Citadel East Coast Regional Datathon

Team 10: Filipp Filippov, Di He, Rachit Surana, Tanner Zachem

March 28, 2021

1.Topic Question

Sports betting is a massive industry, its estimated market size was more than 200 Billion US Dollars in 2020.[1] Nevertheless, profitable sports betting is also known to be notoriously hard: very few individuals can actually make consistent profits out of it. If a certain bettor, on average, is able to predict 53% of matches' outcomes correctly, they are considered a very successful bettor. Considering this context, we decided to pose the following question: **Can we create a model to predict outcomes such that we can make consistent money gambiling, and beat an average gambler?**

2.Executive Summary

The major goal of this project is to predict the outcomes of football matches and then to use these predictions in the betting strategy. The ultimate goal is to come up with such a strategy that will yield consistent profit to the bettor over the long term. This goal is achieved in several steps: at first, the data is cleaned and the features are selected. Secondly, we build a model which predicts the outcome of a certain football match based on the set of features which include team attributes: certain characteristics unique to each specific team, the mean betting odds placed by bookmakers *prior* to the match, as well as some players' characteristics and lagged features corresponding to the outcome of the previous game. Finally, we use predictions of the matches outcomes for determining our own betting amounts and later evaluation of its performance compared to a simple random bettor.

The first problem is creating match predictions. After the data cleaning we obtain the variable called home_win. It is a discrete variable representing an outcome of a particular match taking three possible values: 2, if the match was won by the home team, 1, if there was a draw and 0 if the away team has won. This will be the variable of our major interest in this study. There are some correlations, both positive and negative between the home_win (our target) and features, especially if we consider mean bets. This is not surprising as betting agencies already use immensely sophisticated algorithms for determining the winner and their odds so the mean odds even prior to the match is a good predictor for the outcome of the game. Through the process of model selection we chose the Random Forest classifier. We managed to achieve the in-sample accuracy of 53%, validation accuracy slightly above 52% and testing accuracy of 50.6%. Although these numbers may seem low, when applied to a basic betting strategy, over ten "seasons" the informed model went up 56.40 units whereas an average, random bettor lost 1547.19 units. **If the unit size is $10, then our model outperformed an average bettor (the market) by $16,035.9.**

[1] https://www.statista.com/statistics/1154681/key-data-global-sports-betting-industry/

3.Technical Exposition

3.1. Data Cleaning and Feature Engineering

Since our first goal was trying to predict an outcome of the game, our very first aim was obtaining a data frame, containing the target variable *home_win* as well as all features which could be potentially useful in predicting the target's label.

Our first set of features included the individual teams' characteristics, both for home and for away teams. We dropped the columns *buildUpPlayDribbling* and *buildUpPlayDribbling_away_team* as these had more than a half NaN values, so data imputation of any kind would be worthless in this case. For all other features we imputed the NaN values using the median of the non-null values. The idea was that such characteristics should influence the power of the teams greatly and, hence, the outcome of the games as well. However, the models performed not satisfactory with such set of features, so we decided to search for more.

As we people often guess the outcome of the match based on the two teams' previous performance, we added new features *pre_home_points_avg* and *pre_away_points_avg* to measure the average points home team and away team got in their previous 10 matches.

We decided to include the betting information into our feature space as betting agencies typically possess a lot of valuable information about the upcoming games, so their odds should be necessarily somehow predictive of the match outcome. It is important to note that the betting information we included was *prior* to the match, so there is no information which was known during or after the game. As for the exact features included, we defined 6 variables: *MBOH*, *MBOA*, *MBOD*, *SBOH*, *SBOA* and *SBOD*. The first three correspond to the mean odds placed by bookmakers on home win, away win or draw respectively, while the latter three are the standard deviations of the odds. While it is obvious why mean odds can be useful, let us clarify why we included the standard deviations as well. The less is the standard deviation, the more the betting companies agree on the outcome of the game. The higher confidence of the betting agencies might be a good proxy for the outcome of the game.

We also decided to include some players' features, but not entirely individual ones, rather aggregated: we chose maximum and standard deviation of such characteristics as player overall rating, finishing, crossing and some others. We did so both for home and for away team. The maximum should correspond to the most talented players in the team, "the stars", while the standard deviation of these scores represent whether the players of a specific team differ from each other in terms of their skills significantly: if that is the case, the team overall may not play very well as there will be a large discrepancy between the players and they may interact between each other on the field more poorly.

3.2. Exploratory Data Analysis

One of the most basic and simplistic assumptions about the data one could possibly imagine is an assumption about linearity: whether the target variable is linearly related to the set of features. If it holds, the linear regression model for the regression task and logistic regression for classification task are the most suitable models. In this case it is also possible to use the regular correlation coefficient.

Our initial set of features (which contained only the various aggregate team metrics) actually did not possess very noticeable correlations with the target: the highest in absolute values were around 5-6%. After we extended the feature space to include betting information along with players' characteristics, the correlations overall bumped up significantly: *MBOH* and *MBOA* features (mean bets on home and away wins, respectively) turned out to have the

highest correlation with the target (-0.3392 and 0.3266). Players' characteristics turned out to be pretty noticeably correlated with the target as well, which can be seen from the plot below

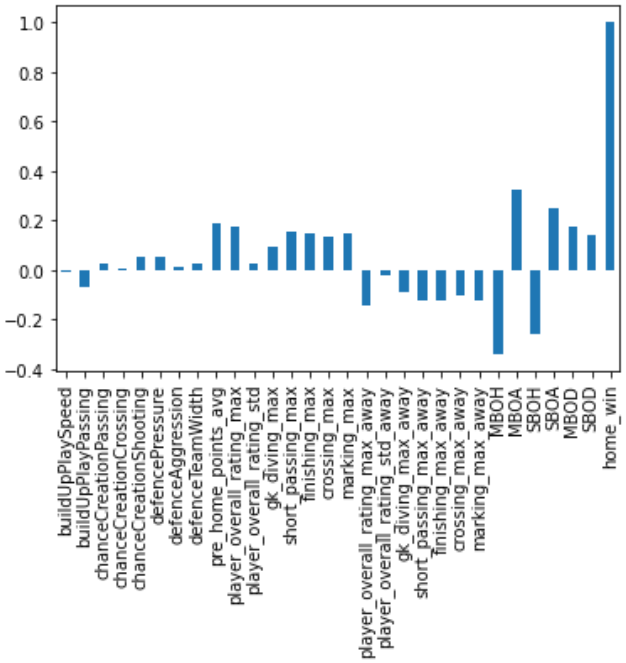

Figure 1. Bar plot of correlations between selected features and the target

The next plot depicts a heatmap of all the correlations between the features and the target

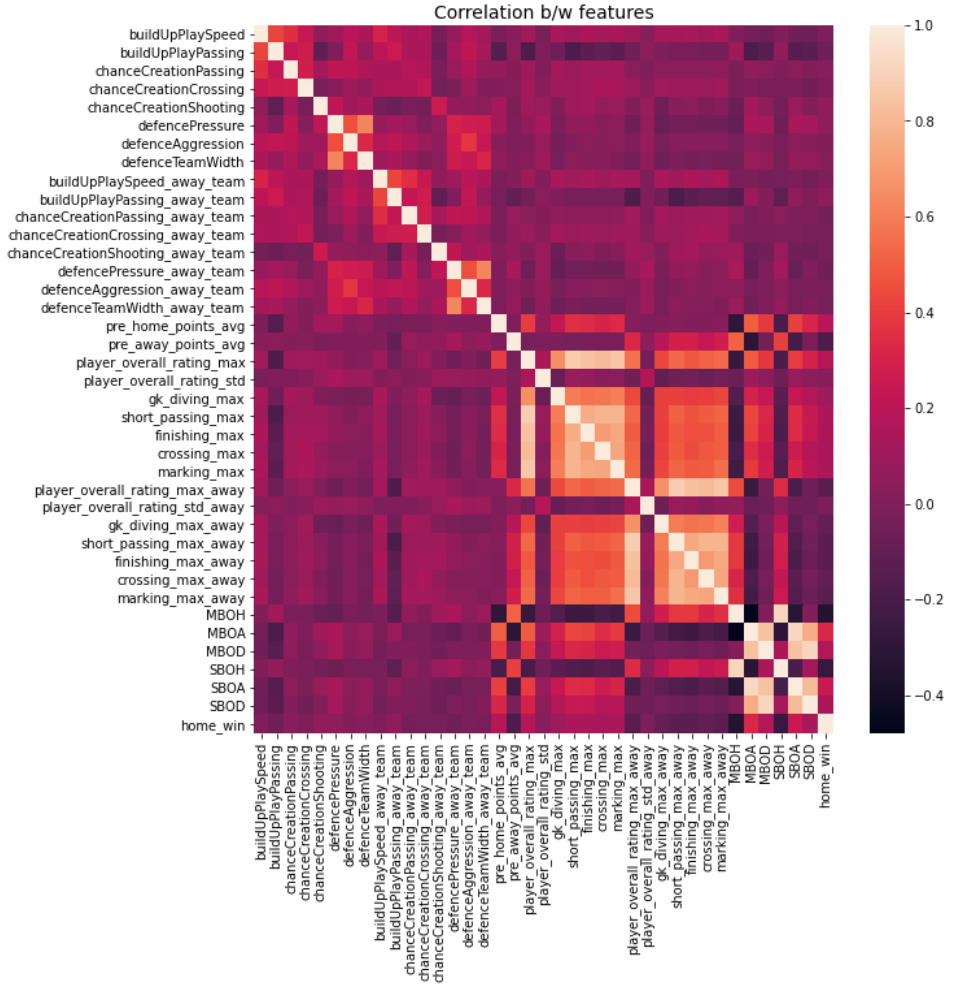

Figure 2. Correlation heat map

From the above two visualizations, we conclude that the target variable and the explanatory variables do not have a robust linear relationship and therefore we should not use a linear model for prediction.

The next plot further illustrates the dependencies between the features and the target. In particular, it illustrates the importance of the players' characteristics (in this case - max rating) and the outcome of the previous game to match outcome prediction. It also illustrates that there is not much difference in the distribution of games outcomes across various seasons, so this distribution is more or less constant over time which should make the models' performance more stable.

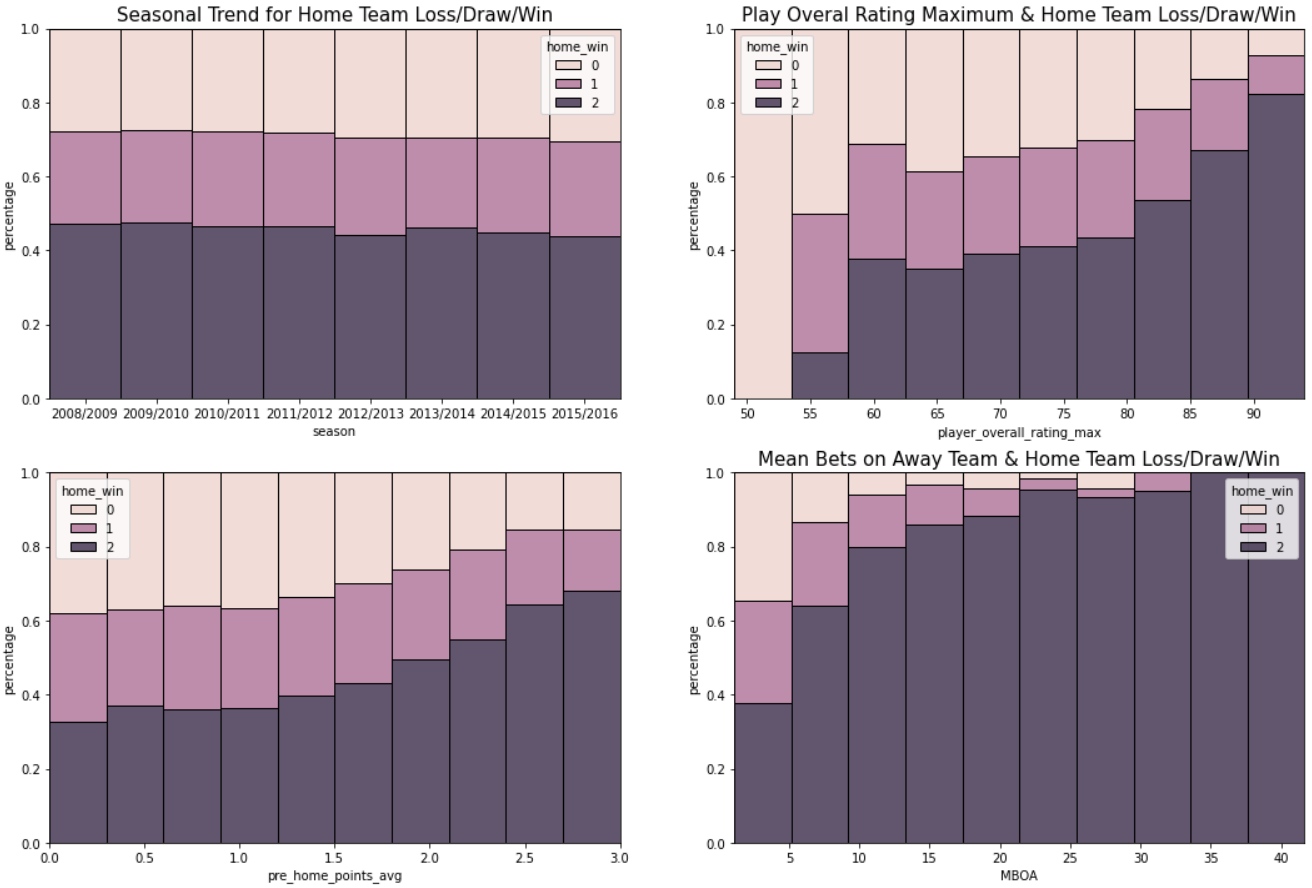

Figure 3. The distributions of home_win with respect to certain variables

3.3. Classification of games' outcomes

3.3.1. Model Selection

In this subsection we describe the chosen model for classification of matches' outcomes as well as some steps which have led to the choice of this specific classifier. We have restricted our choice to 3 popular classifiers: Logistic Regression, Support Vector and Random Forest. We have chosen these for several reasons: they are easy to implement using the scikit-learn package and despite this simplicity in implementation they can be pretty powerful. Moreover, such initial selection allows for high flexibility: if the classes are linearly separable, logistic regression will perform well, if not - SVC and random forest will come at play. Initially we also aimed to build a neural network, but abandoned this idea when, despite its much higher

complexity in implementation, a neural network showed the results not much different from the baseline logistic regression.

A couple of words about the model parameters: logistic regression and SVC are initialised in their most basic form, without any pre-specified hyperparameters. In this sense they are “baseline models''. Random forest classifier does not have many parameters to specify, but we initialised it to have non-default value of the number of estimators (1000 instead of 100) and max depth (7 instead of unlimited). This is done in order to prevent overfitting: variance of the random forest is inversely related to the number of estimators (decision trees) used. Similarly, limiting the max depth is supposed to reduce the risk of overfitting. These measures are taken as we have had a pretty painful experience of having an accuracy score of 85% on the training set and only slightly above 40% on the test set with an unregularised random forest.

As it was mentioned before, with our original set of features, our classifiers didn’t perform extremely well: the very basic logistic regression had in- and out-of sample accuracies around 46%. The support vector and random forest classifiers performed slightly better, yielding the in-sample scores around 49-50% and out-of-sample - 47-48%. This was one of the indications that there are some non-linearities present in our data. Still this didn’t seem too impressive, so we added more features.

Our new features include some time series data, such as outcomes of the previous matches, so standard *train_test_split* function of sklearn is no longer suitable. To handle this, we split the data into training, validation and testing based on the playing season: the test set is the season of 2015/16, validation - previous season of 2014/15 and the training set is the rest of the data.

Figure 4. Training, Validation, Testing Data Separation

After this feature engineering the results noticeably improved, and their performance on training and validation are listed below

Features Used	Model	Train Accuracy	Validation Accuracy
Team Attributes	Random Forest	0.484	0.454
Team Historical Performance	Random Forest	0.499	0.474
Player Attributes Metrics	Random Forest	0.522	0.493
Mean Bets on Match	Random Forest	0.530	0.503
All Above	**Random Forest**	**0.533**	**0.522**
All Above	Logistic Regression	0.521	0.518
All Above	SVM	0.464	0.450

Table 1. Final model performance with and without certain features

3.3.2. Model Evaluation

Overall, based on the above analysis we finally chose the random forest with all features as our model. To evaluate its performance in real words, we put training and validation set together to train the model, and computed the accuracy on the testing dataset (new 2015/2016 season). We finally got an **accuracy of 0.506 at the testing season**. Moreover, we calculated the precision and recall at the testing dataset and got 0.754, 0.506 respectively. And the confusion matrix is as below. Because our goal is to increase accuracy in order to form a better betting strategy, we choose accuracy as our training evaluation metric. However, considering our target labels are imbalanced and we used accuracy as the evaluation metric, our model will prefer to give home team win (*Class 2*) outcome rather than draw (*Class 1*) or away team win (*Class 0*).

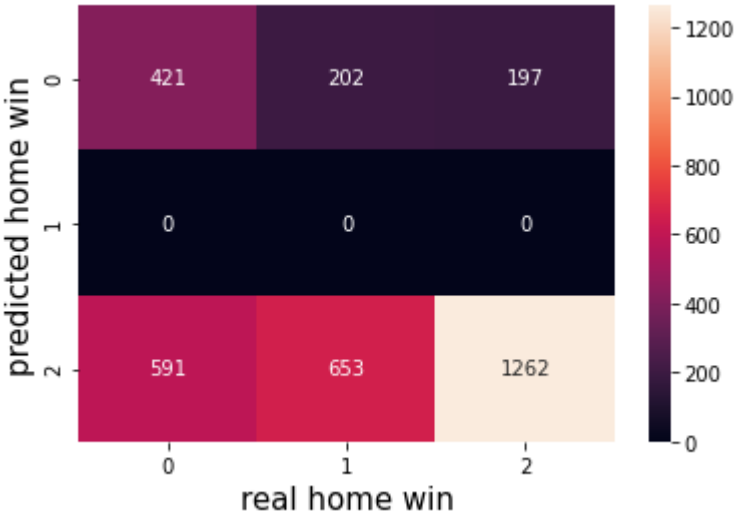

Figure 5. Confusion Matrix of Random Forest Model

3.3.3. Model Interpretation

We use the SHAP package to interpret our model. The picture below shows top 10 important features and their impacts on output classification. We found that betting (top 5) is highly related with our predictions. Home team's historical performance and players' rating is also important.

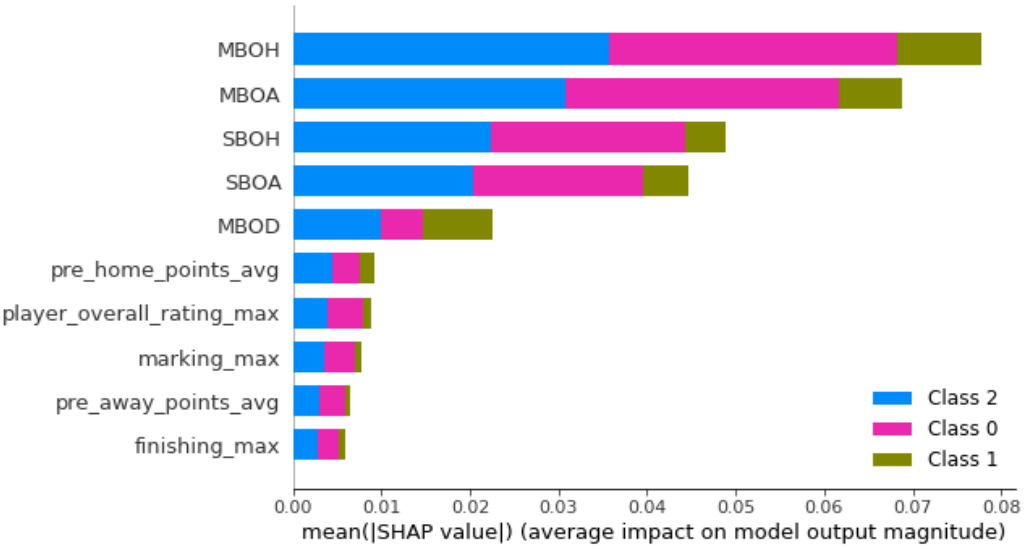

Figure 6. SHAP Summary Plot

Let's analyze those features' impacts on *Class 2* (home team win). From the picture below, we knew lower *MBOH* (average betting on home team), the higher probability of winning. And the better the home team performance in the previous matches, the higher probability of winning. It seems the way our model makes decisions is just similar to humans.

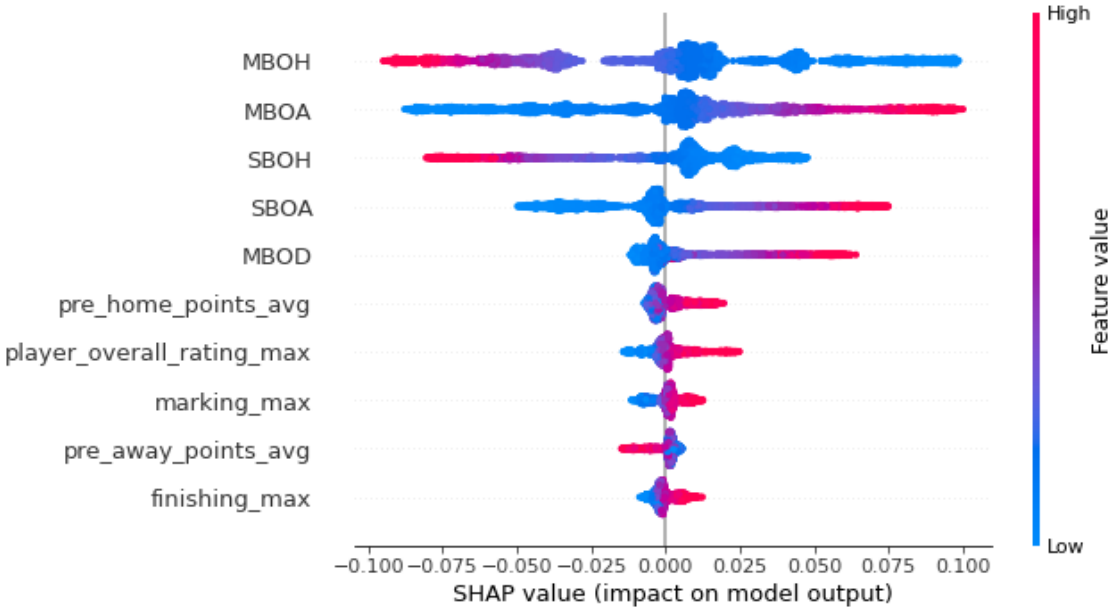

Figure 7. SHAP Summary Plot on Class 2 (Home Team Win)

To help our potential investors better understand how our algorithm makes decisions based on the input features, we randomly selected one match and see why our algorithm says it has a 0.55 probability for the home team to win. We can see the probability grows from base value (0.45) and gets 0.55. Except for those betting features, we found *Marking_Max* is also important. The model thought the home team's ability to track and defend an opposing player is a key to make it win.

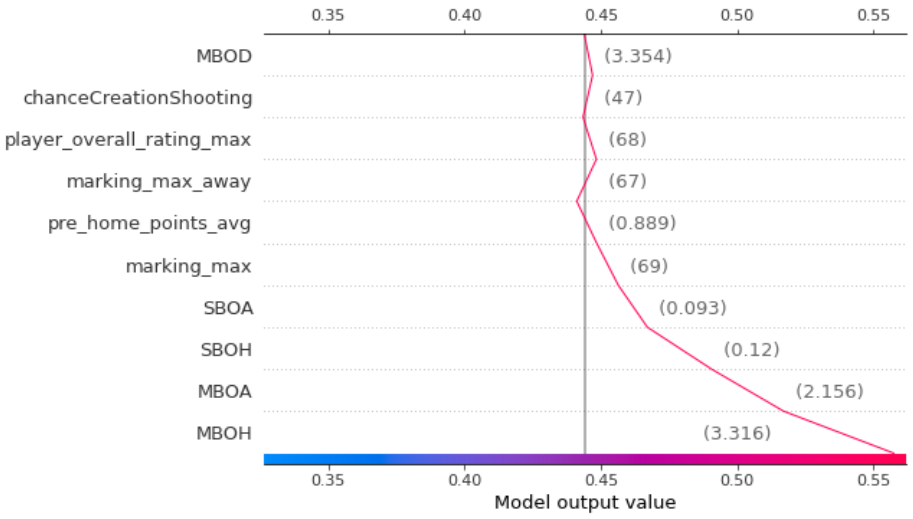

Figure 8. SHAP Plot on One Data Instance

4. Betting Strategy

Gambiling is an industry that is created to ensure that the bettor will lose money in the long run. For our analysis we will only consider what are known as straight bets, bets in which the bettor chooses the outcome of a game. In order to make all choices attractive, depending on the likelihood of each outcome, home win, draw, and home loss, each bet has different odds. The more unlikely the outcome, the higher the payout. In order to assure a net profit for the sports book, bookkeepers do not provide fair lines: if a team has 2:1 odds to win a game, a gambler would only be able to bet on odds worse than that, this is how sportsbooks ensure their return. Due to this, **a random betting strategy is probabilistically ensured to lose money over time**. it should be noted that when the word "random" is used, it implies a bettor that selects a bet because they want to. It is impossible to model the complex emotional factors that go into a bet, but what all bets have in common is that they can be appropriately modeled as random. In order to beat the ensured negative long term return of random betting, the Random Forest model explained previously is used to inform a betting strategy.

The betting strategy employed is quite simple. Since the Random Forest outputs a confidence of each match's outcome, the algorithm will look at the confidence, and only place a bet if it is over a certain threshold. The reason that not every bet is placed is simple: why would our bettor risk units if the RF informing the choice isn't sure of the games outcome (the case where no value is greater than .5) or if the confidence in a games outcome is low (confidence for an outcome is less than some parameter). Past this threshold the algorithm will wager one unit on the outcome, if the model is more confident (.82) the algorithm will wager two units, and at .85 the algorithm wagers three units.

Confidence Value	Wager Size
[0,.75]	0
(.75,.82]	1
(.82,.85]	2
(.85,1]	3

Table 2: RF Confidence Value and Wager Size

Since the odds were given in decimal form, the bet value is equal to the product of the wager and the odds. In order to track this over time, once a wager is placed, the units are subtracted from the bettors wallet. If the bet is correct, the entire bet amount is given to the wallet.

The parameters for the wager cutoffs were found by simple trial and error. In order to see the effectiveness of our algorithm it was necessary to compare it to a random bettors performance. A random person placing a bet is simply choosing from one of three outcomes, home win, draw, and home loss. Therefore, in order to choose what action the random bettor would take, a value was sampled from a uniform distribution between zero and one, and depending on cutoffs was given a bet. Since a draw is bet less than a winning outcome, the range for that bet was lower. If the sampled value was less than .2 a draw was bet, between .2 and .6 (inclusive on upper bound) a home win was bet, and between .6 and 1 a home loss was bet.

Sampled Value	Bet Selection
[0,.2]	Draw (1)
(.2,.6]	Home Win (2)
(.6,1]	Home Loss (3)

Table 3: Sampled Value and Random Bettor Bet Selection

In order to give the average bettor a way to place different sized wagers, a new value between zero and one was sampled from a uniform distribution. The wager values are displayed below:

Sampled Value	Wager Size
[0,.4]	0
(.4,.6]	1
(.6,.9]	2
(.9,1]	3

Table 4: Sampled Value and Random Bettor Wager Size

The values for these cutoffs were chosen at random, however, many different combinations of values were tried and all values resulted in a large net loss of units for the average bettor. Bookmakers do not provide even lines for bets, the lines are created such that in the long run the book will always win, therefore it makes sense that many random combinations of parameters for the average bettor resulted in a net loss.

To test the performance of our bettor vs a random bettor, their performances were tested in 10 trials of the 2015/2016 season of mock gambiling. The results were incredibly clear, although our algorithm never won big, it slowly and consistently won units, whereas the random better would have large wins and large losses, but everytime the losses would be more frequent, and larger than the wins. These results are displayed below.

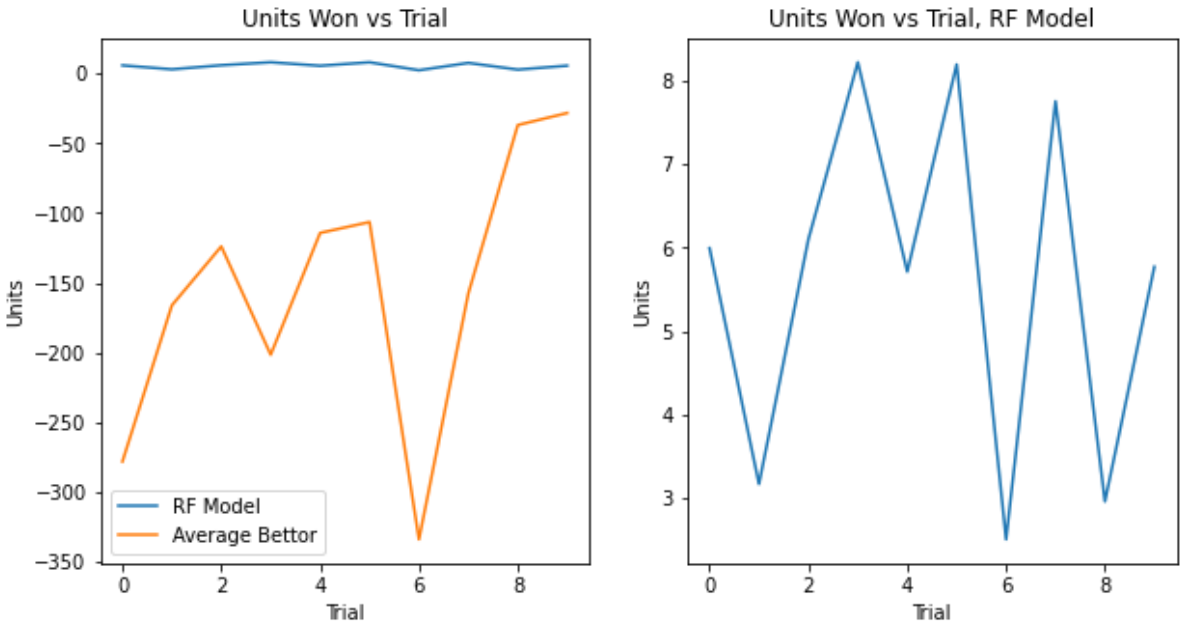

Figure 9. Units Won vs Trial

In these trials, the RF model ended with a gain of 56.40 units, whereas the random bettor ended with a loss of 1547.19 units. A difference of 1629.85.3 units is astounding, especially when it is noted that most bettors have a unit size greater than $1 and frequently bet more than one unit.

Bettor	Average Change In Units per Trial	Total Change in Units	Monetary Change (1 Unit = $10)
Random Forest	+5.64	+56.40	+$564.0
Random Bettor	-154.72	-1547.19	-$15,471.9

Table 5: Bettors and Monetary Change

Our claim of an average, random bettor losing money over time is backed by many studies, but is particularly seen in **Gainsbury, Russel**[2] where out of 65,890 bets on soccer, 73.99% of bets were losses. This study also discovered that in this large sample, the average gambiling return for one bet was -9.12 units. This study provides a statistical backing for the choices made to represent our average bettor.

The main takeaway from these tests are clear, using our RF model to predict match outcomes, to inform a simple betting strategy, our bettor never went down units in any season, and after 10 seasons was up a fair amount of units, whereas regardless of parameter choice, and shown in papers, the average bettor went down. Therefore, our model outperforms the average expected return in the long run.

What makes this RF model even more promising is that the betting strategy employed is incredibly simple, and could easily be adjusted to any specific gamblers preferences. To deploy

[2] Gainsbury SM, Russell A. Betting patterns for sports and races: a longitudinal analysis of online wagering in Australia. Journal of Gambling Studies. 2015 Mar;31(1):17-32. DOI: 10.1007/s10899-013-9415-4.

this model into the real world would be very straightforward, with a laptop, team stats, and publicly available betting lines, a bettor could use the RF model to inform their current betting strategy. Furthermore, the more times the RF model is used, the better it would become. In sports betting if one is correct in the long run any amount greater than 52% of the time, then they are considered to be an expert. Given that our RF model had a validation accuracy of 52.2% and a mean test accuracy in the trials above of 51% it is reasonable to state that the RF model in its current form is an incredibly good gambler.

Can we create a model to predict outcomes such that we can make consistent money gambiling, and beat an average gambler? Yes, as seen above our bettor made consistent money gambiling over 10 trials, never having a losing season. Furthermore, our model significantly outperformed the average gambler. Therefore we successfully answered the question we set out to solve.

5.Conclusion

By using various streams of data, our Random Forest model is able to accurately predict enough football matches such that the betting algorithm can make consistent money and beat the average gambler. In the process of conducting the analysis, important features of football clubs became evident: a team composed of players who are highly ranked at marking (tracking and defending an opposing player) and finishing (the accuracy of foot based shots inside the the penalty area) seem to provide the Random Forest with more confidence that they will win, so it is likely that these skills lead a team to being more successful. With solely looking at aggregated data from 11 countries and leagues, 1,400 football clubs, and 26,000 matches our model significantly outperforms the average gambler.