ESSEC BUSINESS SCHOOL

CentraleSupélec

BIG DATA ANALYTICS
Final Project

Navigating challenges:
How to win at playing the odds on a soccer match bet using historical data?

Flora ATTYASSE
Jennifer VIAL
Sarah BOURIAL
Arthur DEGONDE

CONTENTS

References

Dataset: https://www.kaggle.com/hugomathien/soccer

[1]https://www.bbc.com/sport/football/24354124
[2]https://www.telegraph.co.uk/news/2016/05/02/leicester-city-win-premier-league-and-cost-bookies-biggest-ever/
[3]http://iranarze.ir/wp-content/uploads/2016/12/E3227.pdf
[4]https://biblio.ugent.be/publication/5817852
[5]http://article.sapub.org/10.5923.j.sports.20140403.03.html
[6]https://www.researchgate.net/publication/229329126_Towards_the_perfect_prediction_of_soccer_matches
[7] https://www.bookmakers.bet/6850/roi-yield-sports-betting/

Introduction

Soccer is the most in vogue sport in the world. With an estimated 3.5 billion fans around the world, it is enjoyed by nearly half the world's population. Soccer is the national sport of many countries in the world and its popularity transcends borders. Further, soccer prediction is a multi-billion-dollar industry played by over 250 million football players in over 200 countries.

Betting over soccer matches outcomes is not merely of interest to fans who must satisfy their curiosity, but it is also of significant economical relevance. A BBC (2013) article estimates that the soccer match-betting industry is worth between 700 billion and 1trillion USD a year. History has shown again and again that soccer prediction can be hard. For instance, Leicester City stunned the world in 2016 when winning the English Premier league season 2015/16 after just getting promoted from the second league the year before. This came as a complete surprise not only to fans, but also the major betting companies: Ladbrokes (2016) estimated the chance of this event to 1 in 5000 before the season, costing British bookmakers a record payout of £25 million.

Given the prominence of this business in the global economy, we decide to have a look at the factors that affect a win over bets on soccer matches, and the various ways these factors can be managed so as to insure a win for the player. Here our end goal is to predict the outcome of soccer matches: *we want to find the appropriate method to ensure a bet win by least 60%, using a model that accurately predicts which team is most likely to win the match.*

Related Work

Because of the marketability of soccer, predicting the outcome of soccer has been a hot topic, one that was attempted at using a number of different approaches. The previous studies in this area have focused on two types of approaches: a goal-based and a result-based approaches. The former tries to predict the goals scored and conceded by each team whereas the latter attempts to predict the win-draw-loose outcome directly. Here our work will focus on the second category.

One common prediction method is using collective knowledge. With the availability of online platforms such as twitter, it has become increasingly easier to gain massive amounts of collective knowledge and use them for prediction, for instance using social media sentiment analysis. Other submissions include modeling teams based on their performance in the most recent history of matches. For example, the scholars in use Logistic Regression and Least Squares Regression taking as features the sum of differences in the number of cards and goals score for or against each team during the last k matches. They then use these historical data to predict the outcome of new matches. Finally, some previous works have dipped into systematically finding the most valuable predictors for soccer matches, then build upon that data to achieve maximal prediction accuracy using point spread models or multivariate regression analysis. There thus exists a good deal of literature offering a variety of statistical perspectives to identify, incorporate and predict suitable features that lead to a given outcome of soccer matches.

Dataset

Here we are faced with a multi-class classification problem over a set of input data stored in a .sqlite database including tables named Country, Player, Team Attributes, League, Match, Team, Player Attributes and Team attributes. We found our data on Kaggle, themselves extracted from the FIFA website through an API. Our dataset describes the results and statistics of matches, player, and leagues from 11 European countries, spread over 8 different seasons, with more than 11000 players and 26000 matches. Examples of schemata and data available in our original database included:

- ***Match***
  Labels: id, country_id, league_id, season, stage, date, match_api_id,… etc.
  Values: 1000, 1, 1, 20/2013, 1, 2012-07-29 00:00:00, 1223981, ..., 20747, etc.
- ***Team***
  Labels : id, team api id, team _fa api id, team long name, team short name.
  Values: (43042, 8634, 241, FC Barcelona, BAR)
- ***Player Attributes***
  Labels: id, player_fifa_api_id, date, overall_rating, potential, preferred_foot, etc.
  Values: 6176, 200810, 154238, 2014-09-18 00:00:00, 61, 66, right, medium, etc.
- ***Team Attributes***
  Labels : id, team_fifa_api_id, date, buildUpPlaySpeed, buildUpPlaySpeedClass, etc.
  Values: 1, 434, 9930, 22/02/2010 00:00, 60, Balanced, … etc.

Data Cleaning

The tables Match and Player Attributes had many missing values. Since our focus was set on matches, we decided to remove from the table all matches that were not interesting for our project end goal. We decided to keep the columns:

"country_id", "league_id", "season", "stage", "date", "match_api_id", "home_team_api_id", "away_team_api_id", "home_team_goal", "away_team_goal", "home_player_1", …, "home_player_11", "away_player_1", …, "away_player_11"

We subsequently removed the match for which one of the two teams did not appear on the Team table. Even if the method seems somewhat drastic, we still had a total of 21246 games on which we could train our model, according to a final count of 45 features.

Feature Engineering

Player Attributes

We made the assumption that players statistics were significant for our project. For every match, we had to find the most recent grades attributed to the players. The table Player Attributes displayed many different ratings, so we decided to retain only the overall rating for every player as this feature encompassed all other rating

values. Doing so, we created 22 columns containing the most recent overall rating for every player, and that for every match of the table. We then decided to aggregate these ratings by team to reduce the number of features.

Team attributes

We assumed that adding team attributes to our model could help produce good predictions, as these were central to a successful game. The table Team Attributes was filled with different ratings and their corresponding category. For example:

buildUpPlaySpeed	buildUpPlaySpeedClass
60	Balanced
70	Fast

We decided to only keep associated categories, and to do so we transformed all '……._Class' strings into numerical values for e.g. *Little|Normal|Lots to 0|1|2, respectively.* This lead us to the creation of 24 new features added to our design matrix.

We also turned all League variables (as these are a group of teams) as dummy variables. We generated 35 new features in total.

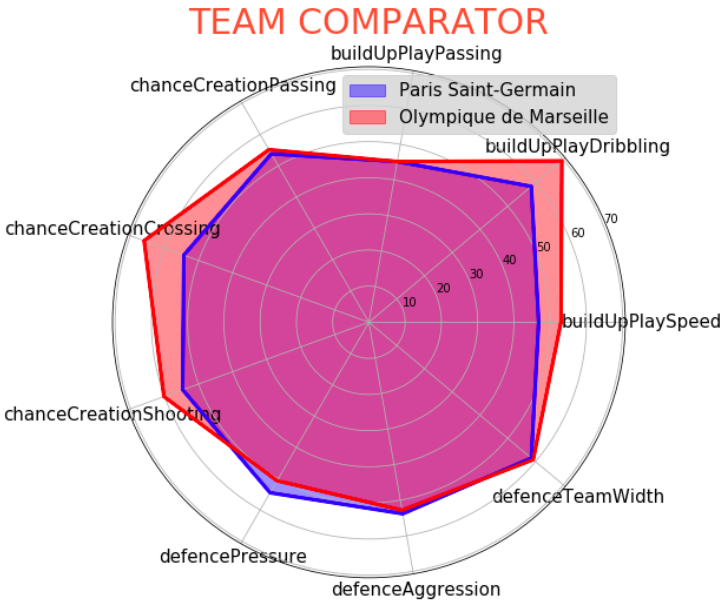


Figure 1. Example of comparison between two teams according to their respective team attributes

Matches statistics

In the football world we often take the history of a team or that of both teams to make presumptions about the result of a match. That is why we computed functions to retrieve teams' history within our database.
For the last *x* matches we managed to compute:

- Goals difference (goals scored - goals conceded) for both teams
- Games won for both teams
- Games won among games between the two teams
- Games lost among games between the two teams

Thus we added 6 new columns to our design matrix.

Model

I. Key Findings on Influential Variables

Given our large number of features, we also made an attempt at restricting our features to a smaller set by running a feature selection. Indeed, dimensionality reduction could help us avoid over fitting as well as substantially reduce computing time. We thus tried a number of different Dimensionality reduction techniques.

Recursive Feature Elimination (RFE)

We started with an RFE with fixed number of features to select (20, 30, …). Results were however not very satisfying, as for the most part they were counterintuitive and did not improve our models accuracy significantly. For instance, when running RFECV from Scikit-learn, it returned almost every features to be selected to produce the best predictions.

→ RFE 20 - Results

games_against_lost, a number of leagues (1, 1729, 4769, 7809, 10257, 13274, 15722, 17642, 19694, 21518, 24558) positioning class of home players (home_buildup and home_chance); home defender line class; classes of away team dribbling, passing, positioning (build up and chance) and pressure.

→ RFE 30 - Results

Goals difference so far of away team, games won and lost so far for home team, a number of leagues (same as before), classes of home players dribbling, positioning, passing, crossing, shooting, defender line; classes of away team pressure, aggression, dribbling, positioning, passing, shooting; and home player's ranking.

Principal Components Analysis (PCA)

Turning to PCA, the best number of components returned was 19. Thereby, our models were trained on 19 features obtained from linear combinations made of the 43 original ones.

II. Model Selection and Regularization

Given the aim of our project – predicting soccer games outcomes to insure a win for a bet, we proceeded in two stages: prediction stage, where we trained our prediction algorithms and selected the most accurate, and then

challenged our findings by comparing them to bookmaker's odds (available in our dataset, we discarded them for prediction only) and to actual matches' outcomes to finally compute the Return on Investment (ROI) of using our model.

1. Predictive Analysis

The model used was Logistic Regression (LR) from the Scikit library. LR is one of the most popular methods used for classification. LR is based on the assumption that the value of dependent variable (Y = matches) is predicted by using independent variables (X). The value of Y that corresponds to the match as either won (Y = 1), came to a draw (Y = 0) or lost (Y = -1) and is summarized by (X=x). From this definition, the conditional probability follows a logistic distribution given by e.g. $P(Y = 1 \mid X = x_i)$. This function called as regression function we need to predict Y. To tackle the multi-class classification problem, we had two alternatives:

One vs All strategy: We used the logistic regression as a binary classifier three times. Each model would give the probability of obtaining one class and not the two others.

Multinomial strategy: The algorithm is often said to be the Logistic regression generalized to multi-class classification (commonly called a Soft-Max regression).

Regularization

Considering the high volume of our train set and our large number of features we decided to apply a Ridge regularization so as to avoid overfitting.

Calibration

Our dataset is quite unbalanced, indeed wins represented twice the numbers of draw and defeats combined. To ensure good probabilities predictions our model needed to be calibrated on a dataset which does not intercept with the train set. Calibration allowed predicted probabilities distribution to match observed probabilities distribution. We used the function CalibratedClassifierCV from Scikit-learn to calibrate our models, applying it subsequently to our Logistic Regression model and then to our Soft-Max Regression model.

Evaluation

a. Parameters Tuning

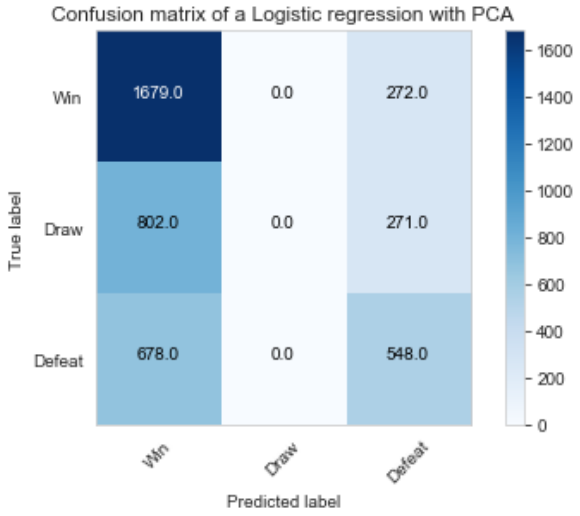
We also decided to reduce the problem of over-fitting by tuning the different parameters: Ridge penalty, PCA components. We attempted tuning our models by running the whole training job, looking at the aggregate accuracy, and adjusting our parameters accordingly, but with non-satisfactory results. We thus resorted to the StratifiedShuffleSplit function from Scikit-learn library, which is a merge of StratifiedKFold and ShuffleSplit, and returns stratified randomized folds method to automate our process of choosing the best combination of parameters. Similarly, to test the different parameters over the cross validation splits we used the function GridSearchCV from scikitlearn.

b. Cross Validation

To get a good estimate of our out-of-sample performance, we then turned to model evaluation procedures. Since we do not have the correct predictions of upcoming matches at hand, we faced 3 options to test the

accuracy and variance level of each classifier: either we voluntarily split our data into a training set and test set, or using k-fold Cross Validation technique. While the first option was simply done using the train_test_split function, we feared our training set and test set are not split evenly and give us a biased division e.g. a new training set with half the team attributes but not the other half, which could have us misjudge model performances. We thus opted for a k-fold Cross Validation.

The idea was to partionate the dataset randomly to create sized-equal subsets, we define one as a test set and all the others as training set and this method is reiterated through all the k-fold. This cross validation was made by splitting our overall training set in a ratio of 70/30 to provide for a validation set, and checking the results for our 60% prediction accuracy threshold. The aim here was to helps us to reduce chances of overfitting and biasedness in our predicted data.



As a model evaluation metric, we used from the Python library Scikit-learn the function accuracy_score. This function computed the subset accuracy by defining whether the set of labels predicted for a sample matched the corresponding set of labels. We also plotted a confusion matrix to help us distinguish between the type of errors our model is making. As we can see, the rate of true predictions is 53%, and that of Type I (false positives) and Type II errors (false negatives) is 12% and 35% respectively – we count draw as being a negative response if the outcome is actually ‘win’ or ‘defeat’.

The performances of our tested models are listed in Table I. Our best model was thus the Logistic Regression performed with a prior PCA and devoid of calibration.

Algorithm	Accuracy Score Training Set	Accuracy Score Test Set
Logistic Regression (Baseline Model – No PCA)	0.5039	0.4892
Logistic Regression (PCA, No Calibration)	X	0.4915
Logistic Regression (PCA, With Calibration)	0.5202	0.5242
Soft-Max Regression (PCA, No Calibration)		
Soft-Max Regression (PCA, With Calibration)	0.5200	0.5224

TABLE I. Comparison on Accuracy on train and test sets of algorithms

2. Return on Investment

In this part, we attempted to produce betting odds and find the expectation of money won using the optimal strategy (which way to bet) given our best model. We here included the various bookmaker's odds/quotas (from different betting agencies – William Hill, Ladbrokes, Bet365… etc.) and converted these to probabilities. We then draw a comparison with the probabilities of game outcome derived from our best model. A snapshot example of the first 10 matches compared is given in Table II below. As on can observe, our model was mostly more accurate in predicting matches wins than selected bookmakers.

We then proceeded to compute our investment return in betting on a given match using our model over relying on bookmakers. Note that reward was computed as the financial profit made upon the probability of win predicted. We achieved an ROI of 0.1521 i.e. 15.21%.

	match_api_id	bookkeeper_prob	model_prob	label	bet	prob_difference
0	1732796.0	0.618058	0.885995	Win	Win	0.267936
1	674461.0	0.702875	0.885995	Win	Win	0.183119
2	1778406.0	0.759794	0.936396	Draw	Win	0.176601
3	1260175.0	0.825266	1.000000	Win	Win	0.174734
4	1260280.0	0.719784	0.885995	Win	Win	0.166210
5	1239476.0	0.778443	0.937565	Win	Win	0.159122
6	530475.0	0.658461	0.815184	Win	Win	0.156723
7	1732919.0	0.843812	1.000000	Win	Win	0.156188
8	875562.0	0.804994	0.958333	Win	Win	0.153339
9	2030393.0	0.741801	0.892940	Win	Win	0.151139
10	1239669.0	0.788746	0.935960	Win	Win	0.147213

TABLE II. Comparison of odds between bookmakers and our model.

Conclusion and Discussion

The end goal of this project was to predict the outcome of soccer matches so as to make fruitful bets. This was broken down into three main branches: (i) identifying, generating and selecting the best features to be used for prediction; (ii) establishing the most appropriate algorithm for it and (iii) maximize our prediction accuracy in comparison with market providers.

To find the best features, we ran a Recursive Feature Elimination and found around 20 to 30 features that were not satisfactory. We turned to a PCA which proved optimal in terms of validation error.

Using a number of different regularization, calibration and parameters tuning techniques, we ran multiple models to find the most suitable one for our dataset and scope at hand. Comparing our algorithms on the basis of cross validation accuracy results, our PCA-based Logistic Regression with no Calibration proved to be the most precise algorithm, leading to about 48% error. Although being only slightly better than random, this algorithm bodes really well compared to other algorithms used, and competes with a fair amount of literature values. However, there are number of ways one could improve upon this result as outlined in the following section.

Finally, our prediction accuracy proved to be satisfactory in real world setting, as it competes in precision with bookmakers' predictions over our set of historical data.

Limitations

Our best prediction accuracy result (52.42%) did not live up to our stated goal of at least 60% precision. That being said, we are satisfied with our result as given the narrow scope of methods we were free to use compared to the complexity of our project, we did get as close a result to our end goal as possible.

Moreover, this project was a hands-on evidence of the importance of starting simple, with relatively common algorithms and applying these first to our datasets before searching for complexity. Indeed, our results were thereby easily interpretable. Speaking of interpretation, our return on investment of 15.21% could seem low, however given the industry average – a 6% ROI over a season is considered a successful betting score in the soccer betting industry (see reference 8). Thereby, we consider our model to have performed well in a real-life setting.

There are a number of fronts we could explore to better our results given more flexibility and computational power. These are as follows:

- Applying machine learning approaches to the data set, particularly Neural Networks Boosting, Random Forest combined with GridSearch (to calibrate parameters) and Adaboost methods. We are confident these would optimize our prediction accuracy above our results.
- Use features used in successful literature reviews, which were not included in our dataset, to get better accuracy levels. Examples here include weather during matches, and number of yellow/red cards received by teams during the games.
- Try a larger range of training sets, preferably a more recent one, and find the optimal time to start prediction during a given season.