Beating the Bookies

Bigdata soccer prediction



Brennen Bouwmeester

Youri van Delft

Willem Verhoef

Carel Schwartzenberg 15-01-2018

**Introduction**

Every year, more money is spent on professional soccer players and more bets are being put onto those players and their teams. Therefore it could be profitable to find a way to predict the outcome of a soccer match. When this is done properly, the world of soccer will change and teams will probably base their purchases on this prediction model. Interesting input values for the model could be a players physical performance. When the average physical stats of a team are known, maybe this can predict the winner accurately. Another interesting kind of data would be the odds of betting companies, or the skills of the players of a team according to the game Fifa, the biggest soccer game franchise existing. When the data of almost 25000 matches is used to predict the outcome of these matches, maybe soccer can be made predictable.

In this paper, a way to predict the outcome of soccer matches is tried to be found using a “kaggle” dataset consisting of 25000 soccer matches played from 2008 until 2016. Stats from matches played in 11 different leagues are known, including the 22 players who played in a specific match, the stats of all of these players, updated twice every season and the identity of a team expressed in 8 values. Also the betting odds of 10 different betting companies are included which represent the profit made when betting on the outcome of a match.

Using the predictors above, a prediction is being constructed using different algorithms like an Artificial Neural Network and a Bayesian Network. Then based on this prediction, the total profit made after betting on a number of matches, according to the predictions, is being calculated.

**“Goal” of the project**

To have a clear view of what has to be produced, a goal is being determined as following:

**“Making profit when betting on soccer matches based on a prediction of the outcome which is calculated using stats from the game Fifa and the odds of betting companies”**

It is very hard to state how much profit is realisable, which is why the goals just says “profit”, which means anything above zero. The reason profit is being chosen as a goal, over for example accuracy, is because of the complexity of the prediction. Normally, a problem that has three possible outcomes (win, draw loss), would be predictable with at least an accuracy of 33.3% (1/3). In this case, this percentage is different. The “home-advantage” causes an increase in the minimal accuracy. When all the matches have “home wins” as prediction, an accuracy of 47% is being achieved. Because of this, the profit will be the output variable that will be measured.

**“Assists”**

Because the main goal is very generic, in this chapter, some assisting goals are stated.

* Finding the best predictable league
* Finding the most influential player in the team
* Finding the most influential team stat
* Finding the betting company that generates the most profit

When these assists are reached, the amount of profit that can be achieved will be maximized.

**Data processing**

Before a prediction model can be used to predict the outcome of soccer matches, the available data needs to be sharpened to get ready for usage. This means removing outliers, joining tables and calculating new variables.

After removing entries that miss either the outcome of a match, any of the players, the date or any of the team identifiers. The tables “match”, “player attributes“, “league” and “team attributes” are joined in a way that the following list of variables in present in the same table. This has been executed in an sqlite editor.

(Match\_id, date, H\_pid1, H\_rating1, H\_po1, H\_team\_id, H\_buildUpPlaySpeed, H\_buildUpPlayDribbling, H\_buildUpPlayPassing, H\_chanceCreationPassing, H\_chanceCreationCrossing, H\_chanceCreationShooting, H\_defencePressure, H\_defenceAggression, B365H, B365D, B365A)

In this table, the Match\_id serves as primary key, for this is a unique value. The date is necessary to calculate the correct ratings of the players. Notice that for each player in the home and away team, the player id, the rating and the potential (pid, rating, po) are in the table (66 values), but are not in the list above, for readability.

This table will function as the base for all algorithms that will be carried out on the data in python. An interesting feature about the data is that it is very well up to date. As there are 8 years of data, players’ ratings change over time. For example Cristiano Ronaldo wasn’t as good 8 years ago as he is now. Both the players’ individual ratings as the team ratings are retrieved on a date nearest to the date of the match. So when a certain match was played on 16-06-2017, the ratings are those retrieved on the closest date before this date. This feature makes the data more trustworthy and probably better predictors.

**Data about the data**

To get a clear view of the processed data, in this chapter a summary about the data is being presented. Some interesting features are also revealed.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| League | #Matches: | Home win | Draw | Away win |
| Belgium Jupiler League | 1215 | 46,7% | 24,7% | 28,6% |
| England Premier League | 2962 | 45,5% | 25,9% | 28,6% |
| France Ligue 1 | 2864 | 45,0% | 28,1% | 26,9% |
| Germany Bundesliga 1 | 2376 | 45,4% | 24,4% | 30,3% |
| Italy Serie A | 2747 | 46,6% | 26,3% | 27,1% |
| Netherlands Eredivisie | 2035 | 47,3% | 23,5% | 29,1% |
| Poland Ekstraklasa | 444 | 46,6% | 27,7% | 25,7% |
| Portugal Liga ZON Sagres | 1198 | 44,1% | 25,9% | 30,1% |
| Scotland Premier League | 1541 | 42,5% | 23,7% | 33,8% |
| Spain LIGA BBVA | 2707 | 48,5% | 23,4% | 28,1% |
| Switzerland Super League | 1157 | 45,3% | 24,3% | 30,4% |
| **Total** | **21246** | **45,9%** | **25,2%** | **28,9%** |

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| Season | #Matches: | Home win | Draw | Away win |
| 2008/2009 | 1763 | 46,8% | 25,2% | 28,0% |
| 2009/2010 | 2462 | 48,3% | 24,8% | 26,9% |
| 2010/2011 | 2674 | 46,1% | 26,1% | 27,9% |
| 2011/2012 | 2778 | 46,7% | 25,2% | 28,1% |
| 2012/2013 | 2880 | 45,0% | 25,7% | 29,3% |
| 2013/2014 | 2777 | 46,4% | 24,2% | 29,4% |
| 2014/2015 | 2981 | 44,8% | 25,3% | 29,9% |
| 2015/2016 | 2931 | 44,0% | 25,3% | 30,6% |
| **Total** | **21246** | **45,9%** | **25,2%** | **28,9%** |

**Prediction strategies**

The first thing that comes to mind when trying to predict the outcome of a soccer match, is the skills of the players in a team. The team that has the best players will win the match, or at least has the biggest chance of winning a match. Different people have tried to declare what makes a player “good”. Mohr, (2003) focusses on the fatigue of players during a match, while Rampinini, (2007) has investigated the physical components of players to predict their performance. Svensson and Drust, (2005) claim that any measurement of players attributes can’t be used to predict player performance in a match. The complex nature of performance in competition makes this an impossible value to calculate. This would mean the players’ skills can’t be used as predictors at all. However, what these publications have in common, is that they use data only about the players performance. Maybe other attributes of a team are more influential when it comes to winning a match or losing it. The game Fifa, created by EA has ratings for all players in the game, and some characteristics about the teams as wholes. These values might be could predictors.

Another type of predictors are the odds that betting companies produce before a match is played. These odds are based on the bets of betters, and earlier played matches. These values can be used as input to an algorithm that predicts the outcome of a match. The goal here would be finding matches that the betting companies have “predicted” wrongly, but an algorithm that uses both the Fifa data as the betting odds as input can actually predict. When this is the case for a match, profit can be made. Let’s look at an example. When team A plays against team B, the odds of a betting company are 6 a home team victory, 4 for a draw and 1.53 for an away victory. This indicates a prediction that the away team will win by the betting company. When the algorithm predicts a win for the home team, which turns out to be correct, this could result in high profit margins.

**Betting strategies**

The goal of this project is to make profit in betting on the outcome of a football match. Before any algorithms are used in this, some standard betting strategies are being tried out.

**Home team always wins.** It turns out that in many sports, the home team seems to have an advantage, if this advantage is great enough, it could be a good strategy to always bet on the home team.

**Always bet on the lowest betting odd outcome.** Maybe the prediction made by the betting company is that good, that following their prediction is the best strategy.

**Random bet.** It seems to be very hard to predict the outcome of a football match, which might mean that it is so random that random betting is the best strategy.

**Betting based on the prediction of an algorithm.** Probably the best strategy to use, as it takes all the other strategies into consideration and uses the data that has the most influence to predict.

**Restrictions**

When the most profitable strategy has been found, it is important to look at restrictions to that strategy. As matches have more attributes than the ones used in the algorithm prediction, it might turn out that the matches in a certain league are easier to predict than those in another league. Also it might be the case that matches played in the end of the season are easier targets, or that matches get more predictable every year. A third option would be to combine the strategies, only betting on a match if the prediction of the algorithm says that the home team wins. In short the dimensions that can be explored are time, league and the outcome prediction.

**Predicting the outcome**

Now that it is clear what has to be explored exactly, these strategies can be tried on the created data table. The first step is to turn the number of goals of both the home and away team into one variable, which has a value of 1 (home team wins), 2 (draw) or 3 (away team wins). Now predictions can be made. For every match in the database, an output of one of these three numbers has to be generated, so that it can be compared to the real outcome and some prediction measurements can be found. When this is done for all of the strategies, the best strategy can be chosen and the restrictions can be tried out to tweak the strategy. When it turns out the best strategy seems to be the use of some prediction algorithm is the best strategy, the parameters of this algorithm can be tweaked as well.`

**Measuring the prediction**

There are several ways to measure if a prediction was successful or not. In this chapter the measures that will be used are established, along with an explanation. Next to the measurements that usually take place within predictions, this prediction problem has a twist. In the end the goal is to make profit out of betting on soccer matches, so that means profit on its own is a plausible metric. Also, even when the usual metrics seem to be high, this doesn’t mean the profit will be high. It is important that the profitable matches are being predicted well, and not so much that as many matches as possible are predicted well.

More in depth, this profit comes from a number of matches, so maybe the profit per match would be a better choice, which leads to profit margin per match as the input bet can of course alter the profit. Imagine a profit margin of €10,- that comes from 2 matches and the same profit coming from 4 matches. With a budget of €4,-, in the first case a profit of €20,- can be achieved, while the second case only establishes a profit of €10,-. However, when all the profit comes from a few matches, this comes along with a high risk. In future predictions, this 1 match could turn out to be wrongly predicted. This wouldn’t be as much of a problem if the budget was divided over more matches. The balance between profit per match and risk needs to be found.

**The prediction algorithms**

As there are lots of algorithms that can be used to produce a prediction, a selection is made here based on what is normally used in predictions in sports. With every algorithm comes a short explanation and the parameters that are important to keep in mind. All the algorithms below together with the betting strategies named above will be superfluously examined. A selection is being made from these strategies and the best scoring algorithms will be examined more thoroughly.

**Classifier selection**  
In order to make sure that the method chosen to predict the outcome of the matches is suitable, the right classifier needs to be found. To find this classifier a number of classifiers where tested to see if their results were good enough. In total 9 classifiers were tried and there results and parameters listed.  
Since not all parameters where used to predict the soccer matches, not all parameters will be explained in what their function is. Only for the parameters that were used will an explanation be given of their use and arguments given to why the chosen values were chosen.   
  
**LinearSVC**   
Although the linear support vector machine was originally meant to be a binary classifier, it can be expanded to a multi-class classifier, by splitting the problem into binary problems. The algorithm creates a hyperplane that separates one class from the other.

|  |  |
| --- | --- |
| penalty | (default = ‘l2’) |
| loss | (default = ‘squared\_hinge’) |
| dual | (default = True) |
| tol | (default = 1e-4) |
| C | (default = 1.0) |
| multi\_class | (default = ‘ovr’) |
| fit\_intercept | (default = True) |
| intercept\_scaling | (default = 1) |
| class\_weight | (default = None) |
| verbose | (default = 0) |
| random\_state | (default = None) |
| max\_iter | (default = 100) |

**OneVsRestClassifier**  
The OneVsRestClassifier produces a single classifier per class, that sees it’s samples as positive, and the others as negative. Then it calculates a confidence score, which makes sure a sample can’t be part of two or more classes. The algorithm has only 2 parameters that can be tweaked, one of which is the chosen estimator to use for the classifier.

|  |  |
| --- | --- |
| estimator |  |
| n\_jobs | (default = 1) |

**DecisionTreeClassifier**

This algorithm creates a tree where samples go to one of the branches until a leaf node has been reached. This leaf node represents the class in which the sample belongs.

|  |  |
| --- | --- |
| criterion | (default = “gini”) |
| splitter | (default = “best”) |
| max\_depth | (default = None) |
| min\_samples\_split | (default = 2) |
| min\_samples\_leaf | (default = 1) |
| min\_weight\_fraction\_leaf | (default = 0) |
| max\_features | (default = None) |
| random\_state | (default = None) |
| max\_leaf\_nodes | (default = None) |
| min\_impurity\_decrease | (default = 0) |
| class\_weight | (default = None) |
| presort | (default = False) |

**Naive Bayes classifier**An important feature of the Naive Bayes is that it assumes independency between all the input features. So that the rating of the keeper of a team has no correlation to the striker of the same team. In practise this will not be the case, but even though this assumption can’t be lived up to, the algorithm still seems to work quite well.

|  |  |
| --- | --- |
| priors | (default = None) |

**LogisticRegression**

This regression algorithm creates a model that predicts a categorical output based on inputs by fitting a logistic regression curve.

|  |  |
| --- | --- |
| penalty | (default = ‘l2’) |
| dual | (default = False) |
| tol | (default = 1e-4) |
| C | (default = 1.0) |
| fit\_intercept | (default = true) |
| intercept\_scaling | (default = 1) |
| class\_weight | (default = None) |
| random\_state | (default = None) |
| solver | (default = ‘liblinear’) |
| max\_iter | (default = 100) |
| multi\_class | (default = ‘ovr’) |
| verbose | (default = 0) |
| warm\_start | (default = False) |
| n\_jobs | (default = 1) |

**RandomForestClassifier**  
This ensemble learning method constructs a multitude of decision trees. The mean mode of the classes is being selected instead of the output of just one tree, which solves the overfitting problem of decision trees. For this classifier there are a total of 16 parameters that can be specified. Below is a table with the parameters and which value was chosen for it.

|  |  |
| --- | --- |
| n\_estimators | (default = 10) |
| criterion | (default = “gini”) |
| max\_features | (default = “auto”) |
| max\_depth | (default = None) |
| min\_samples\_split | (default = 2) |
| min\_samples\_leaf | (default = 1) |
| min\_weight\_fraction\_leaf | (default = 0) |
| max\_leaf\_nodes | (default = None) |
| min\_impurity\_decrease | (default = 0) |
| bootstrap | (default = True) |
| oob\_score | (default = False) |
| n\_jobs | (default = 1) |
| random\_state | (default = None) |
| verbose | (default = 0) |
| warm\_start | (default = False) |
| class\_weigtht | (default = None) |

**MLPClassifier**The multilayer perceptron uses neurons that have a trainable threshold which treats inputs, and passes on outputs based on the activation function until the output layer has been reached. This layer outputs one of x classes. Then backpropagation takes place based on the error.

|  |  |
| --- | --- |
| Hidden\_layer\_sizes | (default = (100,)) |
| Activation | (default = ‘relu’) |
| Solver | (default = ‘adam’) |
| Alpha | (default = 0.0001) |
| Batch\_size | (default = ‘auto’) |
| Learning\_rate | (default = ‘constant’) |
| Learning\_rate\_init | (default = 0.001) |
| Power\_t | (default = 0.5) |
| Max\_iter | (default = 200) |
| Shuffle | (default = True) |
| Random\_state | (default = None) |
| Tol | (default = 1e-4) |
| Verbose | (default = False) |
| Warm\_start | (default = False) |
| Momentum | (default = 0.9) |
| Nesterovs\_momentum | (default = True) |
| Early\_stopping | (default = False) |
| Validation\_fraction | (default = 0.1) |
| Beta\_1 | (default = 0.9) |
| Beta\_2 | (default = 0.99) |
| epsilon | (default = 1e-8) |

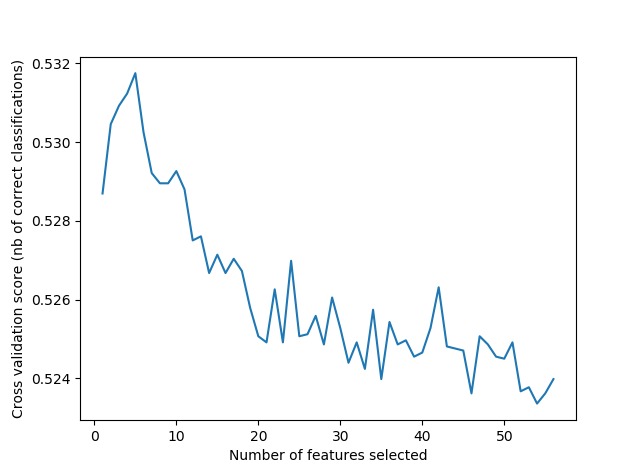
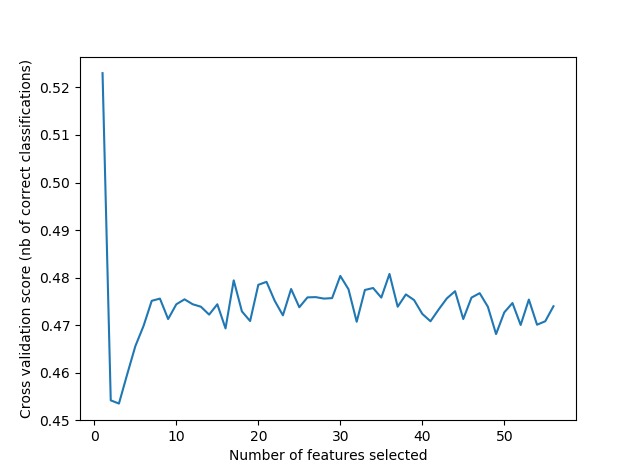
**Gradient boosting**

Like the random forest, this ensemble model uses weaker prediction models to create an output. It generalizes the models by optimizing the loss function.

**Adaboost**

Adaptive boosting can use other, weak algorithms to make a combined classification choice. The downside is that the algorithm is sensitive to outliers within the data, which are almost sure to be present in football match prediction.

**Best predictors**

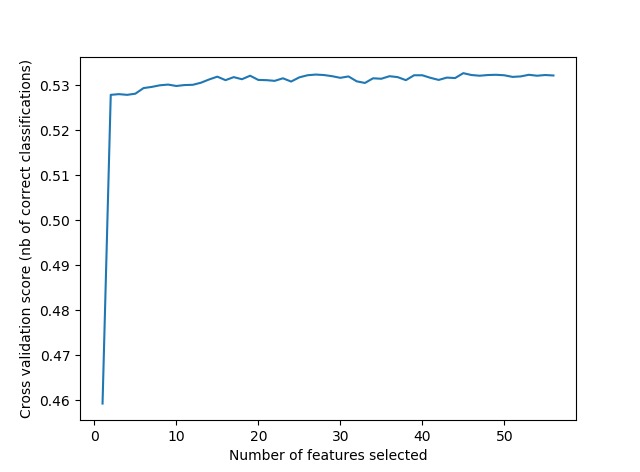
Next to the algorithm used in predicting the outcome of a match, it is necessary to indicate the most important predictors for each of these algorithms. The following figures show the best predictors for every algorithm that can be optimized parameter-wise. The algorithms examined are those chosen in the next chapter, as it is an iterative process.

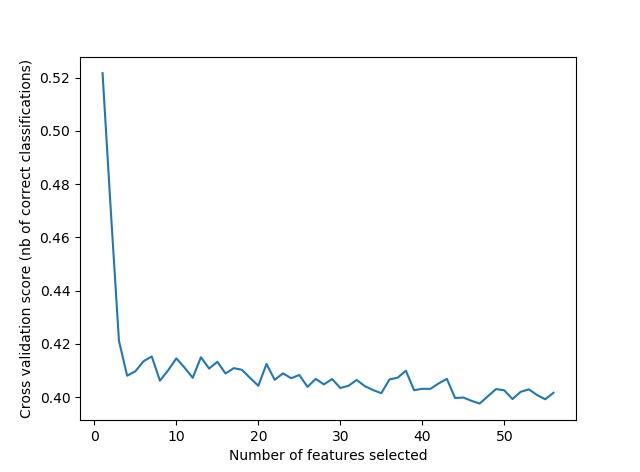
GradientBoosting, Optimal number of features: 5

(H\_rating7, H\_rating9, A\_rating4, A\_defenceTeamWidth, BWH)

RandomForest, Optimal number of features: 1

(BWA)





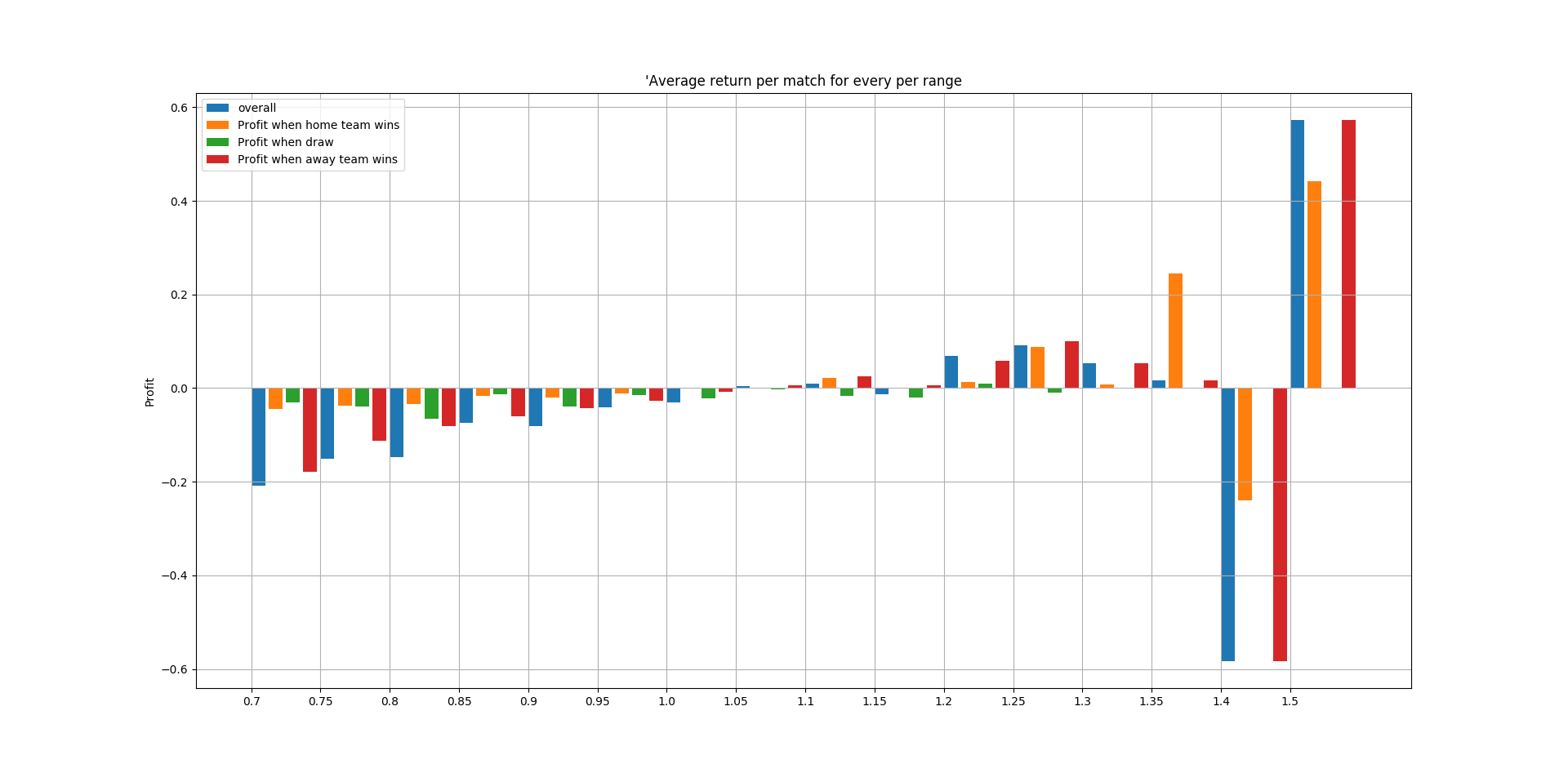
**Where is the profit**

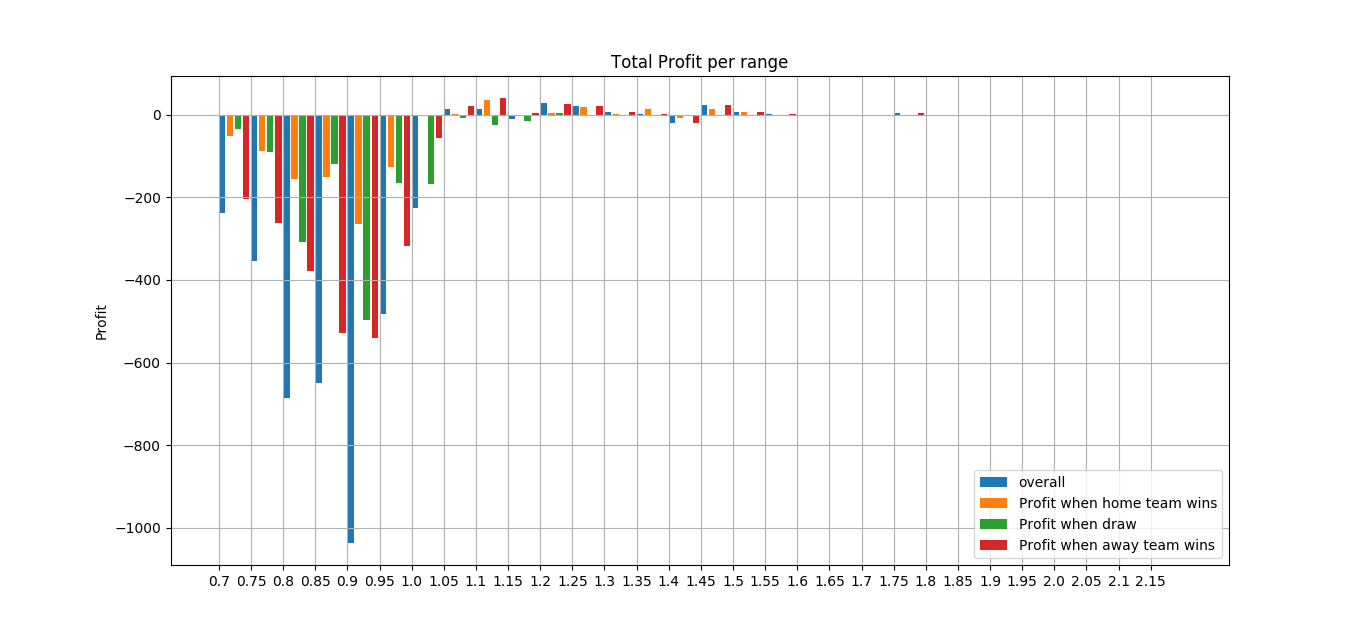
DecisionTree, Optimal number of features: 1

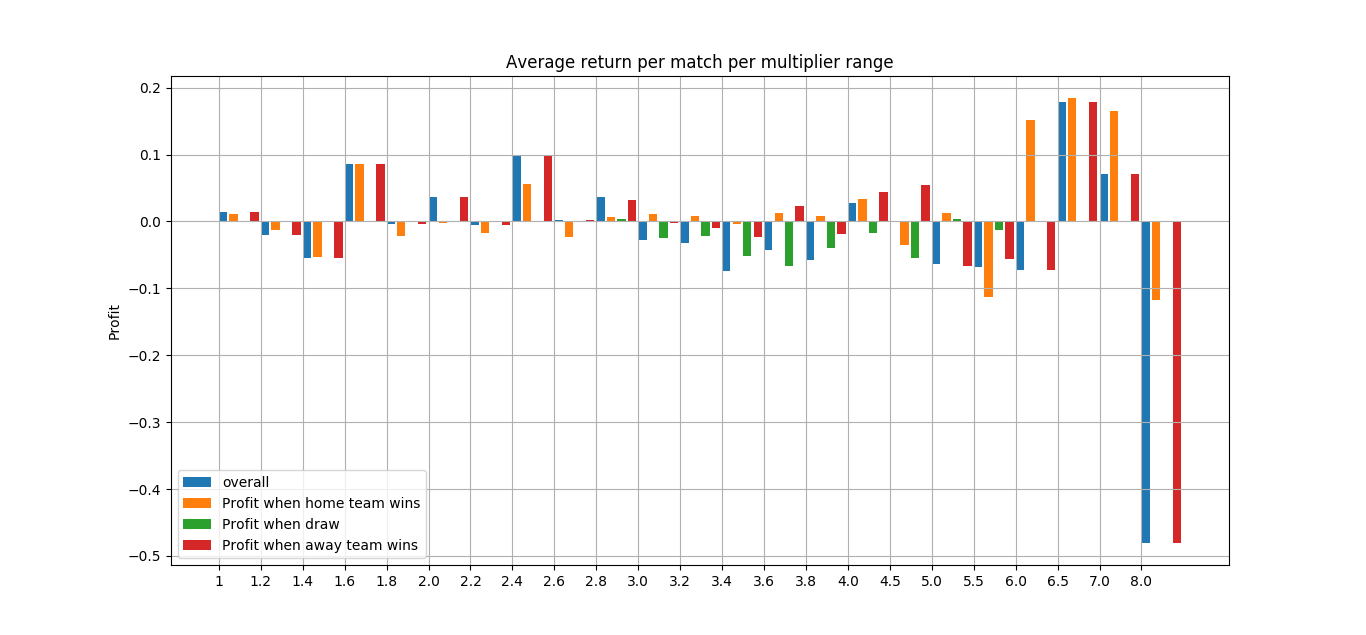
(BWA)

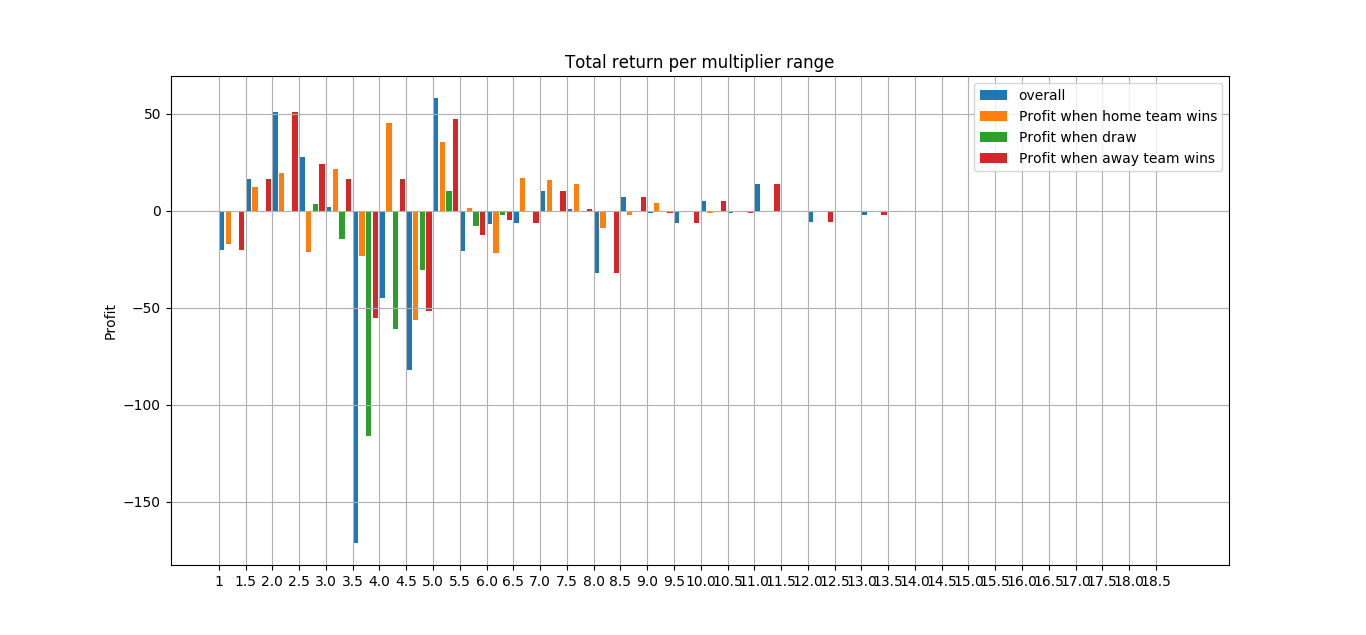
Logistic Regression, Optimal number of features: 45

(Appendix A shows which ones)

An interesting feature of a match outcome is the predicted chance that this outcome will take place. This chance can represent a risk when being combined with the profit margin supported by the betting company. This returns matches that have a high risk and matches that have a low risk. The high risk matches have a relatively small chance of return, but they do have a high return. In theory all matches that have a predicted outcome of higher than their input, are valuable matches that should be bet upon, but maybe it is a better strategy to bet on only matches that have an outcome of at least 1.2 times the input, or at max 3 times the input. Next to parameter optimization, this is also an optimization problem. The next figures show where exactly the profit can be made.







What all of these graphs show, is that there is no correlation between the height of the return of a match and the height of the multiplier (expected chance of winning times the return when winning). This means that the best estimation is that a bet should be placed on every positive expected return value.

**The best prediction strategy**

Now that a selection of promising algorithms has been made, it is time to tweak the parameters of the promising algorithms. This will output the (almost) optimal algorithm to be used in predicting match outcomes. The RandomForest, DecisionTree, Naïve Bayes, Gradient Boosting and Ada boosting turned out to be optimizable and promising for match outcome prediction. The others gave either an unsolvable error or took too long to optimize. Their parameters are optimized and shown below. The outcome below shows the best scoring parameter set per algorithm, next to the accuracy that is achieved. Also the base strategies, “Home team always wins”, “Random bet” and “lowest betting odd” are being previewed in the table. In appendix A, the according parameters per algorithms are shown.

|  |  |
| --- | --- |
| Optimized Algorithm | Accuracy |
| RandomForest | 0.440 |
| DecisionTreeClassifier | 0.463 |
| Naive Bayes Classifier | 0.473 |
| Ada Boosting | 0.461 |
| Home team wins | 0.480 |
| Random bet | 0.333 |
| Lowest betting odd outcome | 0.526 |

Notice that the values in the table are based on a balanced dataset, so a dataset where the outcomes are evenly distributed. This improves the optimization of the algorithms and keeps away the bias of the home team winning more often, which helps creating a realistic view of the prediction of the algorithms used. This is why the “standard” strategies can have higher accuracy scores. (They are based on an unbalanced database). Anyways, the accuracy won’t be the end goal for this project, because that is profit.

Now for all of these optimized algorithms, the profit is being calculated. None of the algorithms reaches a profit while the balanced dataset is being used. This is why they are being tried out on the unbalanced dataset (the real-world dataset). Unfortunately, this doesn’t generate profit either. Funnily enough, a standard logistic regression that was trained on an unbalanced dataset does generate a profit of 3%. Which indicates some imperfectness in the way of predicting here.

**Conclusion**

The goal of this project: **“Making profit when betting on soccer matches based on a prediction of the outcome which is calculated using stats from the game Fifa and the odds of betting companies”**, has been achieved, along with some interesting findings.

According to the data, the Spanish Liga BBVA is the easiest to predict, because the home team gets the most wins of all leagues in the data set. To actually find the easiest league to predict using algorithms is not possible at the moment because of a lack of data. According to the predictor selection, the most influential players of a team are the central midfielder and the striker for the home team, and the left center back for the away team. In terms of football this seems like it could be true, but because of only one algorithm actually supporting the explicit difference between these players and the rest of the team, a conclusion is not possible at this point. Thirdly it seems that the team stats have very few influence on the outcome of a match. Whether a team passes around a lot, has a high tempo in the team or defends aggressively doesn’t predict the outcome of a match very much. One team aspect that did turn out according to the predictor selection is the team width, which indicates how a team positions on the pitch, but again, it is nowhere to being proof. Lastly the betting company that can generate the biggest profit margin could not be found. The number of matches missing per betting company is too big and too different to be able to make a good estimation. Also which betting company would be best, differs per strategy. When high risks are taken, one company will reward that higher than another.

Using promising algorithms on a balanced dataset and optimizing them by using accuracy as metric, did not create the wished outcome. No profit was made with any of the algorithms. Finding out what went wrong, the algorithm linear regression with standard parameters was tried on the unbalanced dataset, which turned out to make a profit of 3%. According to the lack of time this feature wasn’t examined any further and the end result is a profit of 3% using a linear regression.

**Discussion**

**What went wrong**

The best algorithms out there can’t produce profit, but a simple linear regression can. First of all, the difficult high level algorithms might have been overfitting on the training set. Secondly the optimization takes place on the accuracy of the algorithm instead of the profit. The gap between these two metrics can be bigger than expected (if one really hard game gets predicted well and makes tons of profit, this is better than 99% of the matches making just 2 euros). For the same reason, the optimization on profit can’t be done. Thirdly the stats on which the matches are being predicted are only semi up to date. At the end of a season the stats don’t add up as well as at the beginning of a season.

The biggest mistake made was balancing the data. Reading online about this being a good thing when using algorithms like Random Forest, the dataset was balanced so that the home team win bias was removed from the data. This eventually would only have been necessary when accuracy was the metric to focus on, because an unbalanced dataset tempts to make the accuracy look higher than the algorithm can actually reach. But this project focusses on profit, and should have let the algorithms use the home team win bias instead of taking it away. This caused the algorithms to perform way worse than expected and even worse than simple strategies that do take the bias into account. If the time would let it happen, the algorithms should be optimized again but by using the unbalanced dataset. This will probably give more useful insights and generate a higher profit margin.

**Predicting next week’s matches**

To end the examination of predicting match outcomes, some matches that are going to be played in the near future will be predicted in order to find out the lifespan of the algorithm. Here are the predictions for the next round of Eredivisie matches:

[['Ajax-Feyenoord'], ['Utrecht-AZ'], ['Roda-Twente'], ['zwolle-nac'], ['ADO-Venlo'], ['Vitesse-heerenveen'], ['Sparta-Excelsior'], ['Willem-Groningen'], ['Heracles-psv']]

[[ 0.59902482 0.24172617 0.15924901], [ 0.43564353 0.26278875 0.30156771], [ 0.32511611 0.28608709 0.3887968 ], [ 0.59560922 0.2317927 0.17259807], [ 0.47529411 0.26990746 0.25479843], [ 0.53174113 0.25881096 0.20944791], [ 0.41828786 0.27339628 0.30831587], [ 0.43378844 0.27996533 0.28624623], [ 0.12972702 0.17946611 0.69080687]]

['Utrecht-AZ'] 0

['Sparta-Excelsior'] 2

These results mean that only the matches Utrecht – AZ and Sparta – Excelsior are worth betting on. Find out for yourself if these predictions were correct.

**Bibliography**

<https://www.ncbi.nlm.nih.gov/pubmed/16195009>

<http://www.tandfonline.com/doi/abs/10.1080/0264041031000071182>

<https://www.researchgate.net/profile/Samuele_Marcora/publication/6769681_Validity_of_Simple_Field_Tests_as_Indicators_of_Match-Related_Physical_Performance_in_Top-Level_Professional_Soccer_Players/links/02e7e52fe96b6da444000000.pdf>

<https://www.kaggle.com/hugomathien/soccer>

Appendix A

**Random Forest:**

{'bootstrap': True, 'class\_weight': 'balanced', 'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0, 'min\_samples\_leaf': 3, 'min\_samples\_split': 3, 'min\_weight\_fraction\_leaf': 0, 'n\_estimators': 100, 'n\_jobs': 1, 'oob\_score': True, 'verbose': 5, 'warm\_start': False}

**DecisionTree:**

{'class\_weight': 'balanced', 'criterion': 'gini', 'max\_depth': None, 'max\_features': None, 'max\_leaf\_nodes': None, 'min\_impurity\_decrease': 0, 'min\_samples\_leaf': 1, 'min\_samples\_split': 2, 'min\_weight\_fraction\_leaf': 0.1, 'presort': False, 'splitter': 'best'

**Naive Bayes:**

{Gaussian}

**Ada Boosting:**

{'algorithm': 'SAMME.R', 'base\_estimator': RandomForestClassifier(bootstrap=True, class\_weight=None, criterion='gini',

max\_depth=None, max\_features='auto', max\_leaf\_nodes=None,

min\_impurity\_decrease=0.0, min\_impurity\_split=None,

min\_samples\_leaf=1, min\_samples\_split=2,

min\_weight\_fraction\_leaf=0.0, n\_estimators=10, n\_jobs=1,

oob\_score=False, random\_state=None, verbose=0,

warm\_start=False), 'learning\_rate': 1.5, 'n\_estimators': 200}

**Feature indices Logistic Regression:**

[False False False False False False False False False False False False

False False False False False False False False False False False False

False False False False False False False False False False False False

False False False False False False False False False False False False

False True False False False False False False]

**Feature names:**

[H\_rating1, H\_rating2, H\_rating3, H\_rating4, H\_rating5, H\_rating6, H\_rating7, H\_rating8, H\_rating9, H\_rating10, H\_rating11, A\_rating1, A\_rating2, A\_rating3, A\_rating4, A\_rating5, A\_rating6, A\_rating7, A\_rating8, A\_rating9, A\_rating10, A\_rating11, H\_chanceCreationCrossing, H\_defencePressure, H\_defenceAggression, A\_chanceCreationCrossing, A\_defencePressure, A\_defenceAggression, H\_buildUpPlaySpeed, H\_buildUpPlayPassing, H\_chanceCreationPassing, H\_chanceCreationCrossing, H\_chanceCreationShooting, H\_defencePressure, H\_defenceAggression, H\_defenceTeamWidth, A\_buildUpPlaySpeed, A\_buildUpPlayPassing, A\_chanceCreationPassing, A\_chanceCreationCrossing, A\_chanceCreationShooting, A\_defencePressure, A\_defenceAggression, A\_defenceTeamWidth, B365H, B365D, B365A, BWH, BWD, BWA, IWH, IWD, IWA, LBH, LBD, LBA]