Capstone Project Philip Burroughs
Machine Learning Engineer Nanodegree April 27th, 2019

Predicting Higher Scoring European Football Matches

Definition:

Project Overview

Football is a sport loved around the world and has the power to unite a nation of people together, even if only for a month. I have felt the ups and downs and seen some truly amazing storylines unfold in front of me over the course of 90 minutes. Football is only the world-wide phenomenon it is today because of its gargantuan fan base. With the sport often being plagued with negativity in the media and some reports have shown a dip in viewings from time to time [1], this made me worry that one day it might not be globally loved like it is today.

The problem this report will look to solve is how to bring new fans into the sport and how to keep the current fans as excited as ever for the beautiful game. Often the most exciting games, with the best storylines, will have lots of goals in them. With a limited amount of games being show publicly on TV it is essential that the most exciting games are shown to help bring new fans into the sport.

The origins of the data set used in this report is publicly accessible [2] and contains the historic half time & full-time results of every game from the 2012-2013 season to the 2016-2017 season for the top divisions in Europe. Some additional work has been applied to the data pulled from the website too, this additional work has added the average goals for each team, at each instance for each unique season as well as the under / over 2.5 goals form for each team in the last 5 games.

Problem Statement

To help fix the problem of football attracting more fans, this report will aim to build a machine learning algorithm that can identify the game's most likely to have 3 or more goals in. Once the potentially high scoring games have been identified this information can then be used to decide what games should be shown on TV and ultimately help to bring more fans into the sport.

The machine learning algorithm must successfully identify games that ended with 3 or more goals in them, with no preference to which team scores the goals. Multiple algorithms will be tested and will undergo parameter tuning to help increase the percentage of high scoring games identified.

Metrics

The two main scoring metrics that will be used to assess & compare the models will be accuracy and precision.

Accuracy is the percentage of correct predictions made of all the predictions [3]. Whilst this sounds logical, accuracy alone does not always tell the entire story. If there was a heavy bias towards one event result, for example if the majority matches ended with 2 or fewer goals then the model can just predict 2 or less goals for every match and the accuracy would return a good score.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

TP = True Positive, TN = True Negative, FP = False Positive, FN = False Negative

Precision shows what percentage of predictions that were highlighted as a positive were positive [4]. This metric will prove to be more valuable for what we are trying to achieve in this project. The aim of this project is not to predict the outcome of every single game correctly but instead find games that we believe will end with 3 or more goals. This means that we are only interested in the games that are predicted to be high scoring and we want these predictions to be correct, even if it means predicting less games as high scoring.

$$Precision = \frac{TP}{TP + FP}$$

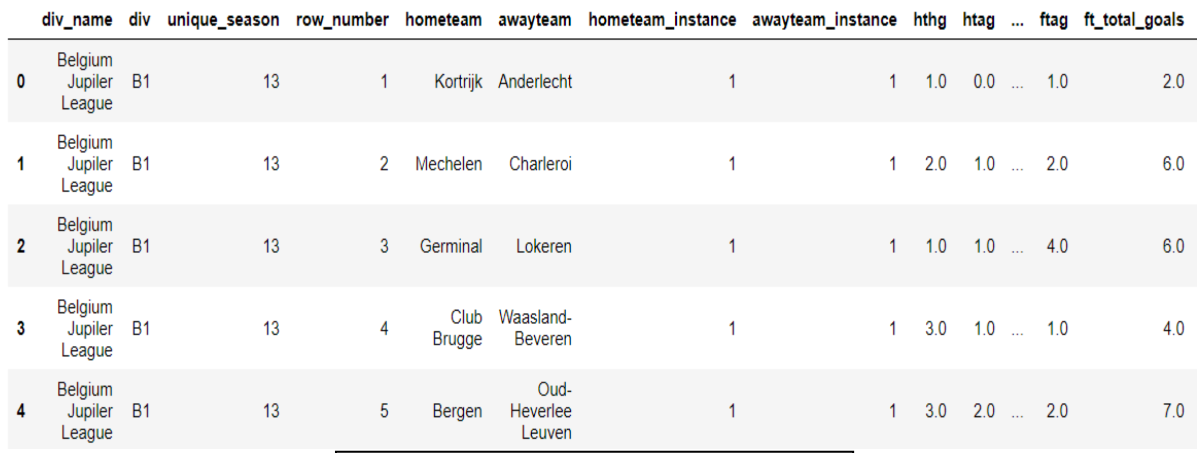
TP = True Positive, FP = False Positive

Analysis:

Data Exploration

The football data set being looked at in this project contains 39,017 rows of data that display the number of goals scored at full time. The original data set contained 39,018 rows, but one row did not have the relevant data populated.

The data contains match data from the 2012 – 2013 season up until the end of the 2016 – 2017 season. The data contains 22 leagues from across Europe. The individual names of these leagues can be seen in graphs later in this report or in the data file attached.



	div_name	div	unique_season	row_number	hometeam	awayteam	hometeam_instance	awayteam_instance	hthg	htag	...	ftag	ft_total_goals
0	Belgium Jupiler League	B1	13	1	Kortrijk	Anderlecht	1	1	1.0	0.0	...	1.0	2.0
1	Belgium Jupiler League	B1	13	2	Mechelen	Charleroi	1	1	2.0	1.0	...	2.0	6.0
2	Belgium Jupiler League	B1	13	3	Germinal	Lokeren	1	1	1.0	1.0	...	4.0	6.0
3	Belgium Jupiler League	B1	13	4	Club Brugge	Waasland-Beveren	1	1	3.0	1.0	...	1.0	4.0
4	Belgium Jupiler League	B1	13	5	Bergen	Oud-Heverlee Leuven	1	1	3.0	2.0	...	2.0	7.0

Figure 1: Input Data Before Processing

Figure 1 shows a snippet of what the initial data set looks like. There are several columns in this data set that will not be useful for prediction purposes, for example row_number & unique_season. The initial data used also contains goals for both sides during the match. Whilst this was useful to calculate the average number of goals scored the columns will not be used for predicting.

The first home game and the first away game for each team will contain missing data in the average goal's columns. These will be removed in the data processing stage.

The aim of this project is to identify matches that end with 3 or more goals in the game. Initial high-level analysis shows that 49.3% of games end with 3 or more goals in. If a spectator was to randomly pick a match then over half the time the match would finish with 2 or fewer goals in, we can do better than this.

<u>Data fields to be kept:</u>

- **div_name**: The name of the division the match is being played in (string)
- **hometeam_instance**: The number of home games the home team has played that season (integer)
- **awayteam_instance:** The number of away games the away team has played that season (integer)
- **ft_total_goals:** The number of goals that occurred in the match (integer)
- **avg_h_scored:** The average number of goals the home team has scored at home, before the instance occurs (float)
- **avg_h_conc:** The average number of goals the home team has conceded at home, before the instance occurs (float)
- **avg_a_scored:** The average number of goals the away team has scored away from home, before the instance occurs (float)
- **avg_a_conc:** The average number of goals the away team has scored away from home, before the instance occurs (float)
- **h_form_over25:** During the last 5 games that the home team played at home, how many ended with 3 or more goals (integer)
- **h_form_under25:** During the last 5 games that the home team played at home, how many ended with 2 or fewer goals (integer)
- **a_form_over25:** During the last 5 games that the away team played away from home, how many ended with 3 or more goals (integer)
- **a_form_under25:** During the last 5 games that the away team played away from home, how many ended with 2 or fewer goals (integer)

The following data fields will be used to help predict the target variable. Using the ft_total_goals column, a binary 1 & 0 field will be constructed, and this will be the target variable. If the game ended with 3 or more goals, then this flag will display as 1. If the game contained 2 or fewer goals, then this flag will display as 0.

All other fields that have not been mentioned above, such as hthg which stands for half time home goals scored, will be removed from the data set during the processing stage.

For the purposes of this report a high scoring game is classed as 3 or more goals.

Exploratory Visualisation

Whilst exploring the data and thinking about what variables should be used to drive the algorithms one interesting finding was that the percentage of high scoring games can fluctuate depending on what division the match was being played in. Figure 2 below shows this.

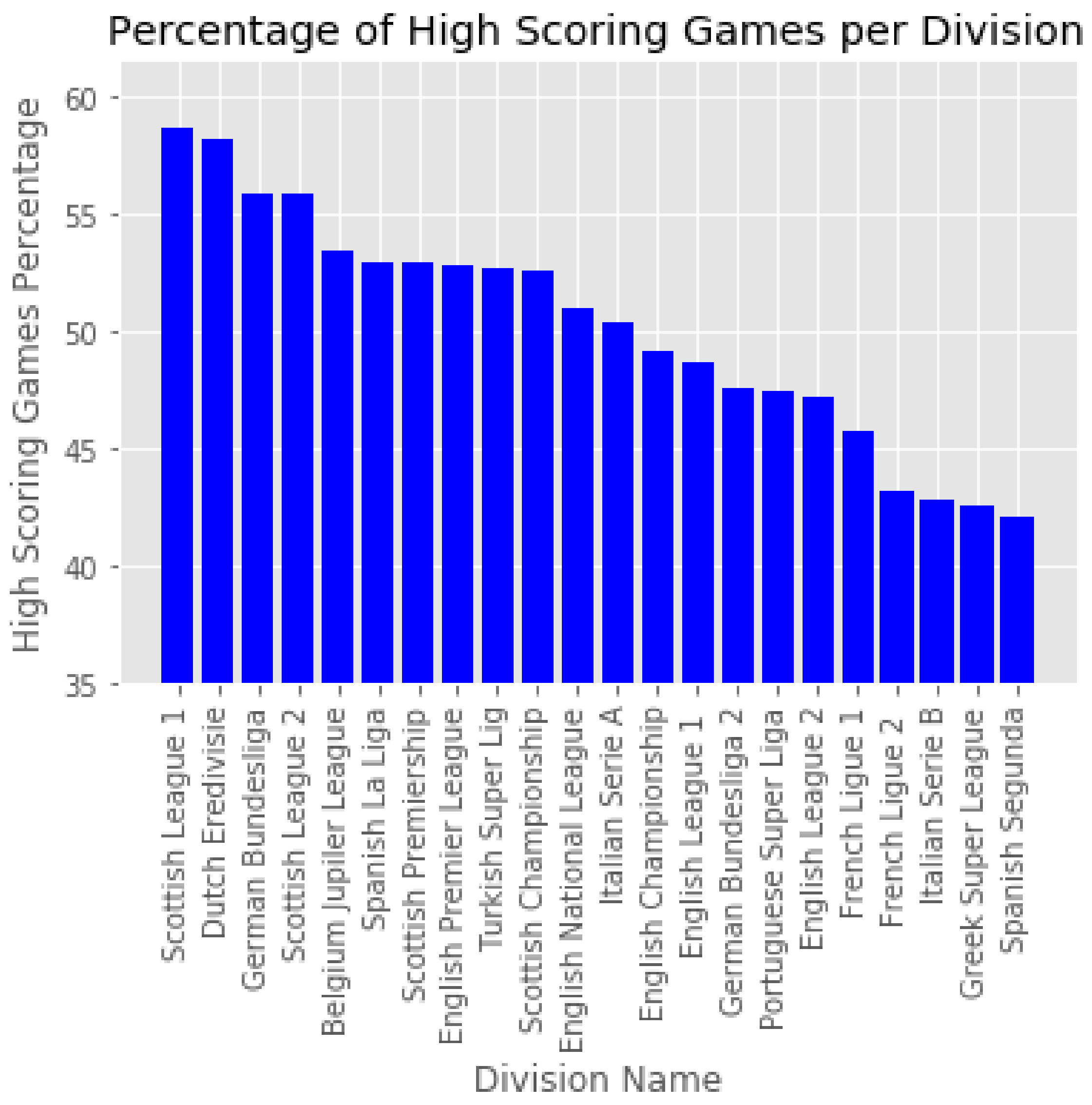


Figure 2: Percentage of High Scoring Games per Division

Figure 2 shows that the most exciting leagues to watch are “Scottish League 1” and the “Dutch Eredivisie”. The least exciting leagues to watch over the last few years are the “Greek Super League” and the “Spanish Segunda”. Certainly, avoid showing new fans of the sport games from these low scoring divisions.

Algorithms & Techniques

This report will be comparing two powerful machine learning algorithms that have yielded good results in literature before. The type of problem that is being solved is a classification problem so the two algorithms I will be using are Support Vector Machines (SVM) and Random Forests. SVM's have been used throughout literature [5] and have solved complex questions such as gene selection for cancer classification. Random forest classification has been used to help aid the diagnosis of heart problems [6].

SVM Default Parameters [7]:

- **C:** The regularisation parameter that controls the trade-off between achieving high testing scores vs achieving high training scores. Default 1.
- **Kernel:** This sets the type of kernel that will be used for the algorithm. The default is radial basis function (RBF).
- **Degree:** If using a polynomial kernel then this will be the used to set the degree of the polynomial, the default is 3.
- **Gamma:** The kernel coefficient for the rbf, and other kernels. This controls the influence of training examples. The default is 1/n_features

Random Forest Parameters [8]:

- **n_estimators:** The number of (decision) trees in the forest. Default = 10.
- **Criterion:** The function that measures the quality of each split made by the trees. Default = Gini impurity.
- **Max Depth:** The maximum depth the trees can get to. The default is none, the tree's keep going until all leaves are pure.
- **Min Samples Split:** The minimum number of samples needed to split a node. The default = 2
- **Max Features:** The number of features to consider when looking for the best split. The default is the square root of the number of features.

Benchmark

If you were to plot the number of goals scored per game in a football match the you will see it closely follows a Poisson Distribution. This can be seen in figure 3 [9]. This has meant Poisson models have been used in literature [10] to model football matches. This report will be taking the average of the 4 average goals scored & conceded fields to build a probability for each instance using the Poisson distribution. If the probability is 50% or greater for 3 or more goals, then the model benchmark model will predict the game will end with 3 or more goals.

The logic used to create the Poisson benchmark model was to average the home goals scored and away goals conceded to give an average of the amount of goals the home team would score. Then the same is applied but instead the away goals scored are average with the home goals conceded. These two values are then used to drive the Poisson distribution equation to calculate the probability of the game ended 0-0, 0-1, 1-0, 1-1, 2-0, 0-2. Subtracting the probability that any of the previous events took place, from 1 will give the probability that the game ends with 3 or more goals. The function can be seen at *poisson_dist.py*

During the data processing stage, the data set will be reduced further to only look at home & away instances greater than 5 to remove average goal outliers therefore the benchmark accuracy & precision will be applied to the same reduced data set.

Using this benchmark model on the data set the Poisson distribution had an ***accuracy of 54.04%*** and a ***precision of 53.68%.*** This is the target for the machine learning algorithms to beat.

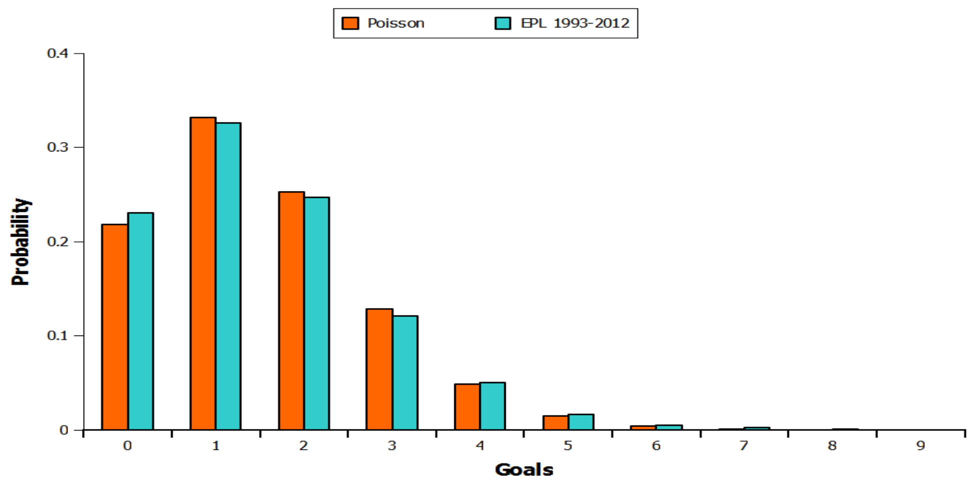


Figure 3: Premier League History Poisson Distribution

<u>Methodology:</u>

Data Pre-processing

As mentioned previously, the original data from the website was first stored and manipulated in SQL to create the average goals scored that season columns, as well as the columns containing under over 2.5 goals form in the last 5 games.

The data prep conducted in the notebook follows two stages. The first was to manipulate the data to report out the accuracy & precision of the benchmark model. The second stage was to prepare the data for training the machine learning algorithms.

Stage One:

1. Altered a division name for aesthetic reasons
2. Match instance 1 removed & rows with null data removed
3. Irrelevant fields removed & target variable added
4. Match instance 1 to 5 removed - This helped to remove outliers
5. Poisson probability function applied - Accuracy & Precision results obtained

Stage Two:

1. Remove all irrelevant columns generated in stage one
2. Identify the target variable & normalise the numerical values
3. One-hot encode the categoric division variable to separate fields
4. Split the data into training and testing sets - 80:20 split

The above stages can be found in the “Data cleansing & analysis” and “Data Preparation” stages of the Python notebook attached.

Implementation

During the implementation stage the project looks at implementing two machine learning algorithms to the shuffled and split data sets. The two algorithms, as mentioned before are Random Forests & Support Vector Machines.

Random Forest:

- Set up the Random forest algorithm with a fixed random state of 95 and all other variables default
- This was easily fitted to the training data to learn and predictions could be made on the testing data set
- The code was easy to implement (as seen below) and the random forest achieved an ***accuracy score of 51.28%*** with a ***precision score of 48.86%***

```
predictions = (clf.fit(X_train, y_train)).predict(X_test)
```

Support Vector Machine:

- Set up the SVM with a fixed random state of 95 and all other variables default
- This was easily fitted to the training data to learn and predictions could be made on the testing data set
- The same code was used and the SVM achieved an ***accuracy score of 53.08%*** with a ***precision score of 51.01%***

No additional complications were raised during the implementation and the original metrics were adequate for the solution acquired. The code can be seen under the "Implement the algorithms" section of the Python notebook attached.

The two initial models that were implemented are achieving slightly worse scores than the Poisson model that is being used for the benchmark. In the next step the project will look to refine & tune the models to achieve better results.

Refinement

The hyperparameters of the model can now be tuned to try and achieve better results. The grid search method will now be applied to find the best combination of parameters for achieving higher results. The f-beta score used to judge the models during grid search. The beta value is set to 0.4 to favour choosing the models with a higher precision.

Random Forest:

The random forest used grid search to test a combination of changing 5 of the hyperparameters. Hyper parameters being tested are shown below.

- n_estimators: [25, 50, 75, 100]
- criterion: ['gini', 'entropy']
- max_depth: [16, 32, 64, 128]
- max_features: ['sqrt', 'log2']
- max_leaf_nodes: [16, 32, 64, 128]

Due to the quick training time offered by the random forest the grid search was applied to a large combination of variables without having to have huge waiting times.

The optimised model achieved an **accuracy of 53.31%** and a **precision of 51.34%.** This is an improvement of around 2% over the initial default model.

Support Vector Machine:

The random forest used grid search to test a combination of changing 5 of the hyperparameters. Hyper parameters being tested are shown below.

- C: [0.1, 1, 10, 100]
- kernel: ['rbf', 'sigmoid']

Due to much longer training times of the SVM, fewer combinations of hyperparameters were used for tuning.

The optimised model achieved an **accuracy of 53.85%** and a **precision of 51.90%.** This is an improvement of around 2% over the initial default model.

Results:

Model Evaluation and Validation

The final model I will using is the optimised Random forest. I have chosen this over the SVM because the accuracy and prevision are very similar and even though the SVM has yielded slightly better results the long training times is making it less practical to tune further and use in a rapid pipeline environment.

Random Forest Hyperparameters:

Apart from the hyperparameters listed below, the rest were set to default.

- n_estimators = 75
- criterion = 'gini'
- max_depth = 16
- max_features = 'sqrt'
- max_leaf_nodes = 16

These final parameters are showing that the best sized forest was one containing 75 different decision tree's in. The maximum depth that any one of these could get down to 16 splits down with the max number of nodes on each leaf also being 16.

In literature it is often seen that Random forests are robust [11] which is another reason I decided to choose the Forest over the SVM. To test the robustness of the model created the project will re-train and re-test the model on a data set containing only the top 5 watched leagues in Europe.
(Italian Serie A, English Premier League, German Bundesliga, Spanish La Liga & French Ligue 1)

The new data set was formed & split into testing and training sets of the same proportions as before. The ***accuracy score was 50.99%*** and the ***precision score was 52.23%***

This result shows that the model used is robust and delivers a slightly lower accuracy but a better precision score with the modified input data set than the original, bigger one use in this project. All the code and results can be found in the "Testing Robustness & Final Visuals" section of the Python notebook.

Justification

There results from the Random forest generated in this report appear to be robust and therefore can be trusted, although the scores are not as high as originally hoped.

Poisson benchmark model:	Accuracy = 54.04%, Precision = 53.86%
Random forest:	Accuracy = 53.51%, Precision = 51.34%

The high-level analysis conducted at the start of this document showed that 49.3% of football matches ended with 3 or more goals. The Random forest model created has proven to be better than the naive approach of just simply predicting every single game would end with 3 or more goals.

The Random forest did not score better than the benchmark model. With the speed and simplicity of the Poisson function, if the results of this project were to be actioned and football matches were to be classified then I would stick with the benchmark model.

Machine learning techniques got close to matching the benchmark model and it is still fully possible that machine learning techniques could beat the benchmark model. This will be discussed further in the improvements section of this document.

Conclusion:

Free-Form Visualisation

Although the benchmark model has not been beaten, yet, useful information can still be taken from the Random forest that was built. Figure 4, next page, shows the importance of each of the features to the Random forest making its classification decisions. *(Taken from the robust forest training data)*

When exploring this field deeper in the future it could be worth trying to find data that can help compliment the avg_h_scored number, for example the cost of the strikers in the home team or the number of goals the strikers starting that game have scored that season. Understanding the important features and what the root cause of generating these features are could one day be the key to solving the problem discussed in this project.

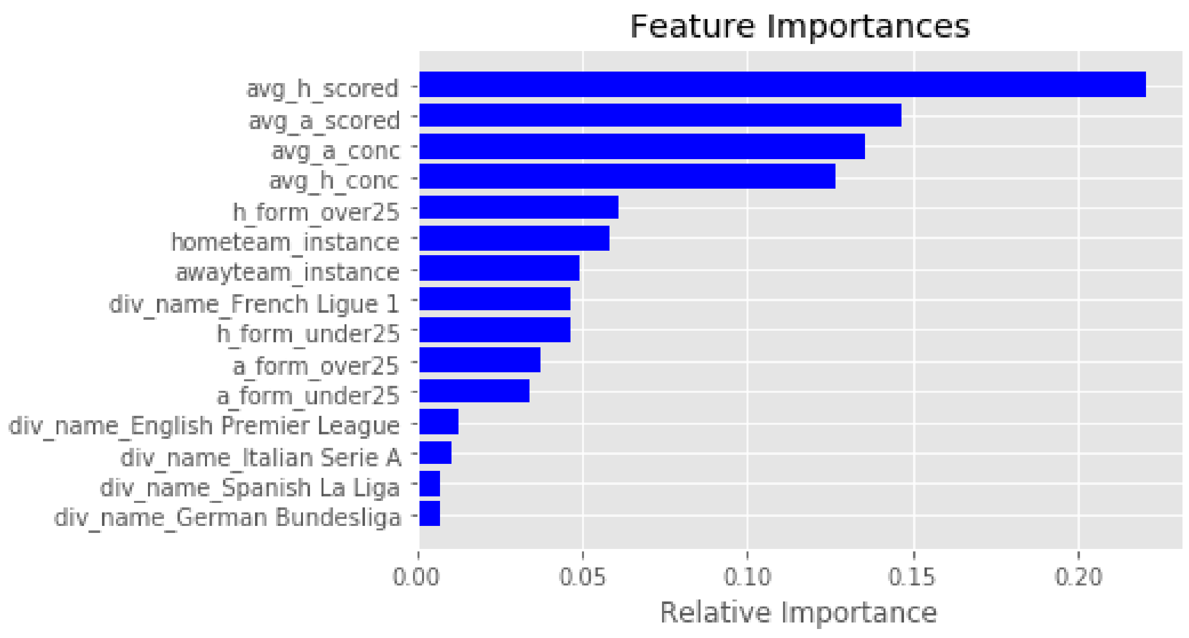


Figure 4: Feature Importance of Robustness Test Forest

Reflection

The project started with initial high-level analysis of the data. It was interesting to see how the number of high scoring games changes per division. Next a benchmark model was formed, this project built a strong Poisson model which also closely follows the distribution of goals in a football match. The data then underwent more preparation and transformations ready for inputting into the machine learning algorithms. Two algorithms were trained but in the end this project favoured the Random forest. The model was optimised and tested for robustness before being compared to the benchmark model. The Random forest got close but could not score better than the Poisson model.

One especially interesting discovering of this project was the feature importance, being able to communicate this type of feature information could be powerful in a business environment and really help people understand what is driving the model. The biggest difficulty encountered in this project was trying to beat such a strong benchmark model but one thing I have learnt throughout my machine learning journey is you have to be brave with machine learning because that’s how you make the ground-breaking discoveries.

Improvements

Without generating more training data or adding more features to the data the biggest change to improve results would be to implement different types of classification algorithms.

It is shown in literature that neural networks and deep learning are often state of the art algorithms for many problems. Neural networks are a start of the art classifier [12] so it is recommended that the work conducted in this project is expanded further but with a heavy focus on designing strong Neural network architecture to build a truly power classifier that can do what the Random forest could not and beat the Poisson model.

The Poisson model of combining the averages of the goals scored & conceded by the home & away teams can be beaten and the work done in this project is only the beginning of a journey to solving this problem.

References:

1. https://www.theguardian.com/football/2016/oct/24/sky-sports-bt-sport-people-switching-football-off - [Accessed 23/03/2019]
2. http://www.football-data.co.uk/data.php - [Accessed 23/03/2019]
3. https://developers.google.com/machine-learning/crash-course/classification/accuracy - [Accessed 23/03/2019]
4. https://developers.google.com/machine-learning/crash-course/classification/precision-and-recall - [Accessed 23/03/2019]
5. Guyon, I., Weston, J., Barnhill, S. and Vapnik, V., 2002. Gene selection for cancer classification using support vector machines. *Machine learning*, *46*(1-3), pp.389-422.
6. Lempitsky, V., Verhoek, M., Noble, J.A. and Blake, A., 2009, June. Random forest classification for automatic delineation of myocardium in real-time 3D echocardiography. In *International Conference on Functional Imaging and Modeling of the Heart*(pp. 447-456). Springer, Berlin, Heidelberg.
7. https://www.hackerearth.com/blog/machine-learning/simple-tutorial-svm-parameter-tuning-python-r/ - [Accessed 24/03/2019]
8. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html - [Accessed 24/03/2019]
9. http://pena.lt/y/2012/10/29/using-poisson-to-predict-football-matches/ - [Accessed 25/03/2019]
10. Koning, R.H., Koolhaas, M., Renes, G. and Ridder, G., 2003. A simulation model for football championships. *European Journal of Operational Research*, *148*(2), pp.268-276.
11. Cootes, T.F., Ionita, M.C., Lindner, C. and Sauer, P., 2012, October. Robust and accurate shape model fitting using random forest regression voting. In *European Conference on Computer Vision* (pp. 278-291). Springer, Berlin, Heidelberg.
12. Baesens, B., Van Gestel, T., Viaene, S., Stepanova, M., Suykens, J. and Vanthienen, J., 2003. Benchmarking state-of-the-art classification algorithms for credit scoring. *Journal of the operational research society*, *54*(6), pp.627-635.