Machine Learning Engineer Nanodegree

Capstone Report

“Win/Lose prediction in Soccer games based on Team Line-up”

Maged Makramalla, 2019

Contents

Definition

Project Overview

Sports Betting Industry rely heavily on developing algorithms that can help predict win/lose percentage based on roster selection and formation, rather than team general expected results.

Machine learning is heavily being used for establishing betting odds and win-lose percentages. This paper actually introduces a similar approach to the one suggested, while focusing on Tottenham Hotspur Football club data in the early 2000s.

This paper also provides a solid starting point for establishing a proper framework using Neural Networks.

There is also a couple of interesting blog posts and discussion boards such as Link1, Link2 and Link3, that provide relevant and up-to-date information of the different approaches used in the current betting industry.

However, as a fan, one could easily predict an outcome of a specific game based on team selection and injuries and general profiles of certain players.

It is therefore important to add a team's lineup into the calculation or at least show that one could predict an outcome with a high level of certainty based on the chosen lineup alone before adding the historical and ranking data to establish a more accurate estimation overall.

Problem Statement

Win/Lose Percentage is usually based on recent team performance and historical encounters between teams. The objective of this project is to develop an algorithm that tis more focused on team selection and lineup in matches.

In brief, a series of classification algorithms will such as Decision Tree, Random Forests and SVM will be used and rated based on their accuracy.

The process will first identify the players of each team that has usually the greatest impact on the result, and accordingly will predict the W/L outcome based on the team lineup.

Accordingly, when providing the team's lineup of player as an input, one should receive a win/lose classification as an output.

Metrics

As most sports fans already know, predicting the outcome of any game with a high certainty is almost an impossible task. There will always be surprises and upsets in all types of sports and competitors.

However, to develop a model that can provide a decent amount of precision, the data set will be divided into training and testing set as usual. Since the dataset is considered labeled and the results are already available, an accuracy score metrics will be calculated for all the different algorithms for both the training and testing set,

The aim is to tune the different algorithms to obtain an accuracy of over 70% for both the training and testing prediction to avoid over and underfitting.

Analysis

Data Exploration

This project will use the European Soccer Database available on Kaggle.

This data set should have all the information required for the project.

It consists of 7 different data sets:

1. Country: 11 x 2 (rows x column)
2. League: 11 x 3 (rows x column)
3. Match: 26k x 115 (rows x column)
4. Player: 11.1k x 7 (rows x column)
5. Player_Attribute: 184k x 42 (rows x column)
6. Team: 299 x 5 (rows x column)
7. Team_Attribute: 1458 x 25 (rows x column)

The data includes:

- Data gathered from 11 different European countries in the seasons between 2008-2016
- Player's and Team's attributes sourced from EA Sports data
- Team line up and formations of each game
- Betting odds from 10 different providers
- Detailed match statistics:
    - Stage
    - Goals
    - Players
    - Possession

The target value will rely heavily on the Match dataset since it includes most of the information needed, along with the Team and Player dataset to support the prediction.

In order to use this dataset, a couple of conversions and additions haver to be performed first.

First of all the sqlite file has to be converted to a csv file to make it easier to inspect and to use using the common libraries such as pandas.

It is also worth mentioning that there are no win/lose variable indicated in any of the datasets, so this crucial variable will have to be obtained using the provided data such as home and away goals.

Additionally, it will be a bit challenging to differentiate between the home and away teams along with all of their stats, which will be more elaborated in the following sections.

Lastly, there are also some cells with missing data, which will be properly dealt with based on the application and necessary calculations.

Exploratory Visualization

This section will focus on how each dataset will be used more thoroughly.

3 data sets will mainly be used:

- Player
- Team
- Match

Player:

The player dataset contains 7 columns as per the below snippet:

id	player_api_id	player_name	player_fifa_api_id	birthday	height	weight
1	505942	Aaron Appindangoye	218353	2/29/1992 0:00	182.88	187
2	155782	Aaron Cresswell	189615	12/15/1989 0:00	170.18	146
3	162549	Aaron Doran	186170	5/13/1991 0:00	170.18	163
4	30572	Aaron Galindo	140161	5/8/1982 0:00	182.88	198
5	23780	Aaron Hughes	17725	11/8/1979 0:00	182.88	154

In this case on the player_api_id and the player_name could be used to effectively determine the most valuable players in any given team

Team:

The team data set consist of only 5 columns as per the below snippet.

id	team_api_id	team_fifa_api_id	team_long_name	team_short_name
1	9987	673	KRC Genk	GEN
2	9993	675	Beerschot AC	BAC
3	10000	15005	SV Zulte-Waregem	ZUL
4	9994	2007	Sporting Lokeren	LOK
5	9984	1750	KSV Cercle Brugge	CEB

In this case the team_api_id and the team_long_name (or team_short_name) will be used to choose the team that should be analyzed.

Match:

This dataset contains most of information required for the analysis. It contains a total of 115 columns, so a snippet won't be that helpful for visualizing purposes.

Instead the main feature that will be highlighted using the backward design or analysis approach, by exploring the desired inputs and outputs of the algorithm

Algorithm Input:

The input of the algorithm should be a list of 11 players that are participating in the game. Although this information is already available, one must consider if the team in question is considered the home or away team in the game. More technical details will be available in the Data Processing section.

The features used are therefore: home_team_api, away_team_api, home_player_1 – home_player_11, away_player_1 – away_player_11; in order to create: team_player_1 – team_playe_r11

Algorithm Output:

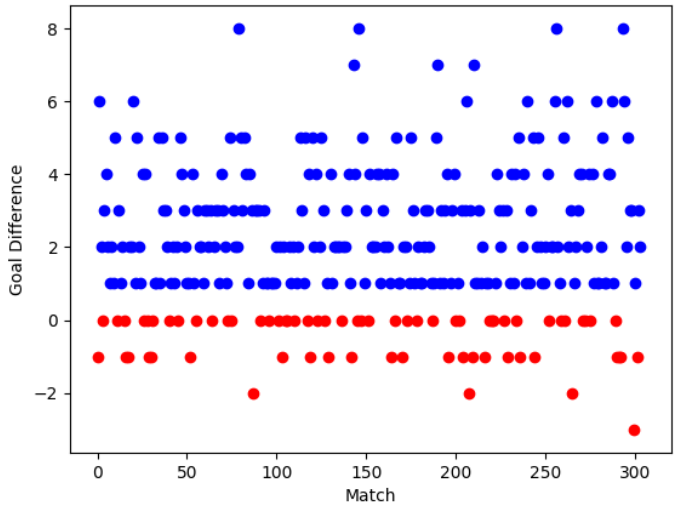
The output of this algorithm should be a Win / Lose classification indicated as '1' for win and '0' for lose.

Since this feature is not available in the data set, this result will be obtained using the goal difference, i.e. goals scored – goals conceded. In this algorithm, a draw will also be indicated, but will count as a lost game. More technical details will be available in the Data Processing section.

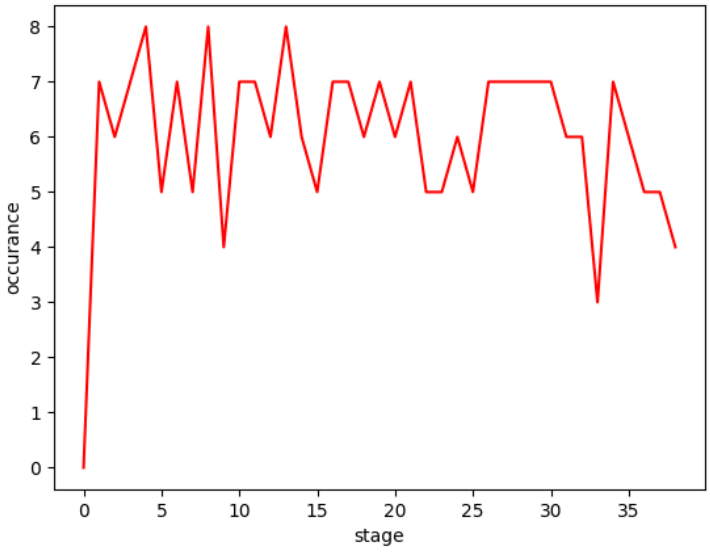
The features used are therefore: home_team_api, away_team_api, home_team_goal , home_team_goal; in order to create team_win, team_draw, team_lose (and goal difference for further regression analysis)

After developing the final dataset that includes all the required data for the development, visualizing some the interesting aspect of the data.

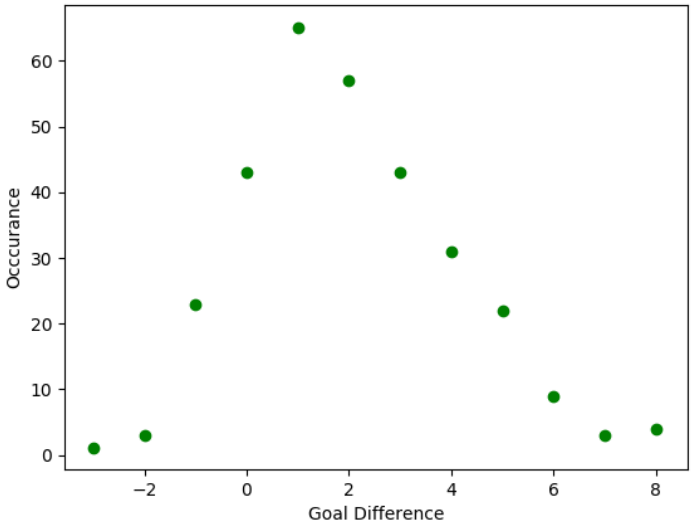
For example this first graphs just iterates through all the games and shows the goal difference of each game, while change the color to red if the game is indicated as not won and blue if the game is indicated as won. This graph servers as a proof for the data preparation step.



This second graph visualizes the amount of time a game has won in each stage of the season. It shows that for this particular team there is a noticeable decrease in games one in the later stages of the competitions, which may be explained to fatigue and injuries.



This last graph counts the amount of times this team won or lost by a specific goal difference in games the won, tied and lost. It is quite clear the usual goal difference is closer to 1, which is usually the case in most of the soccer games.



Algorithms and Techniques

In order to obtain the most effective results, several algorithms will be used throughout this process.

First of all, of all numpy and pandas will be used to process the data accordingly.

Since this is considered a classification algorithm, several classification algorithms will be used in order to obtain different results, such as Decision Trees, Random Forest and Support Vector Machines.

To elaborate more on each of these algorithms and their relevance to this application, this section will briefly discuss each of the utilized algorithms.

The first algorithm used is the Support Vector Machines Algorithm, this type of algorithms basically strives to develop a line or a hyperplane that can efficiently separate the provided data into different groups. In this case the algorithm should work on developing this line that divides outcome of a game into wins and non-wins as efficient as possible. The idea here is to develop a hyperplane and calculate the distance between this plane and data which is also called the margin. So, the overall target is to try and maximize the margin as much as possible while separating the two classification outcomes as wide as possible.

The second algorithm used is the Decision Tree Algorithm. Regardless of the final accuracy of utilizing this technique, the outcome will be the most insightful for our specific application when compared to the other algorithms, mainly due to the way it works. This algorithm works in a way that it tries to partition the data in subsets of data, as efficiently as possible using “Yes” and “No” questions, i.e. “win” or “non-win” outcome classification approach. This approach is crucial to our application, since the first “question” that will be visually placed on the top of the tree will be have the feature (or I our case the “player”) that has the greatest influence on the overall outcome of any game. In other words, this algorithm will not only predict the outcome of a game based on a lineup, it will also show which players are more crucial than other when predicting the outcome.

Lastly, the Random Forest Algorithm, which basically divides the data into randomly created different datasets and then run a decision tree algorithm on each data subset, hence the “forest” notation. To predict the output for this specific application all the trees in the forest will receive the data and “vote” what outcome they each predict and based on that the final outcome will be provided. This ensemble algorithms is considered an extremely powerful algorithm and should have the best reliable and efficient overall results.

The accuracy score metrics will be used to calculate the efficiency of each algorithm based on both the testing and training data.

Additionally, an MLP Classifier will also be used just for reference purposes.

Finally, a histogram will be used to compare all algorithms performance.

Benchmark

The project will use the available data of the betting odds available in the same data set to use as a benchmark to measure the efficiency of the developed algorithm.

Moreover, I believe a simple linear regression model dictating a 50/50 Win-Lose ratio could serve as a great baseline for this algorithm as a sanity check for the developed algorithms

Methodology

Data Processing

Before applying the machine learning algorithms the dataset has to be properly prepared.

This process is all captures in the python file called *team_games.py*.

This script starts by first selecting the team id, in this case the Spanish Side 'Barcelona' was chosen.

First the script captures all the matches that the team has played, either as a home or an away game.

Then the features indices of the players will change by substituting home_player_n to team_player_n and away_player_n to opp_player_n (n being the player number)

Five new columns are created:

- Team_goal_scored
- Team_goal_conceded
- Team_win
- Team_loss
- Team_draw

And finally, a loop will run through all the rows that does the following:

- Choosing the correct player for team and opposition based on team id
- Calculating the goal difference
- Assign the correct win – draw – loss values based on goal difference and team id

It is also worth mentioning that some of the missing data within the dataset will be properly dealt with using the next machine learning scripts, since it is usually handled based on its influence and application.

As a last step a new file labeled 'team_games' is created that includes all the necessary information for the algorithm such as: team_win, team_draw, team_lose, team_player_1 – team_player_11.

The newly created file will now be used to all further calculations and predictions.

Implementation

Before implementing the actual player lineup algorithm, a couple of other rather interesting operations have been performed to mainly explore the data and get some insights on the overall outcome.

3 different operation have been performed:

1. Game Outcome based on Possession and Goal Scored

2. Game Outcome based on stage and Goals Scored
3. Game Outcome based on Goals Scored and Goals Conceded

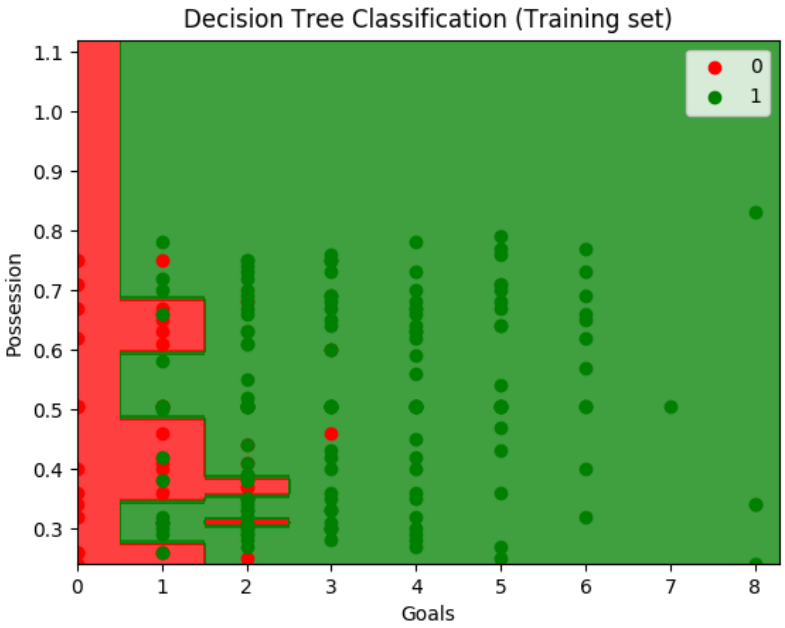
All these algorithms along with the output plots are available in the repository and in the python script labeled *dataexplore.py*.

Below is a brief discussion and plots for each operation, all utilizing a simple decision tree classifier.

Game Outcome based on Possession and Goal Scored:

Description: Predict the Outcome of the game based on the goals scored by team along with their possession through the game

Output Plot:

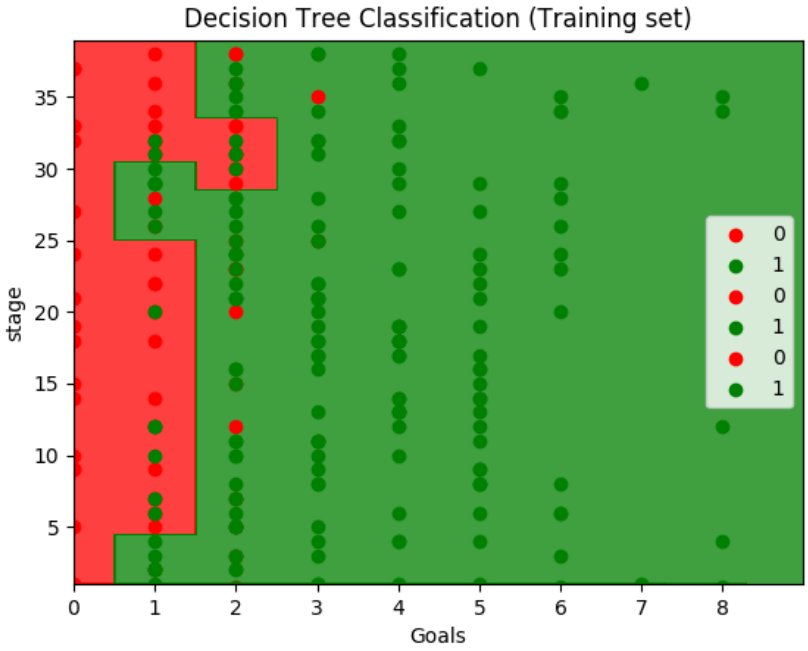


Discussion: The first observation states that the team usually needs 2 or more goals in order to increase in chances to win. Possession doesn't seem to play a big role in the overall outcome however it is worth noticing that a possession rate between 50-60% is more effective in most cases.

Game Outcome based on stage and Goals Scored:

Description: Predict the Outcome of the game based on the goals scored by team along with the stage the game is played in the season.

Output Plot:



Discussion: The first observation states that the team usually needs 2 or more goals in order to increase in chances to win as well. It is however worth mentioning that the later in season, more than 2 goals is required to win, when compared to earlier in the season when 2 goals is generally enough.

Game Outcome based on Goals Scored and Goals Conceded:

Description: Predict the Outcome of the game based on the goals scored by team along with the goals conceded by the team. Although this may seem counter intuitive in our minds, it worth exploring whether the algorithm can capture this pattern and really 'learn' what it means to win a game; scoring more goals than the opponent.

Output Plot:

Discussion: This step-like shape can clearly show that the algorithm was able to identify that scoring more goals than the opponent will guarantee winning the game.

However, it is worth adding that since the algorithm has no knowledge about the outcome whenever an opponent scored more than 4 goals, i.e. a final score of 5-8 for example would be predicted as a win although it should be an obvious loss.

Moving on to the actual algorithm that predicts the outcome of the game based on the team's lineup, which is captures in the python script called *lineup.py.*

There were several steps that had to be performed beforehand. The data provided from the data set was in a form of a unique player id such as 45678. So the fist step was to capture all the players that played in any of the team's games and convert their ids into more simple integer ids such as 1, 2 or 3. A missing player will take the id of -1. From that point on each match will receive an additional list of unique short ids based on the one created by the DynamicLabelEncoder class.

Each player will have his own column with the value 1 if he played in the match and the value 0 if he didn't.

A quick test is then performed to make sure that the sum of the all the value in all the players coulns is equal to 11 to double check the outcome.

Three different algorithms are then performed, namely: Decision Tree, Random Forest and SVC. An MLP Classifier was also created but will be treated as a work in progress in this project.

The efficiency of each model will be evaluated by applying the accuracy metrics function to handle over and underfitting.

During the algorithm implementation, several challenges and complications occurred. The challenge that reoccurred quite frequently whenever a new list is being created was a python error indicating a mismatch in the lists size. The trick in catching these errors early on by either printing the head of the lists and more importantly the “shape”. This gives you he shape of the created object and thus makes it easier to debug and quickly fix with minimal confusion.

The other recognizable issues occurred during the label encoding steps. At the first run, the usual pandas libraries were used to encode the players unique ids. However, this created countless logical and programmatical issues that were really difficult to trace. Therefore a DynamicLabelEncoder() class was created which can be used in most of the similar cases, and is not only applicable in this application. I would encourage anyone to reuse and improve this class, since it would save a lot of time in doing it otherwise.

Refinement

On the first trial of the algorithms showed a significant amount of overfitting which was quite expected. Although it worth mentioning that the testing set still had acceptable results in the range of 60-70%.

This was the preliminary Accuracy results:

Decision Tree:

Train accuracy: 0.978

Test accuracy: 0.671

Random Forest:

Train accuracy: 0.956

Test accuracy: 0.645

SVC:

Train accuracy: 0.969

Test accuracy: 0.803

From the first glance we can see that the SVC has impressive results and based our previous benchmark of 70% no changes will be made3 for this model.

It is however worth mentioning that for this specific application, Decision Trees (and Random Forest) might be more beneficiary to additionally identify the most influential players, indicated at the top of the developed tree. This part will be elaborated in the Visualization portion in the conclusion.

In the next section, the parameters of both the remaining algorithms (Decision Trees and Random Forests) will be fine tuned to decrease this obvious overfitting and improve the accuracy for the testing data.

Results

Model Evaluation and Validation

After several trials of fine tuning for both models these parameters were chosen that proved to produce best results:

Random Forest Classifier:

- n_estimators=20,
- min_samples_leaf=4
- criterion="entropy

Decision Tree:

- min_sample_leaf = 10

Providing these parameters to the algorithms, while adding the ones already established in the previous section for SVC, this is the final accuracy of the provided algorithm:

Decision Tree:

Train accuracy: 0.798

Test accuracy: 0.724

Random Forest:

Train accuracy: 0.781

Test accuracy: 0.829

SVC:

Train accuracy: 0.969

Test accuracy: 0.803

It is clear from the below values that the training model accuracy have decreased to tackle overfitting, however the testing model accuracy have improved drastically reaching around 83% in the case of decision trees.

It is also worth mentioning that the random state of each algorithms have been changed and the result were usually similar to the ones indicated above, as an additional confidence boost.

When comparing this to the original benchmark of 50% and the target of 70%, this model would therefor be considered highly reliable with trustworthy results.

However, it is worth mentioning that this model has still some room for improvements. This is first of all due to the fact that the team we used from our analysis (Fc Barcelona) is considered one of the stronger teams in the league, and thus the overall results could be a bit biased.

In order to test the robustness of this algorithm, a different team_id has to be plugged it and rerun the whole process, which was performed.

After choosing a couple of other teams in different league it is clear that mid-table teams, i.e. team that are neither strong or weak produced weak test accuracy results.

After plugging in three different teams the testing accuracy range dropped down to around 50-60% (53.12%, 57.01% & 56.84%), which is lower that the original target yet still higher than the benchmark. This points towards the fact that although the overall robustness of the model is considered marginally acceptable it might still some further improvements.

Justification

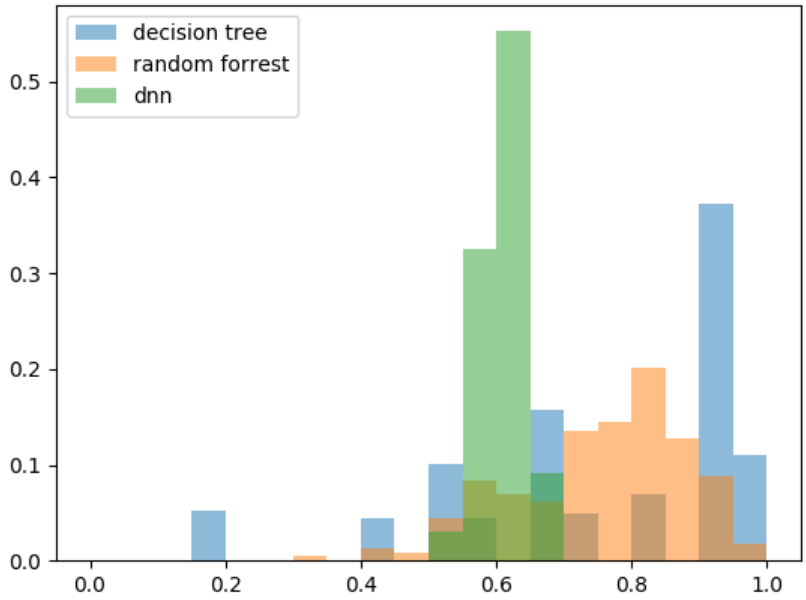
As discussed in the last section the overall performance of the algorithm has provided a reliable solution to the indicted problem.

At this point, providing a list of players to the algorithm should output an outcome with a probability of over 70% which is considered a quite reliable solution.

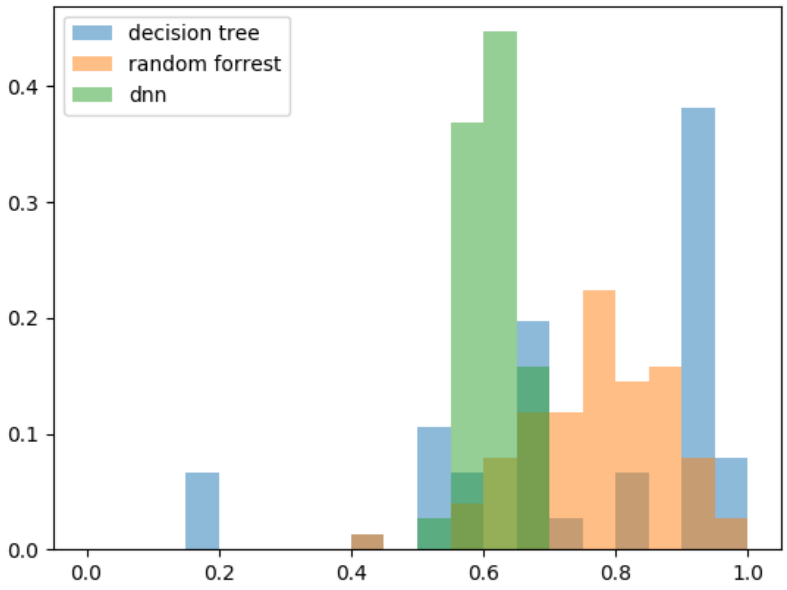
Additionally, as previously mentioned, an previously mentioned an MLP Classifier has been created to provide a different type of algorithm. After some fine tuning the DNN was able to obtain a train accuracy of around 75% and an impressive testing accuracy of around 82%.

Moreover, two histograms have been created to first visualize the routine of each algorithm (excluding the SVM) and to effectively compare the overall performance of all the models.

Training Probability Histogram:



Testing Probability Histogram:



When evaluating all models based on the histograms, one could identify the random forest classifier has a better distribution when it comes to outcome prediction.

The decision tree as expected has a more certainty in its prediction, while not covering the majority of the prediction range.

As a conclusion, I personally believe that based on the outcome and the accuracy calculation of each model, this algorithm has proved to solve the problem with a higher precision and thus provide a significant value for the process.

The algorithm I personally would use to further analyze this process would be the Random Forest Algorithm due to its wider coverage and improved accuracy, however it is worth mentioning that improving the neural network algorithm might also provide interesting insights and maybe even better overall results.

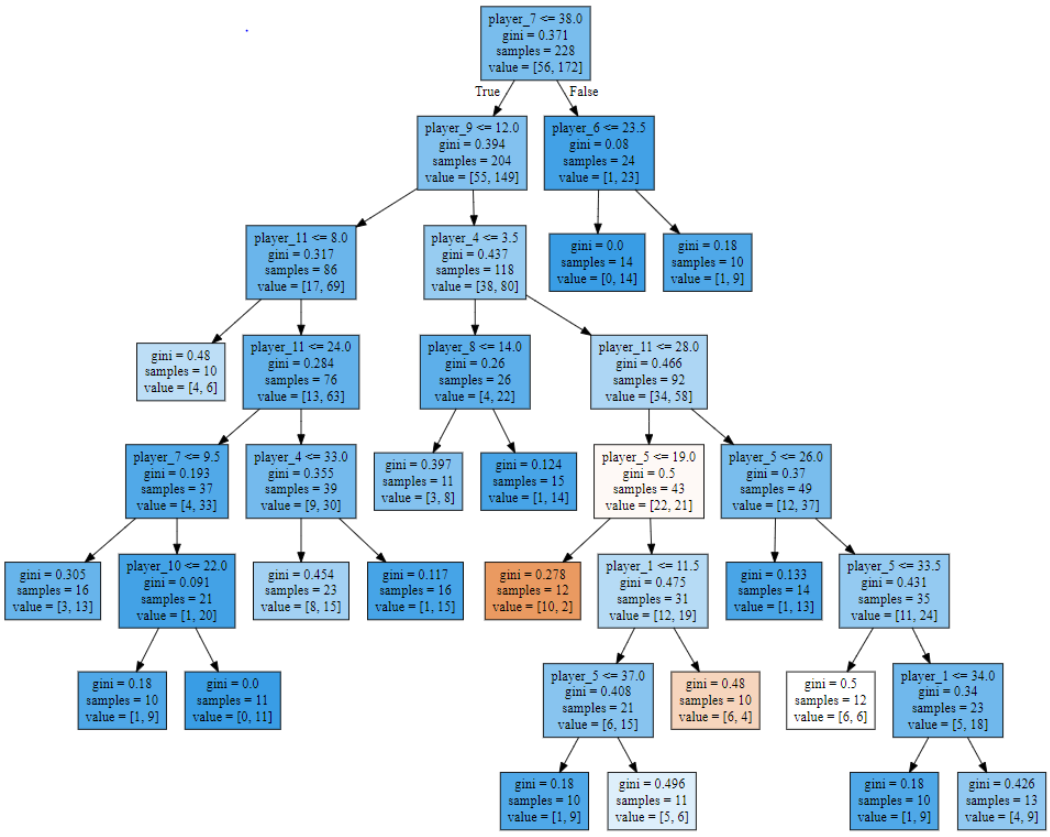
Conclusion

Free-Form Visualization

Throughout this report an algorithm was developed to provide a prediction for a match's outcome based on the lineup of any given team. The algorithm was able to produce a relatively high prediction accuracy of more than 80% when compared to the previously set benchmark of 50%.

I personally wanted to take it a step further in this section just to discuss an additional implication that this algorithm has provided during its development, namely identify the most influential players in the team chosen, which in this case was the Spanish Side FC Barcelona. Without going too much into specifics, I believe it is well known that within the last decade (2008-2016) a certain 'Lionel Messi' has proven to be the base of this Spanish team. One could easily argue that no algorithm is needed to identify that this player's participation must provide the most influence on any of the game's outcome. It would hence will be no surprise that this player should be on top of the overall decision tree.

However, when visualizing the created tree using the http://webgraphviz.com/ tool available online, the following tree is generated:



Based on this created tree, the most influential player indicated at the tip of the tree is not 'Lionel Messi' as expected, it is a holding midfield player called 'Yaya Toure'.

In other words, this player's absence and inclusion in the team's lineup has the highest influence on the overall outcome of the game.

In conclusion, I firmly believe that predicting the outcome of the game based on the provided lineup is crucial to any overall developed odds. Additionally providing insights on the most influential players is a valuable added value.

Reflection

This project was focused on how lineups and team selection can influence the outcome of any match.

Although historical data and team ranking might have the biggest influence in deciding ration and percentages to derive betting odds, one should never overlook the contribution of each player that can directly affect the game results.

The data provided by Kaggle was first used to capture the matches that the chosen team has played within the span of around 10 years.

After processing the data and adding some useful features, several operations has been performed to test the overall data such as the amount of goals and possession need for a team to win, and to teach the algorithm what winning a game actually mean, basically outscoring the opponent.

Afterwards, the data was utilized by several classification algorithms such as decision trees, SVM and Random Forests to establish a process that effectively predicts the match's outcome based on the provided lineup of players.

The accuracy of this model was then calculated to ensure that there is minimal over or underfitting in each model, and to provide a proper stopping point based on the previous benchmark.

Throughout this process there were several learning opportunities that one would never experience unless you actually perform hands on activity. The first and most important lesson learned that one should always set a side more than 50% of the time for data processing. Although at the end of the day the machine learning algorithms produce the overall value, spending enough time initially on organizing and preparing the data will save a lot more time that doing it as one goes and will make life so much easier on the long run.

Another big challenge and accordingly an important lesson learned was the label encoding step in the code as previously mentioned. It took me a lot of time to really try to do it using the usual tools and libraries and thus wasted a lot of time. At the end creating a sperate class that only deals with this issue in a more general way helped solve this issue and will accordingly save a lot of time in future projects.

Finally, whenever any complex errors occur the easiest thing to do is to copy the error into a search engine and look for solutions on the web, check the documentation and reach out to a peer or colleague. Most probably someone had the same issue before and several solutions are usually available, and it is god to build on other's experiences rather than wasting time going in circles trying to figure it out alone.

Overall, I personally believe that this algorithm may serve as great benefit when integrated with other algorithms that overall calculates betting odds, since it may provide a more valuable prediction based on the most vital features in any soccer game, namely the player themselves.

Improvement

It is however worth mentioning that there is certainly a room for improvement that can be built on top of this algorithm to provide more valuable application.

First of all, as previously discussed, integrating this algorithm with additional algorithms that produce betting odds or general match predictions might be extremely useful. In other words, adding the other team into the equation will definitely produce better overall results.

Secondly, this algorithm was mainly focused on either winning or not winning the game, while ignoring the fact that a big percentage of any game may end up with a draw in soccer games. Hence adding a draw scenario to the calculation may also be more beneficiary.

Moreover, using the goal difference as the dependent variable and thus turning this problem into a regression exercise by having a negative end result equate to a loss, a positive to a win and a 0 to a draw could also yield interesting findings.

Last but not least, I believe it might be also interesting to dig a little deeper into the players themselves and thus use deep learning algorithms to create player profiles, across all teams and establish the most effective lineup overall as a start and having a group of players segmented in the same group that can all have the same positive or negative contribution whenever they play, regardless of team or league.