Predicting football results using Machine-learning

Elements of the report

Title and Authorship:

Predicting football results using Machine Learning.

Sara Shamilova sara.shamilova@ufaz.az 21622155

Hikmat Pirmammadov hikmat.pirmammadov@ufaz.az 21622221

Sadig Huseynli sadig.huseynli@ufaz.az 21622150

Fakhri Gulmammedov fakhri.gulmammadov@ufaz.az 21622154

Kanan Jafarli kanan.jafarli@ufaz.az 21621965

Abstract

This project is about learning and implementing machine learning models to predict the outcome of a football match and identify the winning team. We've been really involved in this project, starting from understanding training different classifiers on data, to those techniques to mining data. We have extracted and built our own features that calculate and provides the stats per match. Features are the main essence of our project that highly impacts are end results. Prediction of football match outcome should follow approaches that are more generalized. Hence for our project we predict outcomes of English Premier League based on the historical data of the matches and using machine learning algorithms. We gathered data from past 14 seasons, removed first 3 match weeks and extracted features like away team, home team, goal difference etc. Using seasons 15/16 were for testing results of this project.

Introduction

Index

- Updated_Scrapping.ipynb – loading and viewing the dataset using web scrapping
- PredictionPart_Updated.ipynb – main application, testing the machine learning models

Requirements

- numpy
- pandas
- matplotlib
- scikit-learn
- and all their dependencies

Data selection and extraction

We selected https://www.football-data.co.uk/ website as our main source of data. To extract the data we wrote a script that goes through each year's data and for each year it extracts fixture table after the results of each match day. We will see that the dataset has a mixture of both numerical and non-numerical features, that it contains values from different ranges, plus that it contains a number of missing entries.

Data cleaning is one of the most important steps in data mining. We will have to preprocess the dataset (correcting the data which is reduntant and fixing it with appropriate values) to ensure the machine learning model we choose can make good predictions.

After our data is in good shape, we will do some exploratory data analysis to build our intuitions. To predict which of the given two teams would win, we decided to use csv files, as they are easier to parse and operate on in python.

Machine learning models

1. Logistic Regression
2. Support Vector Machine
3. XGBoost

We split this into two sets training data and test data (12 features 1 target (winning team (Home/Away/Draw)) and compare the results of these algorithms.

Iplementation of our algorithms are done using sklearn package. On the dataset with Xgb giving the highest accuracy of 68%.

Finally, we will build a machine learning model that can predict football match outcome.

Methods

In this section we will describe methods we used for each part of our project and also we will justify why we chose a particular method.

Let's start from first part, which is web-scrapping part. For web scrapping we had to choose between two libraries. First we wanted to use "BeautifulSoup" library, which was almost perfect choice. But then after struggling with some semantic moments, we redecided to use "Selenium" library, which provides much better and easier methods to find particular elements and parse them. But there is a little disadvantage in using "Selenium" module, because it requires to install driver, suitable for each browser. for our case, means for Firefox browser, you need to install driver named "geckodriver" and add to into executable path via terminal in linux, or in system path in windows. Here is the source to install this driver https://github.com/mozilla/geckodriver/releases. And then to make it executable in linux you can follow instructions here https://askubuntu.com/questions/870530/how-to-install-geckodriver-in-ubuntu. After that you should add the path of executable driver as a parameter to the instance creation part. (We put the comments into our code). After all instrucions are done, it should perfectly work without errors. Now let's explain the code part in a nutshell. First we import such libraries, as "os" - for interacting with operating system, "csv" - for writing our parsed data into csv file format, "pandas" - for creating tables where we should store our parsed data, and "selenium".

Here we have 3 classes. "Scraper" class is base class to initialize selenium browser that other classes can inherit from. Notice that in constructor part where we initialize Firefox webdriver, we have "executable_path" parameter. Here you should put the executable driver which you downloaded and turned into executable. Besides constructor we have function "close_driver" that simply closes our browser after we scrapped all data from website. Then here comes second class "LeagueScraper", which takes as parameter the instance of base class. Its role is Scrapping csv files of previous seasons and saves them to directory defined as "DATA_DIR". Although there is a ParserError saving seasons 2003-2004 and 2004-2005 but they are not needed though. The last class is "LeaguePositionScraper" class, which also takes as parameter the instance of base class, and it scrapes league positions from Wikipedia and saves them to "DATA_DIR" directory. But unfortunately it saves league positions in wrong format, which didn't suit our model, we decided to use dataset from the internet, which we will put in folder with name "EPLStandings.csv". Now we done with first part of our project.

Here comes the second part, which is about scrapping, mining and cleaning our data in order to make it fit to the model.

The objective of this part of project is to obtain from parsed statistics the stats that we need for our model, in other words the stats that affects gameplay the most, and use them for our model.

First we import needed libraries, which are numpy, pandas, os, csv and datetime - for parsing the date. Then we import datasets scrapped from first part using pandas. Then to avoid some missing data from our datasets, we drop those rows. Then we start manipulations with features we have to obtain new features which we will need to train our model. First we parse date from string to datetime object. Then we define dataframe with columns which we will need for obtain new features, because in our datasets there are too many extra features that we don't need. Then we create functions to obtain those features, such as "home and away team goals scored and conceded", "home and away points", "home and away team forms". Then after we got all our needed statistics, we rearrange the columns according to stats that we calculated from our functions. Afterwards we got statistics from teams themselves, we would like to get stats from our dataset named "EPLStandings", which contains information about positions of teams. From there we get position of home and away team in tournament standings, in this case from last year. Also we get stats about matchweek, where each matchweek normally contains 10 games. Then we concatenate statistics into one dataframe so we can index data easier. Then we add such additional features, as "home and away teams' points streak", "home and away teams goal difference", "points difference" and "difference in last years' positions". Then we scale them by dividing these features into matchweek feature. After we got all statistics, which are our features, we define our target feature, which is "Final Time Result" and add it into dataframe. That's it. We now have all statistics related to gameplay. The only think left is to save updated dataframe into csv file named "final dataset". The second part is done.

Here comes the final episode of our project which is about applying our final dataset into machine learning. In this part we will mainly explain the reason we used undertaken models. But let's start from coding part. First we import libraries: pandas, xgboost - extreme boosting model, and mostly known library named sklearn, which contains such models as Linear Regression, Random Forest Classifier, Support Vector Machine and etc. Then we import dataset named "finaldataset.csv" that we obtained in first part. Then we start counting statisctics from third matchweek, because in first three matchweeks we dont't have enough relevant stats (Most of them are zero or even undefined) and we also drop irrelevant statistics, that we don't need for our model. In one word we need only 12 main features, which are "Home and away teams goal difference", "Home and away team points", "differents in points and last year positions", home and away teams last 3 results, and the target feature which is "Final Time Result". Then we in manual way calculate some stats for being sure everything is fine. After, we separate our dataset into feature set and target variable. Then we standardize the data in order to model to be fittable and more accurate. We also have some categorical features (categorical - not numerical), which we convert into dummy variables, in other words, we concatenating them and give variables either 0 or 1. So after preprocessing features, we now have 24 total features. And now the main objective of third part and in general, goal of our project begins. Now with

help of beautiful library named sklearn, we split our dataset into training and testing set, where X_train and y_train are respectively feature and target variables for training, and X_test with y_test are for testing. Then we create functions, which are fitting the model classifiers into our data and with help of these model classifiers predicts the result. Then we initialize these model classifiers, which are Logistic Regression, Support Vector Machine and XGBoost and use our functions to predict the result. And what you can notice, the best result we obtained for the xgboost classifier, as it has the highest F1 score and accuracy score on the training set. Afterwards we just use GridSearchCV function from sklearn library just to update our hyperparameters. Now we want to explain the main objective - the reason why we chose those classifiers:

SVM

First of all - what is SVM ? The Supervised Vector Machine. Algorithm can be used for classification or regression problems . By using the technique called the kernel trick in order to transform your data, then based on these transformations to find optimal boundary between the possible outputs. Simply put, by doing extremely complex data transformations, then figuring out how to separate your data based on the labels or outputs you have defined .

So , why do we actually use the SVM ?

First of all , it works really good with a clear margin of separation and it is highly effective in high dimensional spaces.

It uses a subset of training points in the decision function (called support vectors), so it is also memory efficient.

Also , the major advantage of the SVM algorithm is that it can be paired with the kernel trick. So, theoretically, by exploring and fine tuning kernels you may create appropriate feature spaces, where the linear classification is able to classify data created by non-linear phenomena.

There are ofcourse problem that we can not fix , mainly it is the training time , because it does not perform well when we have big data set , so the required training time will be much bigger. One of the other problems is that SVM doesnt perform well, when the data set has more noise that is target classes are overlapping.

Logistic Regression

In our project, a logistic regression model is built to predict matches results of League seasons for win or away win and to determine what are the significant variable to win matches. Our work is different from others as we use only significant variables gathered from researches in

the same field rather than making our own guess on what are the significant variables. We also used data gathered from FIFA, it showed us that including data from the FIFA could improve prediction quality. The model was built using variations of training data from seasons. Logistic regression is a classification method which can be used to predict sports results and it can gives additional knowledge through regression coefficients. The variables used are “Home Offense”, “Home Defense”, “Away Offense”, and “Away Defense”. We conducted experiments by altering seasons of training data used. Prediction accuracy of built model is more than 70%. We found our work improved significantly. Logistic regression seeks to design the possibility of an event occurring based on values of independent variables, that could be categorical or numerical. It is a statistical method that operates on data-sets having one or more independent variables which decide an outcome. The objective of this method is to compute the perfect fitting model that describes the interrelationship amidst the dependent variable and a list of independent (predictory) variables. Our problem is a multi-class classification problem as there exists more than two possible outcomes i.e, Home Win, Draw and Away Win. Hence we are going to be using Multinomial Logistic Regression.

XGBoosting

Boosting reduces variance, and also reduces bias. It reduces variance because you are using multiple models (bagging). It reduces bias by training the subsequent model by telling him what errors the previous models made (the boosting part).

There are two main algorithms:

- Adaboost: this is the original algorithm; you tell subsequent models to punish more heavily observations mistaken by the previous models
- Gradient boosting: you train each subsequent model using the residuals (the difference between the predicted and true values)

In these ensembles, your base learner must be weak. If it overfits the data, there won't be any residuals or errors for the subsequent models to build upon. Why are these good models? Well, most competitions in websites like Kaggle have been won using gradient boosting trees. Data science is an empirical science, "because it works" is good enough. Anyhow, do notice that boosting models can overfit (albeit empirically it's not very common).

Another reason why gradient boosting, in particular, is also pretty cool: because it makes it very easy to use different loss functions, even when the derivative is not convex. For instance, when

using probabilistic forecast, you can use stuff such as the pinball function as your loss function; something which is much harder with neural networks (because the derivative is always constant).

Results

In this section we will explain which results we obtained, comparison between predicted results and true labels, and also we will explain which changes can be done to obtain other result:

After all manipulations with training our model is done, we now can test our results with function predict, where we obtain predicted results, and we have true labels also in our hand. We write a small functions that compares these results and we see that the accuracy is almost the same. But now don't hurry up! We undertook two types of classifications, which are binary and multiclass, where in binary we consider only two labels, which are either W - win or NW - not win, and where in multiclass we consider three labels: Win, lose or draw. With binary we obviously obtained better result, almost 70% of accuracy. But for multiclass we obtained less, almost 60% after we updated hyperparameters. But now I want to refer to your question during presentation, where you asked why f1_score and accuracy are the same: I did a few research and came to conclusion, that I was right when I said that for binary classification problem of our model, these metrics are absolutely different, but for multiclass classification problem, where we had 3 target features, the f1_score and accuracy are the same, but not always. I used only one estimator parameter called "micro-averaging" parameter, where these metrics are actually the same, but it gives the best result among the rest parameters, which are "macro" and "weighted", where the metrics like f1_score, recall, accuracy and etc are not the same, but their values are small enough comparing to what I used in my model.

Discussion and Conclusion

We got accuracy about 60%, it means that our model is good predictor of games. You will ask why accuracy is not such high? Because there are too many factors: first we have chosen the most toughest league, where teams from last position can defeat any team from top, and also such factors as pitch and weather conditions, referees, and corruption with fixed games are also main factors, you know :)

References

1. Douwe Buursma; Predicting sports events from past results, University of Twente, 2011.
2. Nivard, W. & Mei, R. D.Soccer analytics: Predicting the of soccer matches. (Master thesis: UV University of Amsterdam), 2012.
3. Ben Ulmer and Matthew Fernandez; Predicting Soccer Match results in the English Premier League, cs229, 2014.
4. Data mining [Online]. Available: https://en.wikipedia.org/wiki/Data_mining
5. Machine Learning [Online]. Available: https://en.wikipedia.org/wiki/Machine_learning

6. A.Yezus. Predicting outcome of soccer matches using machine learning (pdf)

Appendix

Predicting the Winning Football Team

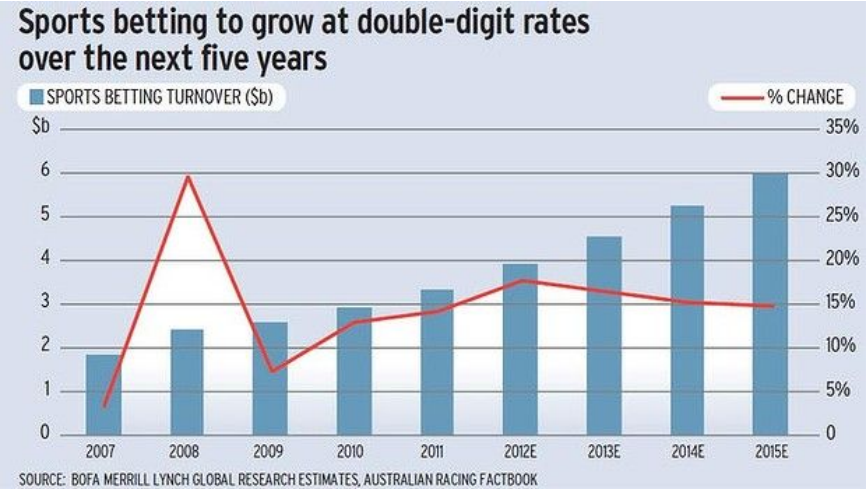
Can we design a predictive model capable of accurately predicting if the home team will win a football match?

Steps

1. We will clean our dataset
2. Split it into training and testing data (12 features & 1 target (winning team (Home/Away/Draw))
3. Train 3 different classifiers on the data -Logistic Regression -Support Vector Machine -XGBoost
4. Use the best Classifer to predict who will win given an away team and a home team

History

Sports betting is a 500 billion dollar market (Sydney Herald)



Kaggle hosts a yearly competiton called March Madness

https://www.kaggle.com/c/march-machine-learning-mania-2017/kernels

Several Papers on this

https://arxiv.org/pdf/1511.05837.pdf

"It is possible to predict the winner of English county twenty twenty cricket games in almost two thirds of instances."

https://arxiv.org/pdf/1411.1243.pdf

"Something that becomes clear from the results is that Twitter contains enough information to be useful for predicting outcomes in the Premier League"

https://qz.com/233830/world-cup-germany-argentina-predictions-microsoft/

For the 2014 World Cup, Bing correctly predicted the outcomes for all of the 15 games in the knockout round.

So the right questions to ask are

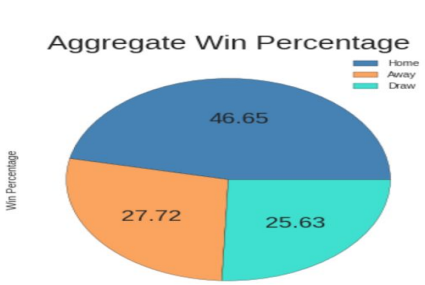
-What model should we use? -What are the features (the aspects of a game) that matter the most to predicting a team win? Does being the home team give a team the advantage?

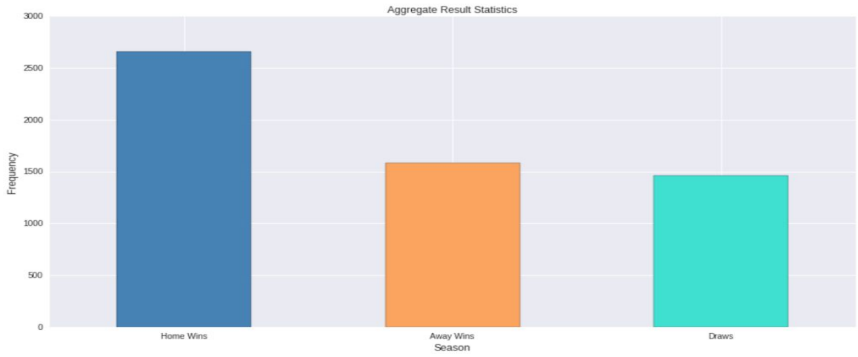
Dataset

- Football is played by 250 million players in over 200 countries (most popular sport globally)

- The English Premier League is the most popular domestic team in the world
- Retrived dataset from http://football-data.co.uk/data.php

	Home Wins	Away Wins	Draws	Total
Overall	2659	1580	1461	5700





- Football is a team sport, a cheering crowd helps morale
- Familarity with pitch and weather conditions helps
- No need to travel (less fatigue)

Acrononyms-
https://rstudio-pubs-static.s3.amazonaws.com/179121_70eb412bbe6c4a55837f2439e5ae6d4e.html