International football results

Mihai Matei
Exploratory Data Analysis Course
Data Science Master, 2019
University of Bucharest, Faculty of Mathematics and Computer Science

Abstract

This paper tries to apply machine learning and statistic methods to identify the total number of goals in international football matches. Two types of regression models are being compared, namely linear models in different flavors against tree-like models such as decision trees. For linear models gaussian basic model is used along with AIC and BIC information criteria methods as well as generalized linear model with L1 and L2 penalization. For tree-like regression simple decision trees with pruning and ensemble methods of the above such as bagging, random forest, boosting. The results will be presented as well as the problems encountered.

1. Introduction

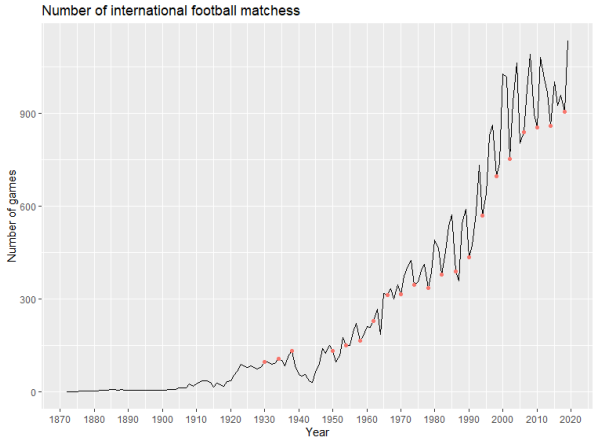
Football is probably the best known sport in the world that attracts a lot of media attention, fans and, of course, money. Though most people are interested in the win or loose of a specific match in order to use the prediction in a gambling scheme this paper tries to identify the number of total goals that are being scored in a match. Since this is clearly a machine learning regression tasks the classical linear model, with different flavors, is being compared with the family of tree-like algorithms, such as decision trees. I will focus on comparing the standard regression metrics such as R squared metric and mean squared error. All models were implemented in RStudio and are being run on a custom data set created by joining 3 different information sources. The football matches I am trying to predict are between national teams in different kind of tournaments. An important aspect of the data model is that the matches are spread though the years and there is no more than one or two matches per year between two different national teams. I found that the feature with the biggest relevance seems to be the rank of each individual team in the match as well as the continent where the team is part of.

2. International football data set

Prediction a football result is hard due to the low amount of data that exists between same team matches in a reasonable period of time. The [1] data set contains 41540 football results between 1972 and 2019 of national country teams. The data set columns are date home_team, away_team, home_score, away_score, tournament, city, country, neutral where city and country represents the location where the match took place and neutral if it was a friendly match or not. The first record of this database contains a match between Scotland and England played in Glasgow, Scotland the finished with no winner:

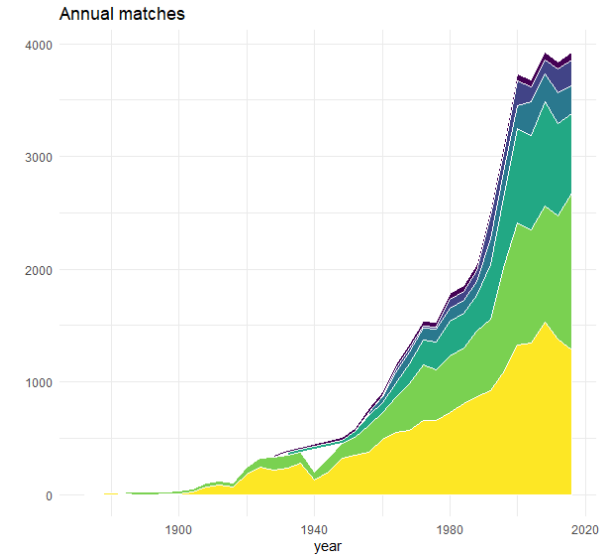
1872-11-30 Scotland England 0 0 Friendly Glasgow Scotland FALSE

To see the adoption of this sport a plot of number of games per year is especially useful and since the teams involved are national teams the red dot represents the year when World Cup took place.

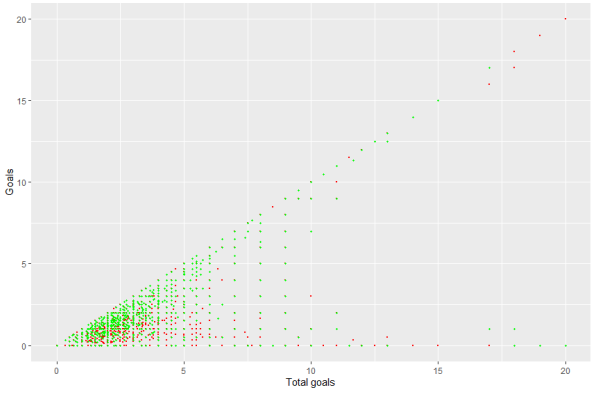


We can see that there are less that 100 matches per year before 1900 and peeks at a bit over 1000 matches after 2018. Another interesting plot is the number of matches per year per type of tournament. Below I select the top

5 tournaments with the most matches and there evolution in the number of matches played per year:



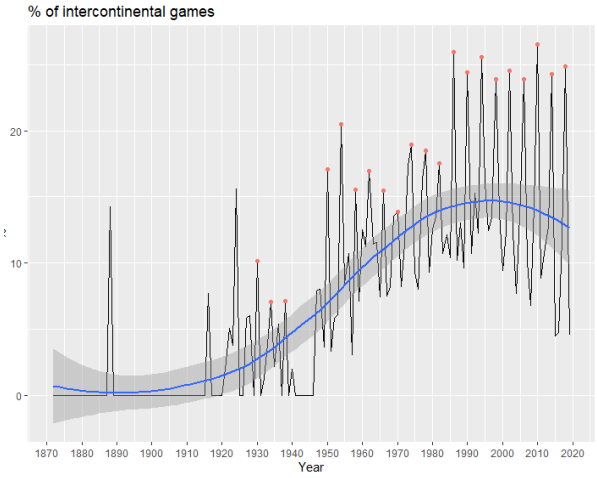
Other entry in green contains the sum of all other type of tournaments and is the just after the Friendly type which represents training matches. Since I am interested in predicting the number of total goals that are scored in a game another useful plot is the distribution of the home or away goals by the total number of goals between 2007 and 2020:



Green represents home goals and red the away goals. One can notice that the home goals have an upper larger triangle distribution than the away goals suggesting a team usually wins matches that are being played on their home field. Another interesting aspect of this plot is that we can identify that the total goals scored in a match is roughly between 0 and 8 meaning games that have more than 8 goals can be treated as outliers.

2.1. Augment by joining new data sets

Unfortunately the linear models that are being tried do not really know how to correlate relative results between teams in the same league and to make a quantitative ordering between teams. Example a linear model cannot infer that if team A beat team B and team B beat team C team A would most likely beat team C. Applying the linear models on the above data set alone achieved poor performance in identifying the number of goals scored. To address this problem we join the original international results data set with two additional data sets. The first data set [2] that will joined contains the list of national teams or countries affiliated to a federation. The database contains a list of 6 federations namely UEFA, AFC, CAF, OFC, Concacaf and Conmebol. I will map each federation to the continent that they belong to. For example UEFA maps to Europe, CAF to Africa and the last two to America. I the add 2 new columns to the original data set, home_continent and away_continent by doing a inner join on the 2 data tables using R dplyr package. Adding this 2 new columns augments the data set by identifying the football matches played between teams in different continents. The percentage is shown below:



The last data set [3] that will be merged contains a lot of information about the evolution of national team ranks between years. It contains a total of 57793 entries of 16 columns of rank information. I select only the national team name, rank and rank year making sure that we only have distinct records to filter out multiple ranks of the same team in a given year. Than I do an inner join with the original data set to create 2 new columns home_rank and away_rank. This yields only

17857 records because some teams might not have a rank for the year when they played a match. Now the augmented data set has the following entries

date home_team away_team home_score away_score
1993-01-01 Ghana Mali 1 1
tournament city country neutral home_federation
Friendly Libreville Gabon TRUE CAF
home_continent away_federation away_continent
Africa CAF Africa
home_rank away_rank
48 74

2.2 Feature selection and data processing

The first thing we do before choosing our feature and target set is to take only the matches played after year 2000 that have less then or equal to 8 total goals scored in a match. This way I remove the outliers given by the number of goals scored and take a date period where the data set contains more games that have been played. I also take only the top 20 tournaments and group the rest of them in an "Other" tournament type as such the tournament column factor type will have only 21 types. The chosen features are:

feature_home_continent = as.factor(home_continent)
feature_away_continent = as.factor(away_continent)
feature_tournament = as.factor(tournament)
feature_date = date - start_date
feature_home_rank = home_rank
feature_away_rank = away_rank
feature_home_field = (country == home_team)

where the start_date is the date of the first match in the filtered data set and feature_home_field is a boolean value indicating if the match played on the home's team field. The target is chosen to be the log of the total goals scored, due to the fact that regression seems to give much better metric results if the target is smoothen by the logarithm (1 is added to avoid infinity for games with no goals scored):

target = log(home_score + away_score + 1)

A train-test split is done choosing the test set to be 0.1 percent of the data. The validation set, if needed when choosing the hyper parameters of some models, will be chosen automatically from the train set. I also check that the resulting data set has no NaN or missing entries. Train set will contain 12407 entries while test set only 1379.

Each resulting train or test set is a named R list that contains 3 fields: data – the R dataframe format of the set, X and Y the same dataframe converted to matrix format for the features (X) and target (Y) because some library models require the above matrix format.

3. Regression metrics and utilities

In order to asses model regression performance across models of different type implemented in different R models functions to measure mean error rate, R squared coefficient was implemented as well as using the plotmo R library to show some information about residuals in different models.

Mean squared error is the easiest regression metrics that measures the total error of the prediction using:

mean((predicted - true) ^ 2)

R squared or coefficient of determination [4] measures the variance in the predicted target values that can be explained by the independent features. It can be seen as the variance in the predicted values as opposed to the variance in the true values:

$$r^2 = 1 - \frac{\text{SS Error}}{\text{SS Total}} = 1 - \frac{\sum(y_i - \hat{y}_i)^2}{\sum(y_i - \bar{y})^2}$$

One interesting fact about the R squared is that the if one adds a new feature to the model R squared always increases. To address this the adjusted R squared is used that takes into account the number of features that the model is given. Another fact that I had to take into account is R squared value for a linear model with no intercept. As [5] explains in this case R silently omits the expected true predicted value from the formula.

For most of the models I will use plotmo library to plot different measures of the residual of the model such as Residuals vs Leverage, Residuals vs Fitted, Cumulative Distribution, Residual QQ. They are best explained in [6]

3.1 Parallel computing using R clusters

Because training linear models takes a a lot of time, and even if tree-like models or ensemble methods are inherently parallel some libraries do not implement any form of parallelism. To overcome this limitation cluster API was used from the parallel R library. A R cluster is basically a new R process that can share part of the memory with the parent process. For this we use the *clusterExport* method to share the some of the data such as the training set or the formula for the model configuration. I then train each model in a separate process in parallel (check the *parralel_compute* parallel model training implementation). Since I can take advantage of the number of logical cores of my local processor now the training process for several model takes on average the maximum training time of all the models.

4. Linear models

Linear models are the most well know and studied data science statistical models. Since their output can be statistically determined to have a specific amount of variation it is one of the most used models when you have to justify your predictions. Hence the phrase *"the most money are run on linear models"* being widely used by investment or pension funds. On the selected train set we apply 5 types of linear models: standard gaussian linear model from the olsrr (Ordinary Least Squares) R library, then apply AUC and BIC information criteria on then and finally use generalized linear models with L1 and L2 regularization from the glmnet R library. The total training time for the 5 models in parallel is under 2 seconds.

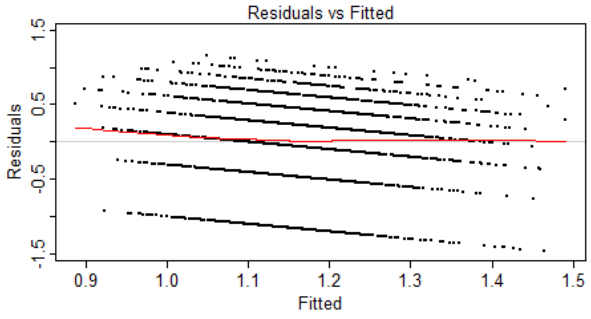
4.1 LM OLS results

As the OLS name suggests the linear model in this case is constructed using the standard least squares lost that can be statistically shown that it leads to a prediction Y random variable that has a Normal / Gaussian distribution. To construct the model we use a R formula *target~.-1* because we want the model not to have an intercept since I am interested in the linear model regression line to go through 0 (in our case 0 si log(1) and remember that *target = log(home_score + away_score + 1)* and hence a 0-0 score will produce this result). Training the model produces the following:

Residual standard error: 0.5187 on 12374 degrees of freedom
Multiple R-squared: 0.8342, Adjusted R-squared: 0.8337
F-statistic: 1886 on 33 and 12374 DF

The 12374 degrees of freedom in the F statistic comes from the number of features that the model has compared to the null test (here the linear model with no model parameters). It is computed as *N-P-1* as best explained in [7] where N is number of observations, 12407 in our case, and P the number of features, 5+5+21+3 since each factor type has x number of features where x is the number of factor levels.;

Plotting the Residuals vs Fitted of the linear model



shows that the residuals are equally distributed on both sides around a horizontal line seems to indicate that the data does not have any non-linear relationship.

4.2 AIC and BIC information criteria

To improve on the results by adding and removing predictors R offers the *step* function that can be used to generate models using the AIC [8] (Akaike Information Criteria) and BIC [9] (Bayesian Information Criteria). The statistical criteria they follow is to minimize the error against the null test by limiting the number of predictors.

The AIC results:

Residual standard error: 0.5187 on 12378 degrees of freedom
Multiple R-squared: 0.8341, Adjusted R-squared: 0.8337
F-statistic: 2146 on 29 and 12378 DF

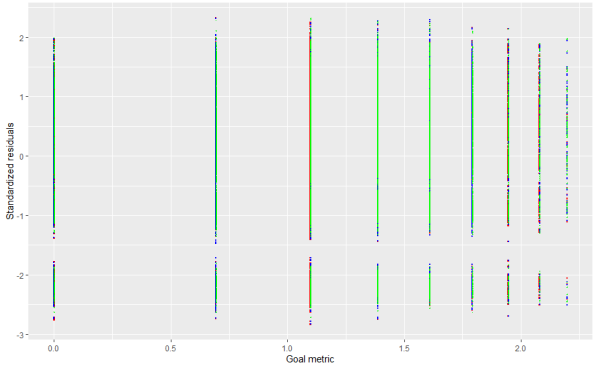
The BIC results:

Residual standard error: 0.5198 on 12399 degrees of freedom
Multiple R-squared: 0.8331, Adjusted R-squared: 0.833
F-statistic: 7736 on 8 and 12399 DF

Notice that the number of used features (-1) and degree of freedom changes in all cases from 33 in the LM model case to 29 for AIC and 8 for BIC (near the F-statistic summary).

Unfortunately there is no big difference in the R squared model parameters, but less features means a much more simpler and computational efficient model.

The standard residuals (the residuals divided by their standard deviation) are plotted for the 3 models:



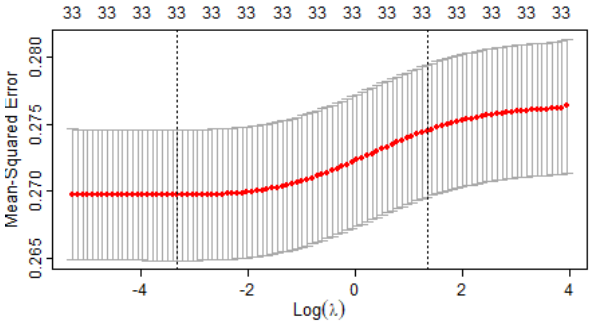
4.2 Glmnet's Lasso and Ridge models

Glmnet [10] is a package that fits a generalized linear model via penalized maximum likelihood. Glmnet solves the following equation:

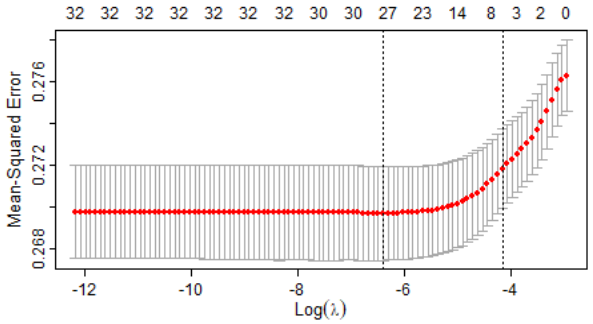
$$\min_{\beta_0,\beta} \frac{1}{N}\sum_{i=1}^{N} w_i l(y_i, \beta_0 + \beta^T x_i) + \lambda\left[(1-\alpha)||\beta||_2^2/2 + \alpha||\beta||_1\right],$$

One can see that if alpha is 1 this is equivalent to L1 Lasso regression if alpha is 0 this is instead Ridge L2 regression. Because the model depends on the lambda

hyper parameter we use cross validation to find best value of lambda and use that value for both regularized models to compute the metrics. For L1 lasso we have



and for L2 ridge regression the plotted error is:



The minimum error is for lambda 0.001662157 for L1 and 0.03665273 for L2.

To get the full picture for the training metrics the 5 linear models have we compute the MSE and R2 for each of them (on the training set):

	R2	MSE
LM	0.8334899	0.2694223
AIC	0.8334354	0.2695105
BIC	0.8323754	0.2712256
LASSO	0.8333677	0.2696200
RIDGE	0.8334095	0.2695524

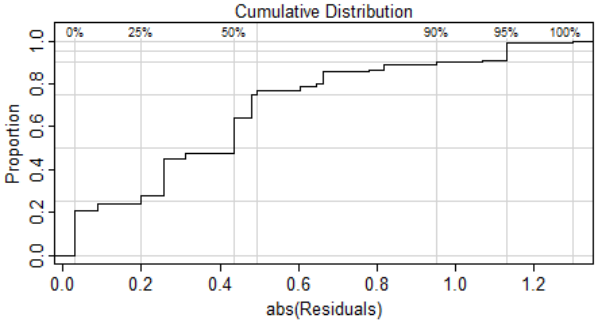
Notice that all models have equal performance on the training set, the differences are small. This might show that the data set has some clear linear characteristic that is approximated by all models.

5. Tree-like models and ensemble methods

Tree like models are combinatorial linear models that divide the data set into areas according to some statistical information such as the information gain by splitting the space by a feature plane. This information gain can be both entropy information (sometimes called deviance which differs from entropy by a factor of 1/N) or GINI index. The advantage of using tree like methods is that they work pretty well for out of the box data with no feature selection. Their disadvantage is that they do not really generalize well for data that is out of the learned trained space.

5.1 Tree model

The standard tree model is the decision tree which in R is implemented in the tree package. Trees are prone to over fitting the trained data since they try to split the trained data to achieve the best deviance. To address this I use cross validation and choose the model with the minimum deviance. The resulted tree is pruned to have the same feature number of splits that was identified in the best deviance cross validation.
The best tree is found to have only a split node and 2 leaves and the split feature is *feature_away_rank*. Below you can see the size of the residuals of the models as percentage of all residuals.

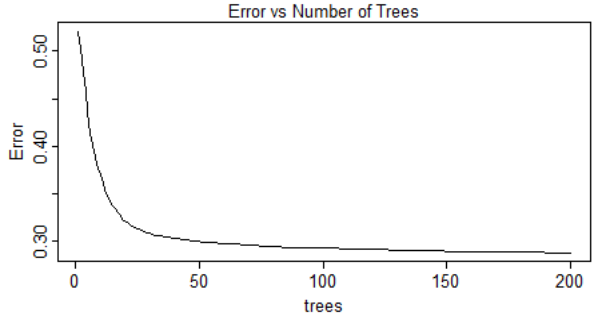


Notice an almost linear line meaning the equal number of residuals for the different error sizes.

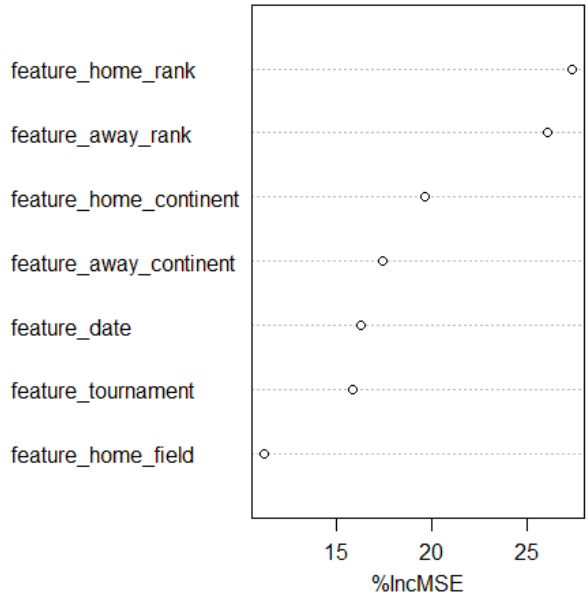
5.2 Bagging and Random Forest

Boostrap aggregation or bagging is a standard technique in machine learning to reduce the variance of a learning algorithm by training the model on different slice of the data and averaging the results. To test this method on our data set we used the randomForest R library where we set the number of features to be used to be considered for a split at a tree level to be the same as the number of features in our data set.

Bagging is also an ensemble methods because multiple trees are constructed against slices of the trained data. To do some sort of cross validation we try to check the model MSE on a different number of trees.

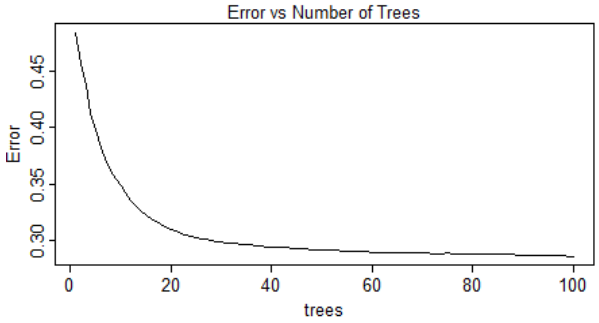


We can see that the total error of the models decreases very slowly after 100 trees are used so there is no point is searching larger amounts of trees. One other important aspect is to see which features are best used in the splitting process and leads to the higher information gain:

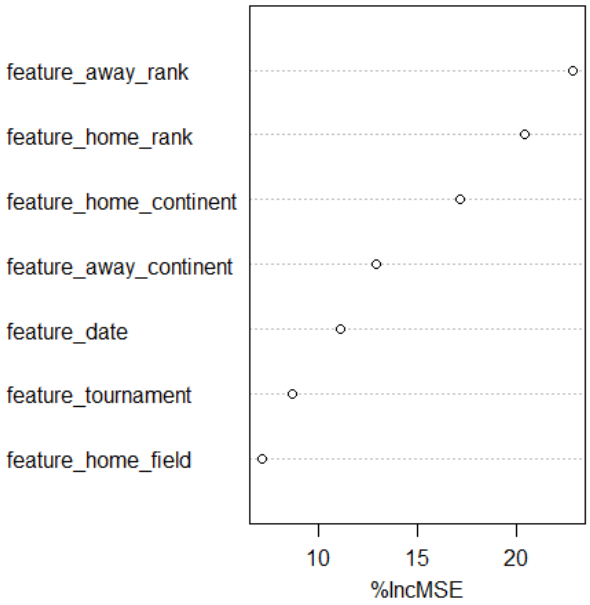


As expected the team ranks are the best predictors for the bagging tree.

Random forest further try to improve the variance of the model by trying to reduce the correlation between features. The best difference between bagging and random forest is that in the case of the latter model not all features are used in the splitting process. For each tree in the ensemble methods the algorithm only chooses random features from a subset of features, which in our case is set to be the feature set divided by 3. To do a bit of cross validation we also try to vary the number of trees to be used in the model.



As in the case of bagging we notice that the error decreases very slowly after 60 trees. This means that random forest require somewhat less number of trees to achieve an error rate similar to the bagging mode. I also plot the feature importance for the random forest:



As in the bagging case the rank features are the best predictors.

5.3 XGBoost

Boosting is a very popular machine learning algorithm that achieved best results in some of the newest Kaggle competitions. The model is best explained in [11] and improves the booting technique by applying a gradient descend algorithm to select the new data to train on based on previous runs. Xgboost is an algorithm that also applies regularization and other techniques to fit the data and can be trained in parallel.

The result of all tree models on the training test is shown below:

	R2	MSE
TREE	0.8304741	0.27394652
BAGGING	0.9643205	0.05765652
RANDOMFOREST	0.9526619	0.07649634
XGBOOST	0.8389108	0.26031320

Notice that on the training test the Tree and Xgboost model has an MSE close to the linear models but the random forest models outperform all the other.

6. Conclusions

To test the actual models we need to do it on a data the models have not seen yet. This is the only way to analyze and compare the models between each other and to measure how much over fitting is done for each one of them. The results on the test data set are:

	R2	MSE
LM	0.8342803	0.2645889
AIC	0.8343658	0.2644523
BIC	0.8338633	0.2652546
LASSO	0.8343147	0.2645339
RIDGE	0.8342556	0.2646282
TREE	0.8324718	0.2674762
BAGGING	0.8244099	0.2803480
RANDOMFOREST	0.8266280	0.2768065
XGBOOST	0.8340421	0.2649692

The interesting fact is that even if the train data set suggested that the tree models might perform better there is no significant difference between all the models. This suggest that perhaps the models only learn to use only a little subset of features, home_rank and away_rank being perhaps the most informative.

7. References

[1]https://www.kaggle.com/martj42/international-football-results-from-1872-to-2017

[2]https://www.kaggle.com/phjulien/national-football-team-affiliations

[3]https://www.kaggle.com/tadhgfitzgerald/datasets

[4]https://medium.com/analytics-vidhya/r-squared-vs-adjusted-r-squared-a3ebc565677b

[5]https://stats.stackexchange.com/questions/26176/removal-of-statistically-significant-intercept-term-increases-r2-in-linear-mo

[6]https://data.library.virginia.edu/diagnostic-plots/

[7]https://boostedml.com/2019/06/linear-regression-in-r-interpreting-summarylm.html

[8]https://en.wikipedia.org/wiki/Akaike_information_criterion

[9]https://en.wikipedia.org/wiki/Bayesian_information_criterion

[10]https://web.stanford.edu/~hastie/glmnet/glmnet_alpha.html

[11]https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d