Predicting the Outcome of Football Matches with Supervised Learning

João Paulo Abelha
Porto, Portugal
up201706412@fe.up.pt

João Rafael Varela
Porto, Portugal
up201706072@fe.up.pt

Vítor Hugo Barbosa
Porto, Portugal
up201703591@fe.up.pt

***Abstract*—This article presents the details about our approach to predict results of football games using data from the *FIFA* game. After a brief introduction this document provides a description of the problem and analysis of the used data set. Later, it explains our approach to the stated problem and the thinking process behind it, as well as various performance comparisons between different machine learning algorithms and between different subsets of data used to train the classifiers.**

***Index Terms*—machine learning, supervised learning, deep learning, feature selection, decision trees, random forest, k-nearest neighbor, neural network**

I. Introduction

Machine learning is increasingly becoming a part of our daily lives and has shown promising results in the domains of classification and prediction. One of the expanding areas necessitating good predictive accuracy is sport prediction, due to the large monetary amounts involved in betting. In addition, club managers and owners are striving for classification models so that they can understand and formulate strategies needed to win matches. These models are based on numerous factors involved in the games, such as the results of historical matches, player performance indicators, and opposition information.

One of the most important tasks of machine learning consists in predicting a label for previously unseen data (classification). This classification is usually done by a predictive model that uses training and test data to increase its accuracy in correctly predicting the results for a given problem. This is known as supervised learning, a branch of machine learning.

Predicting the outcome of a football game is itself a classification problem, as we can predict a game to result in a win, loss or draw.

II. Problem Definition

A. Goals

The main goal of the work developed was to build different classifiers to accurately predict the results of football games, maximizing their performances and comparing them between the different classifiers.

Performance isn't everything, so we also considered relevant comparing the classifiers' performance using different sets of features from the dataset. Using less data can sometimes lead to better results and will result in faster training times.

B. The Dataset

The original dataset [1] contains over 25 thousand european football matches as well as some statistics from the players playing on it (sourced from EA Sports' FIFA video game series). This games all happened between the 2008 and 2016 seasons, and were played in 11 different regions (on their respective main championship). For each match, the dataset also contains the teams playing on it, the result and odds sourced from 10 different major betting websites.

III. Approach

Let alone the different algorithms and techniques used to reach our goals, we followed a very traditional supervised learning pipeline.

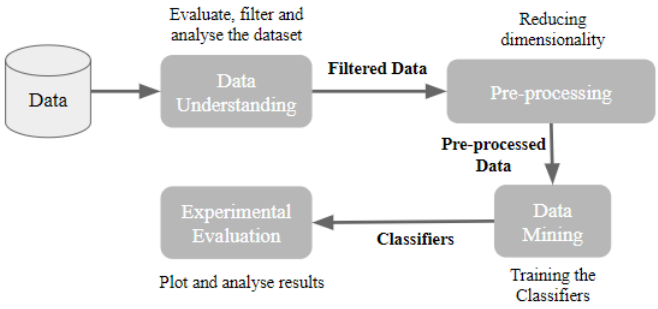

Fig. 1. Supervised Learning Pipeline

A. Data Understanding

To better understand the data we were working with we started with an exploratory analysis by measuring a set of different statistics and converting the data to a more user friendly format, such as plots or graphics.

Exploring the data is a very important step since it allows us to detect important details on the data such as patterns, tendencies or relationships between different features, making it easier to later reduce the problem's dimensionality (since some features might be ambiguous). It also permits us to determine the data quality by finding any outliers, noise input, missing values or even duplicate data (which must be treated accordingly).

In order to understand if our classifiers will be able to predict game results for different championships, it is important

that the data set contains a decent amount of games for each of the 10 different leagues, which we've confirmed:

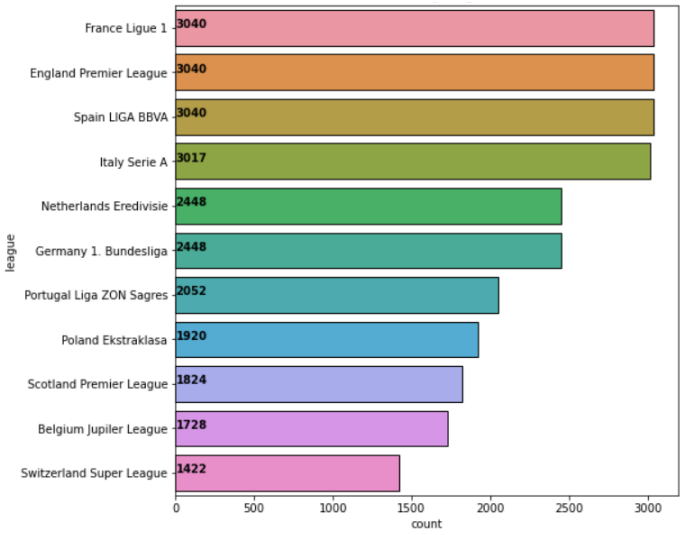

Fig. 2. Number of Matches By League

When analyzing the different tables from the data set we haven't found many problems, except for the fact that some players had all of their attributes missing. We tackle this problem later by calculating the team's average to substitute the null attribute.

After making an in-depth analysis to the different player's attributes, we have come to the conclusion that most of them do not have a particular strong relationship with a player's overall rating, but instead this overall depends of the players features grouped together.

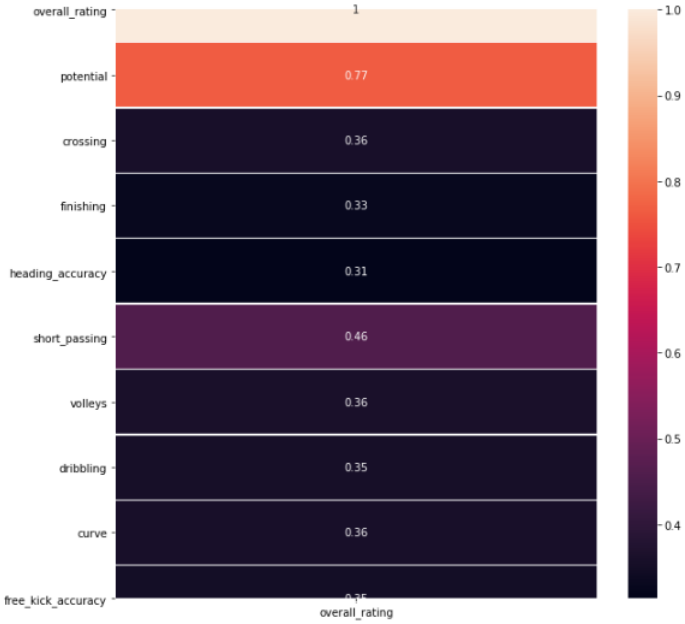

Fig. 3. Weight of Attributes on Player's Overall Rating

After filtering the data, we ended up with a data set containing information about 21374, from the original 25 thousand, which we still consider a fair amount to be worked with.

B. Pre-processing

Data preprocessing is an integral step in Machine Learning as the quality of data and the useful information that can be derived from it directly affects the ability of our model to learn. Therefore, it is extremely important that we preprocess our data before feeding it into our model.

Typical components of this step include aggregating data, reducing data set size, reducing dimensionality (number of features), data normalization and so on. Applying this techniques can bring important advantages such as reducing the training time, reducing the chances of overfitting (when the classifier function is too close to the data set training), and even leading to a higher predictive accuracy in some cases.

1) Data Aggregation: When aggregating the data, which wasn't that easy of a job due to how spread the data we wanted was across different tables, we opted to include, for each match, the overall of each player instead of some of their other attributes. We did this not only because our data set would have too many attributes, but also because the data analysis showed us that those attributes were somewhat ambiguous and were only useful when grouped together, which they are in the overall rating calculated by FIFA.

Besides, we also included the average odds for each possible game result (win,draw and loss) since the odds from the different websites never differentiated too much from each other.

We also thought it would be interesting to include some statistics from the previous games, since momentum is such an important thing in football, and from the games played between the same two teams. Therefore we aggregated, for each team on the match and from the previous 10 games, the difference in goals and the number of wins, as well as the number of games won against each other.

2) Dimensionality Reduction: At this point, our data set holds exactly 31 different attributes which might seem a lot, but not for a complex game like football where an enormous amount of variables can influence the result of a game. Nevertheless, we applied 3 different techniques so we could try reducing the number of features.

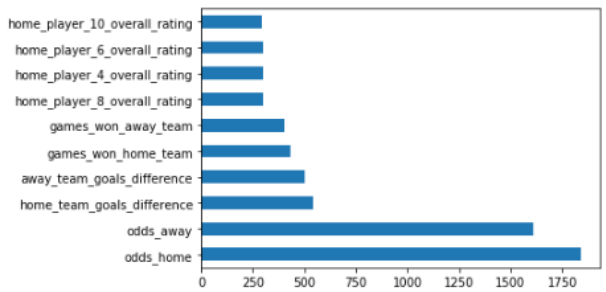

Fig. 4. Univariate Selection (top 10 features)

We started by the **Univariate Selection** technique which uses statistical tests that show how strong is the relationship between a feature and the output variable. As expected, the results shows us that the odds of the game team winning (odds_home) and losing (odds_away) have indeed an important role on predicting the result. The same can't be said for the drawing odds, probably due to it's unpredictability. The statistics we added about the latest previous games are somewhat important too, but much less than the mentioned odds.

The second technique works similarly from the first one and measures the relevance of a feature towards the output variable. That is **Feature Weighting**

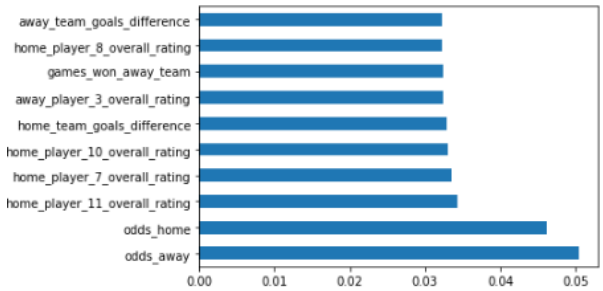

Fig. 5. Feature Weighting (top 10 features)

This time, the odds of winning or losing show up relevant again. The same can not be said for the other features which all fall in the same category.

At last, we applied one of the most powerful and most used techniques when it comes to dimensionality reduction, which is the **Principal Component Analysis** (PCA). In a nutshell, PCA makes use of the data covariance and it's eigenvalue decomposition, calculating the most relevant features on the data set.

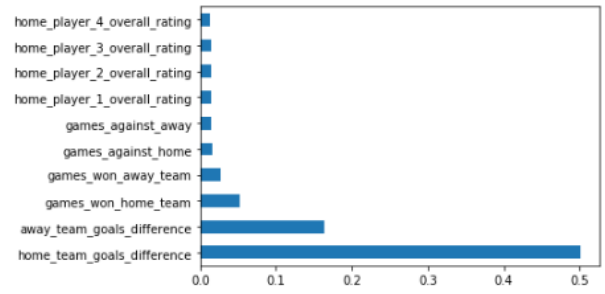

Fig. 6. PCA (top 10 features)

Surprisingly, this time around the odds were not chosen as a relevant feature and the statistics from the latest previous games came on top.

By analyzing the results of the 3 algorithms applied, we concluded that the odds and statistics from the latest games should be kept on our data set. The rest of the features consist on the overall ratings of the twenty-two players on the field which have not shown a particular relevance from the result of the game. This does not mean that the players' skills have no impact on the game's result, but in this case it might mean that those ratings are only relevant when grouped together (as a team). We confirmed this idea by applying the 3 dimensionaly reduction techniques again, but this time grouping the overall ratings for each team. Here is how the Feature Weighting results came out to be this time:

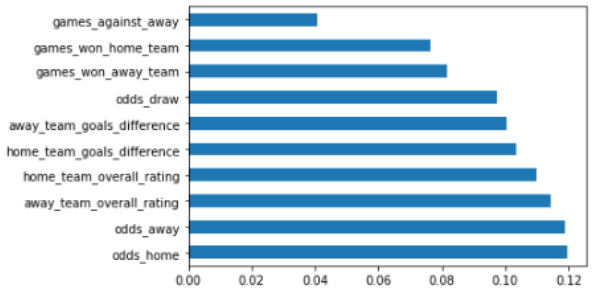

Fig. 7. Feature Weighting Selection (top 10 features)

We can see that when grouped together (average) the players' overall ratings are indeed important for the game's result.

Not knowing for sure that aggregating these players' ratings would result in better accuracy from the classifiers, we decided to kept going with both data sets: one with twenty-two players overall ratings (which we will call normal data set) and other with the average of those ratings for each team ("minified" data set).

3) Data Normalization: The goal of normalization is to change the values of numeric columns in the data set to a common scale, without distorting differences in the ranges of values. This is a very importance step since variables that are measured at different scales do not contribute equally to the analysis and might end up creating a bias. This happens all across our data set, for instance the overall ratings range from around 50 to 100 whereas the number of wins in the last 10 games will always range from 0 to 10. After calling the following function on our data frames, all the values were nicely spread between 0 and 1.

```
def normalize(df):
    result = df.copy()
    for feature_name in df.columns:
        max_value = df[feature_name].max()
        min_value = df[feature_name].min()
        result[feature_name] =
        (df[feature_name] - min_value) /
            (max_value - min_value)
    return result
```

C. Data Mining

After having our pre-processed data, we are now ready to start building our classifiers. Classifiers (or models) are functions that will return an output for a given input and they can be used in some different ways.

In a supervised learning model, the algorithm learns on a labeled data set (game result in our case), providing an answer key that the algorithm can use to evaluate its accuracy on training data.

We wanted to try out some different supervised learning algorithms so we could compare results between them and know which is the best to solve this kind of problem. Hence, we tried 5 different algorithms: **Decision Trees**, **Random Forest**, **K-Nearest Neighbor**, **Support Vector Machines** and **Neural Networks**. For each, we invested a lot of resources in testing and tuning parameters: we've done this by using **Grid Search** which is the process of scanning the data to configure optimal parameters for a given model.

Before feeding the data to our classifiers there is a crucial step that aims to avoid overfitting and allows us to test how well our model is able to get trained by some data and then predict data it hasn't seen yet. In other words, it allows us to generalize our results. That step can be done using the **K-cross fold validation** technique.

The algorithm splits the original data set into k subsets, use one of the subsets as the testing set, and the rest of the subsets are used as the training set. This process is then repeated k times such that each subset is used as the testing set exactly once. Considering the size of our data set, we chose a k value of 10.

It's important to notice that we are working with an unbalanced data set: not all classes have the same number of elements (e.g. there are more wins than losses). This could represent a problem making our classifiers biased. Therefore, we ended up using the **Stratified K-cross fold validation** technique, which does the same thing that the original K-cross fold validation does, while maintaining the class proportions the same across all of the folds, which is vital for maintaining a representative subset of our data set.

IV. Experimental Evaluation

The job is not done right after building the classifiers. One of the most important steps of this pipeline is testing and analyzing the results so we can reach well justified conclusions. We started by applying the Decision Tree algorithm:

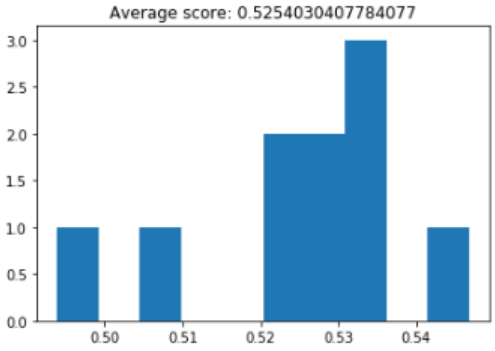

Fig. 8. Decision Tree Accuracy

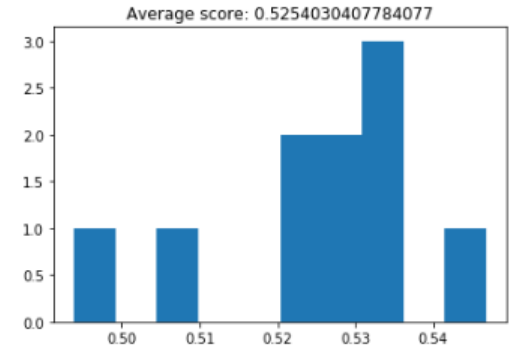

Fig. 9. Decision Tree Accuracy ("minified" data set)

We were able to obtain an average accuracy of 52%, with a training time of 31 seconds (includes the paremeter tunning with Grid Search). When using the "minified" data set we achieved exactly the same accuracy. This was replicated across all algorithms (the results obtained from both data sets were always very alike), so from now on, only the "minified" data set results will be stated. This means we could reduce the data set from 31 attributes to just 11 while maintaining the performance.

With the Random Forest we obtained similar results, and because this algorithm uses a set of Decision Trees this means that there wasn't much room for improvement with this data.

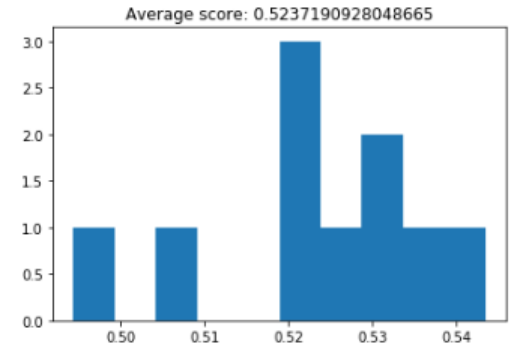

Fig. 10. Random Forest Accuracy

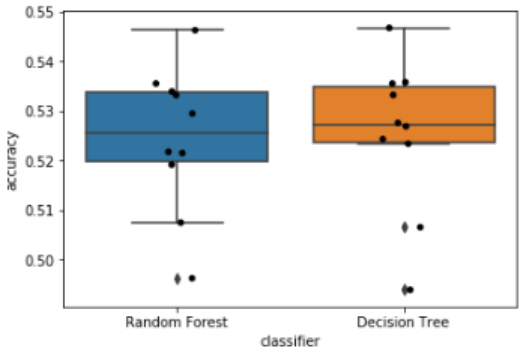

Fig. 11. Random Forest vs Decision Tree

This goes to show that a more complex algorithm does not automatically ensure better results. In this case, the simplest one (Decision Tree) achieved the better accuracy even when the Random Forest took 879 seconds to train. For the K-Nearest Neighbors and Support Vector Machine algorithms, the results were somewhat disappointing:

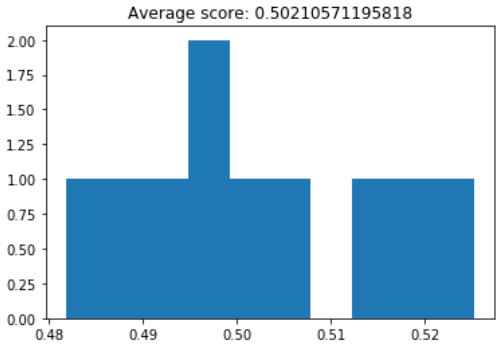

Fig. 12. K-Nearest Neighbors Accuracy

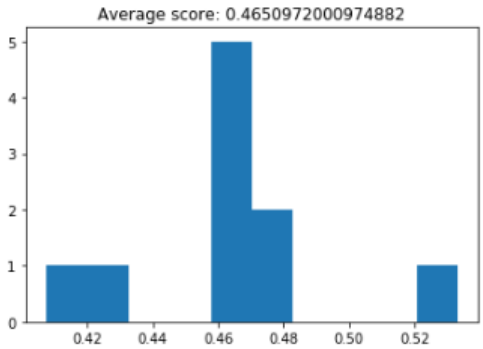

Fig. 13. Support Vector Machine Accuracy

We could not let out the usage of one of the most powerful tools in the deep learning field: neural networks. We've tried some different architectures for those networks and came across the best results our own simple custom model with 7 hidden layers.

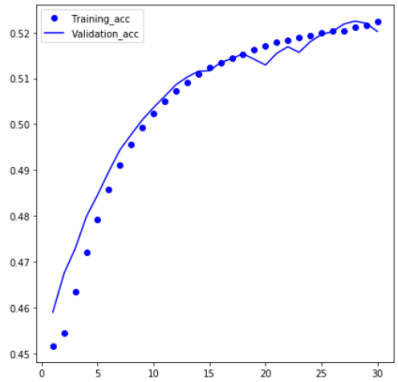

Fig. 14. Neural Network Accuracy across the Epochs

The model was able to reach it's highest performance at around the 30th epoch, starting to overfit after that. The training took only about 25 seconds. Once more, we were not able to achieve better results than with the decision tree, reaching an accuracy of 52%.

We decided to plot a confusion matrix in order to understand the classifier's behaviour.

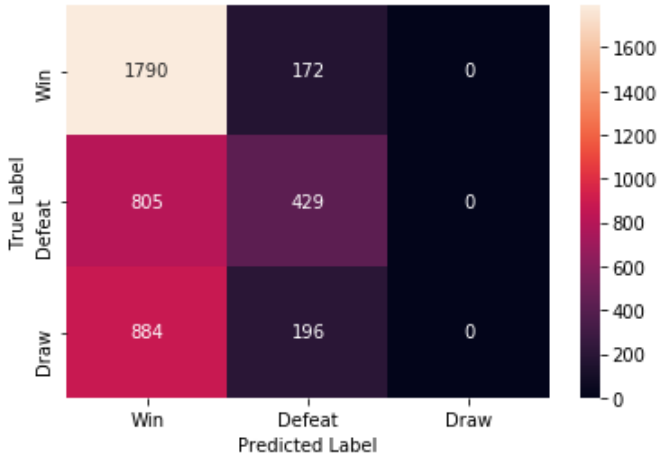

Fig. 15. Neural Network Confusion Matrix

The model clearly has problems to predict draws, opting to never even predict it. This happens due to the natural unpredictability of a draw happening. Even for teams that have somewhat close ratings, it's more likely that someone will win over the other than a draw. This difficulty is present in every classifier we have tried.

Here's how the classifiers compare themselves in terms of metrics:

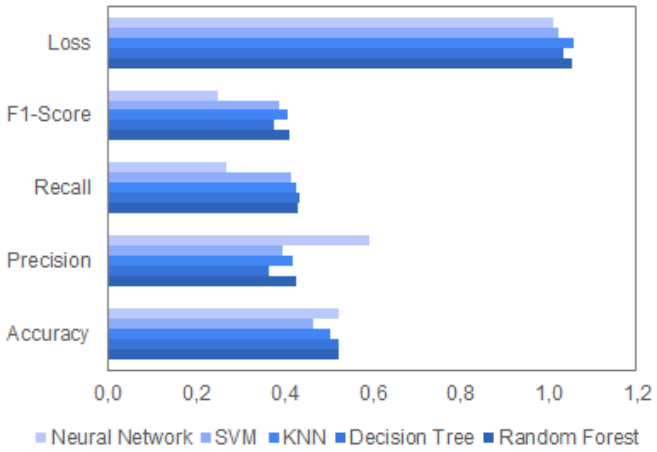

Fig. 16. Classifiers Comparison

To conclude the experiments, we though it would be appropriate to compare our results with the bookkeeper odds. After all, it would be nice to predict games even better than those odds.

The odds could predict correctly 51% of the matches, very similar to our best results. However, they face the same flaw of not being very good at predicting draws.

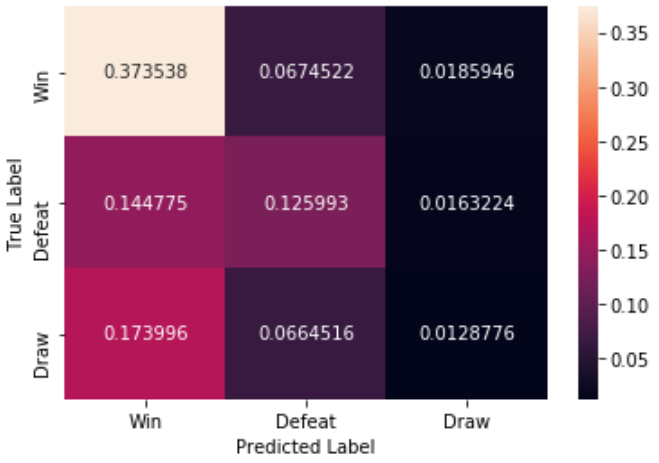

Fig. 17. Odds Prediction Confusion Matrix

V. Conclusions

In summary, the results of the project reflect a solid effort in predicting match outcomes in football using machine learning techniques. By comparing multiple classification algorithms in combination with dimensionality reduction, it was discovered a simple algorithm such as the Decision Tree provided the best results. Combining the model selection and tuning process with cross-validation, the robustness and replicability of results is ensured.

We believe that our approach can easily be applied to all sorts of classification problems, since we followed, very carefully, a pipeline that aims to tackle most of the problems when it comes to supervised learning classification.

The data from the original data set was in a very raw state, making it hard for us to chose what aspects of a football match to take into account for prediction. Nevertheless, we achieved an accuracy above 50% which is relatively high, considering the amount of factors that affect the outcome of football games in real life.

Through the development of this project, we've also come to understand even better how to utilize various machine learning tools provided by Python libraries and how important it is to study those tools so we can apply them for the right situations, in the right way (e.g. changing a simple parameter can result in much different performances).

VI. Improvements

Even though we achieved satisfactory results there is still a lot of room for improvement. For starters some other features could be created which might have resulted in better performances (e.g. goals difference on the current season).

Including other data sources besides the game FIFA would be interesting too, not only because some of the game's statistics can be considered biased but also because there are other important aspects that influence a game's outcome, such as the coaches, the current classification, etc. We believe we could even predict draws more accurately this way since there are some situations where teams are more likely to draw: when both have similar quality, both teams are low scoring or both are happy with a draw.

Investigating some more different supervised learning algorithms and techniques could also lead do better results.

References

[1] www.kaggle.com/hugomathien/soccer, "European Soccer Database," Hugo Mathien, October 2016.