  final <- merge(play, last_oddsX, by = "matchId")
  final <- merge(final, portfolio_df, by = "matchId")
  final <- final[,c(1,2,3,4,10,12,11,5,9)]
  final$initial <- 10
  final$onhand <- 0
  final$cumulative <- 0
  final$net <- 0

  if (final$winner[1] == "oddX"){
    final$onhand[1] <- final$odd[1] * 10
    final$cumulative[1] <- final$onhand[1]
  }
  for (i in (2:nrow(final))){
    if (final$winner[i] == "oddX"){
      final$onhand[i] <- final$odd[i] * 10
    }
    final$cumulative[i] <- final$onhand[i] + final$cumulative[i-1]
  }
  for (i in (1:nrow(final))){
    final$net[i] <- final$cumulative[i] - i * 10
  }
  View(final)
  #tree_fit <- train(winner ~ ., as.matrix(final[,c(2:9)]), "rpart")
  plot_ly(x = 1:nrow(final), y = final$net, name = "Cumulative Net Gain", type = "scatter",
mode = "lines+markers") %>%
    add_trace(y = final$onhand, name = 'On Hand Money', mode = 'markers') %>%
    layout(title = '2017-2018 Season Odd Strategy',
           yaxis = list(title = 'Gain'),
           xaxis = list(title = 'Match Number'))

  return(final)
}
```