# Prediction of Football (Soccer) shot Outcome

Shashank Bangalore sbangalore@wisc.edu
Kelsey Buley buley@wisc.edu
Matt Voss mcvoss@wisc.edu

## Abstract

*Soccer is the most widely played and viewed sport in the world, with fans eager to find the outcome of shots. Matches are often uneventful with only a few shots and even fewer goals. Their importance in the outcome of a match is crucial in determining the winning and losing team. Having the power to predict which goal will or will not be made would be a great tool. With this knowledge, obviously one can predict which team will win which match. Here lies our motivation. We used a data set from a well-known and respected source on football, Statsbomb. We used the last 5 seasons, 2014-2019, (available in this dataset) from La Liga, the top Spanish football league. Much of the research in soccer analytics is predicated on predicting the outcome of matches. What separates this project from other related works is our focus on the most crucial moments of the game: shots. In this project we use Nearest Neighbor Classification, Random Forest, Histogram based Gradient Boosting Classification and Extreme Gradient Boosting Classification. We wanted to use a wide range of machine learning algorithms within the supervised spectrum to create a model that most accurately predicts the outcome of a football match.*

## 1. Introduction

Soccer is a game in which two teams of eleven players pass a ball mostly using their feet, but may also use any other body part besides the arms and hands. The objective of the sport is to dribble the ball and shoot the ball beyond the goal post plane to score a goal. There are various, non-discrete categories of players: attackers, defenders, and midfielders. The main objective of an attacker is to score, a defender to prevent scoring, and the midfielders handle duties between the attacking and defending thirds of the pitch. At a professional level, most matches produce only a few goals. This is one reason why football is such a difficult game to analyze via rudimentary statistical methods. During the match, the motivation of the players is to create opportunities that will allow them to make goal-scoring moves, through means of moving the ball, whether it be to a teammate or by taking shots at the goal.

With more and more teams demanding more out of their players, optimizing for every last drop of performance, the demand for advanced analytics has never been greater. Prediction can be implemented in many different areas. In this case, we are using prediction to benefit those who bet in sports, in particular soccer. We will be using Machine Learning methods to predict the outcome of individual shots.

### 1.1. Background

The usefulness of predicting soccer goals has a broad application for soccer enthusiasts, people betting on matches, and team managers. We believe that this model would be especially useful for team managers since our project is not as focused on the outcome of matches, but rather the outcome of individual shots. Our model will be useful for team managers seeking new talent based off game results.

### 1.2. Application

#### 1.2.1 Betting

An obvious application for the prediction of soccer match outcomes is sports betting. Betting is big; online betting is even bigger. Sites like RooBet and others allow users to place their money in hopes of gaining or, worse, loosing. Having the knowledge, or predicted knowledge, of what is supposed to occur would place the odds in your favor and allow you to reap maximum benefits.

#### 1.2.2 Recruiting

Another application of the predicted outcome of a soccer match would be recruiting. Recruiters are looking for the players who will bring their team the most goals. Our algorithm also includes expected goal by player, allowing recruiters to tap in and look at which players score, or are expected to score, the most goals.

#### 1.2.3 Fantasy Leagues

Another application of the prediction of soccer match outcomes would be the competition of fantasy leagues. Participants wish to select the teams who will score the most goals

and win the most matches. If they had the ability to guess which team would win the most amount of matches and score the most amount of goals, they would have an edge against the other players in their competition. This would allow the player to achieve maximum results and possibly win their competition.

## 2. Related Work

There has been extensive previous work done in this subject using Machine Learning. One such example is by Albina Yezus from Saint-Petersburg State University in 2014 [13]. Their goal, much like ours, was to predict the outcome of soccer matches. They took data from the English Premier League (while we took data from only La Liga). They also completed feature selection, choosing and deciding which features are most influential to the prediction of the outcome of the match, something we did not do. Their goals were to achieve 70 percent accuracy when predicting results of a game and to make a profit in betting. Our goal was similar, although we did not set a concrete number for accuracy. They used four algorithms: K-nearest neighbors, random forest, logistic regression, and support vector machine. They resulted in the random forest with the highest accuracy, with an accuracy of .63. K-nearest neighbors was second, with an accuracy of .55. They found the two most influential features were Goal Difference and Score Difference.

Another example is by Ben Ulmer and Matt Fernandez from Stanford University in 2013. [11] Their goal was the same as the previous project, to find which algorithm results in the highest accuracy when attempting to predict the outcome of a soccer match. Similar to Albina, this group used data from the English Premier League. They also participated in feature selection, something we did not do. They used Linear from stochastic gradient descent, Naive Bayes, Hidden Markov Model, Support Vector Machine (SVM), and Random Forest. They found that the Support Vector Machine gave the highest accuracy, an accuracy of .5. The second most accurate algorithm was Random Forest, an accuracy of .5. The third most accurate was Linear classifier, an accuracy of .48. For error rates among the models, Baseline had the highest, followed by Gaussian Naive Bayes, Hidden Markov Model, Multinomial Naive Bayes, SVM, Random Forest, Linear SVM, and One versus All. They found that every model under-predicted the number of draws, i.e. the expected number of draws was always lower than their true number of draws. They state it is from the fact that "Draw" is the least likely event to occur. This affects the predictor in some way, even though "Draw" is only 6% less likely to occur than "Win" or "Loss". They tried to correct this error but it only resulted in a model that predicted "Win" and "Loss" less accurately. They stated their model predicted accuracy of "Win/Loss/Draw" to around the same degree as a human analyst would. They believed, however, that their models are lacking compared to other Machine Learning models on this same topic. They blame this shortcoming on the randomness and lack of data. Their earlier seasons had less features than the more recent ones. Their data ranged from 2002 to 2013, so obviously there is some range in advances in modern technology present.

In conclusion, we can assume that a model produced by means of random forest will give us the most accurate result. If we were to implement an additional method, Support Vector Machine would also be another really good option; though, we did not implement it. Feature selection would have been another useful thing to implement. Our surplus of data and features made available to us may have allowed our model to achieve such a high accuracy. We did not have problems such as the lack of features or randomness of data that some other experiments had. This allowed us to create a very accurate model.

## 3. Proposed Method

Our proposed methods are K-Nearest Neighbors, Random Forest and Extreme Gradient Boost. They all were tested for accuracy and fit using mlxtend and scikit-learn.

### 3.1. K-Nearest Neighbors

K-Nearest Neighbors is a supervised machine learning algorithm. It predicts if a data point is in the same group as its surrounding data points. This "K" can be any number, usually odd. In most cases, the larger the number, the more accurate the results will be. It uses Euclidean Distance to measure similarity. [2]

### 3.2. Random Forest

Random Forest is an advanced form of the decision tree algorithm. The decision tree algorithm uses a parent node, which is then split into two child nodes. From there on, the child nodes are split again if it is deemed appropriate. It is used for categorical purposes, and can be implemented for continuous and discrete inputs or outputs. Random forest, then, is a complex combination of these decision trees. It is an ensemble method, meaning it is a group of these base models that each make their own prediction. [12]

### 3.3. Extreme Gradient Boost

Extreme Gradient Boost (also known as XGBoost) is similar to Random Forest in the sense that it is composed of many decision trees. It implements Gradient Boosting, which is a special type of boosting that uses a gradient descent algorithm. Gradient descent algorithms optimize by finding the local minimum of the differentiable function. Thus, this method of Machine Learning is robust. It attempts to minimize error, optimize accuracy, and reduce training time. [3]

### 3.4. Histogram-Based Gradient Boost

Histogram-Based Gradient Boost is similar to Adaboost since it uses many weak learners to make a strong learner. One key advantage to this algorithm over other boosting algorithms is its computation efficiency. [9]

Histogram-Based Gradient Boost aims to decrease the amount of time it takes to run the algorithm. It does this by use of feature-split histograms. It tries to find a spilt point in each of the features. Splitting the features reduces the run-time complexity.

One downfall of this algorithm is how it handles a large number of features. Histogram-Based Gradient Boost does not work well with many one-hot encoded features.

## 4. Experiments

The algorithms we chose to run are K-Nearest Neighbors, Random Forest, Extreme Gradient Boost and Histogram-Based Gradient Boost. All of the tests we use have cross validation to lower bias of the training set and ensure a more accurate result.

For this project we focused on Extreme Gradient Boost. We felt this algorithm would yield a model with the highest accuracy. We saw in homework examples K-Nearest Neighbors would not be an appropriate algorithm for our data set, as it gives a discrete result. The result we are looking for, or presume is most accurate, is continuous: a prediction of the number of goals a team will make in a prospective game. We want a discrete predictor which predicts the outcome of a shot.

One important caveat of the data we must include in our experiment is the number of shots that ended up being goals. About 14 percent of the shots in the data set were goals so if one was to just guess a miss every time they would be correct 86 percent of the time. Our goal in each of the algorithms is to exceed 86 percent test accuracy.

### 4.1. Dataset

The dataset we used is from Statsbomb, a football (soccer) data and analytics company. Our dataset consists of 4499 shots from 5 seasons of La Liga. There are many, many features in this dataset. We used the "outcome" category of data as our predictor, which lists the events of all attempted shots on goal as our target. The features included in "Shots" are: match_id, team, player, x, y, type, end_x, end_y, technique, aerial_won, follows_dribble, first_time, open_goal, deflected and Expected Goal. Type is one of the categorical features which is composed of free kick, open play, and penalty shots. Technique was another categorical feature which describes how the player scored or missed the goal. Our main focus is on outcome. This tells if the shot was: "Blocked", "Goal", "Off-T", "Post", "Saved", "Wayward", "Saved Off T", and "Saved to Post". "Blocked" means a defender blocked the shot before entering the net. "Goal" means the shot entered the net and thus the team scored. "Off-T" means the shot went beyond either side of the posts. "Post" means the shot hit one of the posts. "Saved" means the shot was saved by the other team's goalkeeper. "Wayward" means the shot was either too weak or nowhere near the goal and was deemed not a threat. "Saved Off T" means the shot was saved by the goalkeeper but was not a threat. Finally, "Saved to Post" means the goalkeeper saved the shot and it hit off of one of the posts.

For the categorical features we used a pandas function, 'to_dummies', to transform the data into numerical columns where 1 is true for an example if the feature is at play or 0 otherwise. The Technique and Type were also similarly split between true and false creating a new row for each unique category. When splitting the data into training and test sets we split them by .5. We we wanted to keep the over fitting as constrained as possible while also proving our models with enough information to train them on since the distribution of goals and misses was so different.

We modified a Statsbomb json scraper created by Devin Pleuer, analytics director for Toronto FC in order to accommodate for more peripheral data describing the conditions surrounding the shot than a typical model just taking into account the position of the player. [4]

### 4.2. Software

The models developed in this project were crafted in Microsoft Visual Studio Code. The version of Python we used is 3.9. The Python libraries we used were numpy and pandas, of which were used to clean the data. We also used mlextend and scikit-learn in order to utilize Machine Learning algorithms.

### 4.3. Hardware

The programs were run on each person's laptops, of which all include MacBook equipped with MacOS Catalina version 10.15. and ARM equipped MacBook.

## 5. Results

### 5.1. K-Nearest Neighbors

KNN had the worst test accuracy of the algorithms used with a little less than the base test accuracy of 86 percent. To combat this poor test score, we reduced the dimensions of the data by first using Principal Component Analysis to transform the features. In figure 1, one can see that the number of components after 4 has a poor return so we decided to use 4 components.[1]

After finding the optimal number of principal components, we use Scikit Learn's Neighbor Component Analysis to transform the features. With the features transformed we rerun Grid Search Cross Validation to tune KNN. In figure

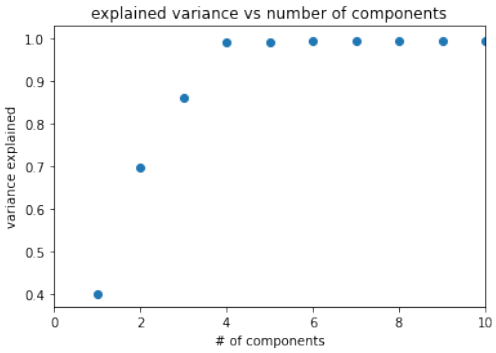


Figure 1. Explained Variance of the features with each addition of components.

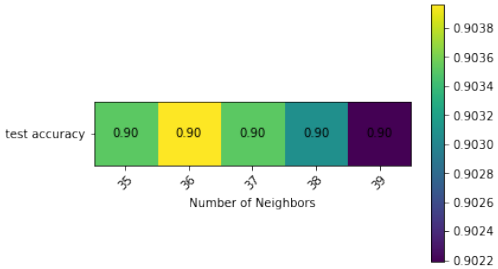


Figure 2. A heat map showing our final step in tuning the number of neighbors to use with KNN.[8]

2, one can see that 39 neighbors was the optimal number to generate the highest test accuracy of 87.8 percent. By first selecting number of neighbors from 10 through 60 increasing by 10 we narrowed the number of neighbors down to 39 through trial and error. Fortunately, the test accuracy increased by about 2 percent, but the test accuracy is still poor. This is pretty poor, since this beats guessing missing every time by just 2 percent.

### 5.2. Random Forest

Random Forest produced a much higher test accuracy than KNN with no transformations required. First, we used class weights on this algorithm since the number of goals was considerably lower than the number of shots becoming a miss. By using Grid Search Cross Validation to once again search for the optimal hyper parameters, we tuned two hyper parameters within sci-kit learn's random forest classifier, criterion and number of estimators. We also set the cross validation to 10 to ensure a more accurate test accuracy. In figure 3, one can see that there is not much of a difference between the number of estimators within this range, but there is quite a difference between Gini and Entropy. With number of estimators at 305 and criterion of entropy giving us the highest test accuracy for Random Forest at 92.71 percent.[6]

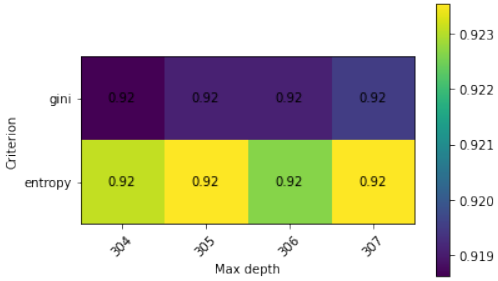


Figure 3. A heat map showing our final step in tuning max depth and criterion for random forest

### 5.3. Histogram-Based Gradient Boost

Histogram-Based Gradient Boost was tuned using maximum depth and minimum sample leaf. By establishing a maximum number of splits between the nodes, we ensure that there is not too much over-training in the model. If we left this at its default of none, the model would make more negligible splits and would not represent the overall distribution of our target so it would not generalize as well. The minimum sample leaf has a similar goal of reducing overfitting to ensure that our model generalizes well on our test set. By establishing that a split must have at least n number of samples in the split, the training set will not put a minuscule number of samples onto a leaf. While we were making our preliminary investigation into our model, we thought it would be helpful to find a rough idea of maximum depth and minimum sample leaf so that we do not have to use trial and error on the later stage of model selection which is more expensive to conduct. Our optimal hyperparameter for Histogram-Based Gradient Boost was a maximum depth of 3 and minimum sample leaf of 35 and a test accuracy of 93.91 percent. Although this test accuracy is still higher than the test accuracy for the other models, this had a much higher training accuracy of 96.35 percent. This is a classic warning of overfitting and we think that other algorithms will be less biased and closer to the true accuracy.[7]

### 5.4. Extreme Gradient Boost

We used both XGboost and XGboost as the function with SciKit-Learn's bagging algorithm. We decided to use a bagging algorithm in this case as in a typical match, the number of shots on goal is a relatively low number. For example, in the 2018-2019 season, Barcelona led with having 6.7 shots on goal on average per match [2]. We also decided to use a weighted classifier due to similar reasons due to the relatively small number of goals scored in our dataset. As can be seen, the $weight = negatives/positive$ examples where negative examples were shots missed, and positive examples corresponded to shots resulting in a goal. Our weight ultimately corresponded to the approximate value

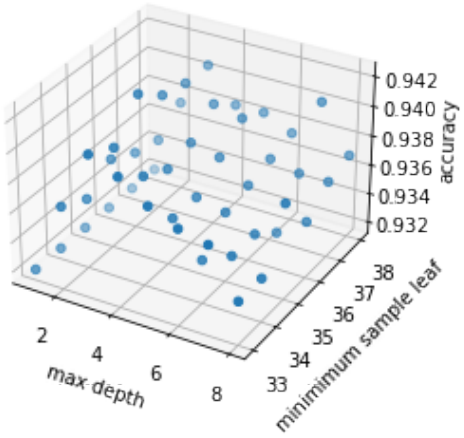


Figure 4. Max depth and minimum sample leaf were used to tune the hyper parameters on the Histogram Gradient Boost

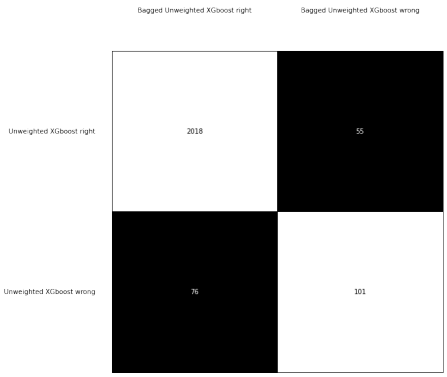


Figure 5. A confusion matrix describing the difference between bagging and not bagging for the unweighted Extreme Gradient Boost

6.49. Both of these modifications allow for the model to more accurately account for the low volume of shots taken on goal. We also decided to run the models in an unweighted fashion, which allows us to determine the effect which weighting has on the accuracy of our models, and to truly see if weights help in compensating for the imbalanced ratio of $shots : goals$. We optimized the hyper parameters on number of estimators and used the range of 5 to 10.

### 5.5. Bagging

Bagging methods are ensemble methods which produce great results using weak learners. Bagging methods work by training data on a small subset of the entire training dataset (which is randomly chosen). This allows weak learners to be aggregated in order to produce a stronger model (law of large numbers). This notion is natural for

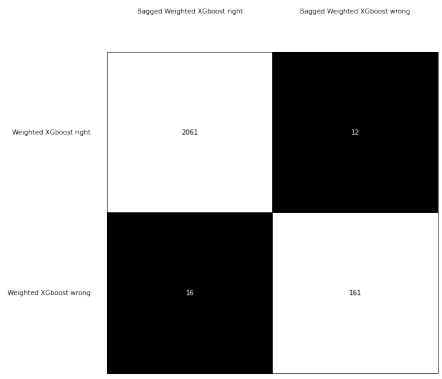


Figure 6. A confusion matrix describing the difference between bagging and not bagging for the weighted Extreme Gradient Boost

training data in games such as football where the number of attempts on goal and goals is relatively small compared to other sports. We can see this in action by observing the relatively high accuracy of the weighted training model. We can observe that bagging vs no bagging produced a statistically significant result. We used bagging techniques in conjunction with the Extreme Gradient Boost to harness the power of weak learners.

## 6. Conclusions

After work on the algorithms, Histogram-Based Gradient Boosting had a fairly high test accuracy with 93.91 percent. Although this score was higher than the other algorithms, the others used class weights to balance the data set. Random Forest also seemed to be a strong contender for our best model, because of its strong test accuracy and the training accuracy was not too high. Ultimately, we decided to use Extreme Gradient Boost as our final model, because of its high test accuracy and low relatively similar test accuracy. Within Extreme Gradient Boost, we produced a few models with subtle differences such as weighting vs not weighting the number of shots and bagging versus not bagging to find the optimal version for the Extreme Gradient Boost algorithm. Both of the test accuracies were incredibly close so we decided to run a McNemar test to see if the differences between bagging and not bagging was significantly different. When testing for the difference between bagging and not bagging we found that there was a significant difference between bagging and not bagging for the Extreme Gradient Boost model with no weight with a p-value of 0.12. This proves there is a marginally significant difference between the bagging and no bagging for no weight used. When comparing with weights there was not a statistically significant difference between Extreme Gradient Boost with weights with a p-value at 0.45. [5]

We faced numerous challenges while compiling this data. Our first, and possibly most difficult challenge was deciding on a parameter to optimize towards. Due to the low degree of positional restrictions (only the offside rule), the lack of rigid positions, unlike other sports, rapid shifts in tactics and team shape, and the continuous nature of the game makes it difficult to find a single parameter that accurately represents the nuances of the game. Realizing this would be a complex and costly task, we instead decided to shift our focus on anticipating when a shot would be successfully converted into a goal. This, while providing significantly less information about a team's style and performance compared to other features, would work as a valuable classifier which requires significantly less computational power. One of the main metrics used in how well a player performs would be the Expected Goals (XGoals) metric. This is a regression based method which assigns a value from 0 to 1 to a player on how often a player will score goals from that position on the field. "Position" is determined by breaking up the field into a discrete $n \times n$ grid, where probabilities to score are calculated using regression analysis for each rectangle on the grid. XGoals is purely based on a player's position on the field and not any other "intrinsic" qualities of a player.

Dimensional reduction was helpful in improving the test accuracy for the

For smoother future research, feature importance could be analyzed. We used all the features in our testing algorithms. The number of features could be reduced. This can easily be done and simplify the models.

Features could also be tested for correlation. Highly correlated variables could be useful to know. Those variables that are not very closely related could be deemed not useful.

## 7. Discussion

After working with the data more within the last few weeks of the semester, it was difficult to learn about algorithms, research them, and try to implement them into our project. We thought we were partially constrained to working with just the type of algorithms used and presented in class and the algorithms implemented may not have fitted our goal of the project as some of the other machine learning algorithms. Our group also struggled to learn Python and its syntax since most of the group were more familiar R and Java. The biggest hurdle for our group was importing the data into a data set that was usable for our group.

Our first attempt to access and clean the data was to clone the data set which had millions of rows of data and took hours and hours to load and concatenate the soccer matches into a data frame. We spent most of the time in October and early November trying to work on cleaning the data which we had cloned from the Statsbomb repository. Fortunately, we found Devin Pleuer's, Toronto FC's analytics head, who offered a Statsbomb scrapper and modified it for our use. Besides our technological constraints, the data was also limited in access. Statsbomb offers a free and a paid version and we only used the free data within just La Liga. Although Statsbomb provided us with access to a fraction of the data it would have been better for our model to have access to more date since we only implemented supervised machine learning algorithms.

We were also unable to get the Logistic Regression to work properly with our data. We asssumed Logistic Regression would work well since all our data was numeric and our predictor was binary. Unfortunately sci-kit learn's algorithm was not cooperating and we decided to move on to other algorithms.

If we could return to the project with more technological know-how, we think the project could be improved by implementing bootstrap and .632+ to limit the bias we had for our test accuracy.

In the future we would like to expand our area of investigation outside of just shots and use more features in a model. One useful feature for analyzing soccer is expected threat which measures individual players on an event level and goes beyond goals scored. For more information follow this article describing expected threat. [10]

Take note that the p-values in our McNemar test for comparing the XG boost have been changing after fitting the model more than once. In the future we would like to fix this problem with either our random state or some other part in our model that is changing the p-value.

## 8. Acknowledgements

Each of the team members, besides Shashank, had no previous knowledge of or in Machine Learning. Matt and Kelsey both had no previous knowledge of Python coding. Without the lectures, teachings, notes, and slides from Professor Sebastian Raschka, we would not have any direction in which way to go. He has taught us extensively and gave us an operation knowledge of Machine Learning algorithms. His guidance and suggestions have helped us understand and improve upon our project.

We also give acknowledgement to online sources such as DeepAI and Towards Data Science, as they provide great, usable information on Machine Learning concepts, which proved useful in our journey through this project.

The scrapper provided by David Pleuer was key for our data set by providing us with easy access to clean data.

## 9. Contributions

Each team member contributed to all parts of the project:

1. Shashank

   (a) Responsible for writing the code for the Extreme Gradient Boost algorithm
   (b) Solving overall technical and coding issues
2. Matt
   (a) Responsible for writing the code for the Histogram-Based Gradient Boost, Random Forest, and k-Nearest Neighbors algorithms
3. Kelsey
   (a) Responsible for writing most of the report
   (b) Responsible for brainstorming ideas

## References


[1] Dimensionality reduction with neighborhood components analysis.

[2] O. Harrison. Machine learning basics with the k-nearest neighbors algorithm. *Towards Data Science*.

[3] V. Morde. Xgboost algorithm: Long may she reign! *Towards Data Science*.

[4] D. Pleuer. Data extraction and transformation.

[5] S. Raschka. Contigency table for mcnemar's test.

[6] S. Raschka. Ensemble methods part 1/2.

[7] S. Raschka. Ensemble methods part 2/3.

[8] S. Raschka. Model evaluation 3– cross validation and model selection.

[9] P. L. Saint. The many flavors of gradient boosting algorithms. *Data Iku*, 2020.

[10] K. Singh. Introducing expected threat.

[11] B. Ulmer, M. Fernandez, and M. Peterson. *Predicting soccer match results in the english premier league*. PhD thesis, Doctoral dissertation, Ph. D. dissertation, Stanford, 2013.

[12] T. Wood. What is a random forest? *DeepAI*.

[13] A. Yezus. Predicting outcome of soccer matches using machine learning. *Saint-Petersburg University*, 2014.