QMBU450

ADVANCED ANALYTICS IN PYTHON

Spring 2019

Term Project:

Number of Corners in Premier League Prediction Model

Oğuzhan Akan

ID: 30223

Instructor: David George Carlson

Table of Contents

1. Introduction

Every business sub-sector tend to interpret the data and draw conclusions out of it in this era, and football is one of those sectors that well suited for data usage as there are many available. Therefore I decided to predict the number of corners (target) in a football game via numerous statistics (features) that are gathered from different sources, with the aim of exercising Machine Learning algorithms that we have learned throughout the course. *Data Report* and *Model Selection and Justification* constitutes the main backbones of the project. The former explains data sources and features are to be used for the project while the latter describes the feature and model selection process.

2. Data Report

2.1 Data Sets, Span and Sources

2.1.1 Game Data (./term_project/dataset/game_data/)

England Premier League consists of 20 football teams and each of them play with one another 2 teams every season, e.g. a total of 380 games are being played. Taking the 20% test data into consideration, I decided that 5 seasons (1900 rows) would suffice – 4 seasons for training and 1 season for testing. The *game_data.csv* is the main dataset for the model and other datasets are plugged to the game data with appropriate adjustments.

The game data is simply a combination of rows at which each row explains one game – who are the teams, number of goals, shots, shots-on-target, yellow cards, red cards, and even the referees. Full column list and their descriptions can be found in Appendix 1.1.

2.1.2 Team FIFA Stats (./term_project/dataset/team_fifa_stats/)

There are many team-ranking benchmarks online, and the video game FIFA is one of them which I think quite fair in terms of scoring teams with respect to their attacking and defensing features. I gathered the data from www.fifaindex.com from the 2013-2014 to 2017-2018 seasons using Data Miner – a Chromium extension. The *team_fifa_stats.csv* contains each season's attacking, midfield power, defensing, and overall scores with their season aliases.

2.1.3 Team Values (./term_projec/dataset/team_values)

Teams' monetary values are also indicative for the game flow, especially when it comes to Premier League. So much money are being spending on building teams and transferring players, and mostly the ones that have the most money are the most successful. Therefore, I went to Transfermarkt and scraped the Premier League teams' average squad value for the last 5 seasons. The related file can be found at the directory in title with filename: *team_values.csv*.

2.2 Feature Generation (.term_project/feature_dictionary)

Obviously, downloading the data is not enough to create a model. Especially, the *game data* consists of data points which are actualized, meaning that one cannot use, for example, the number of shots-on-target in a particular game to predict the number of corners in that game. Because he cannot know how many shots-on-target will be before the game ends. Therefore, I needed to generate features for every game at which the events are already occurred.

Lastly, I decided to generate all the features that I could think of and determine which ones are going to enter the model afterwards. Table 1 explains the feature families briefly. For full feature dictionary, please check *./term_project/feature_dictionary/feature_dictionary_final.xlsx*

Table 1. Feature Families

Feature Family	Description	Type	Source
HOME_LG_GOALS_ (AWAY_LG_GOALS_)	Number of goals home (away) team scored in last 1, 2, 3 weeks in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_GOALS_.py fetaure_generation_functions/AWAY_LG_GOALS_.py
HOME_LG_GOALS_H_ (AWAY_LG_GOALS_H_)	Number of goals home (away) team scored in last 1, 2, 3 weeks in home games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_GOALS_H_.py fetaure_generation_functions/AWAY_LG_GOALS_H_.py
HOME_LG_GOALS_A_ (AWAY_LG_GOALS_A_)	Number of goals home (away) team scored in last 1, 2, 3 weeks in away games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_GOALS_A_.py fetaure_generation_functions/AWAY_LG_GOALS_A_.py
HOME_LG_SHOTS_ (AWAY_LG_SHOTS_)	Number of shots home (away) team had in last 1, 2, 3 weeks in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_.py fetaure_generation_functions/AWAY_LG_SHOTS_.py
HOME_LG_SHOTS_H_ (AWAY_LG_SHOTS_H_)	Number of shots home (away) team had in last 1, 2, 3 weeks in home games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_H_.py fetaure_generation_functions/AWAY_LG_SHOTS_H_.py
HOME_LG_SHOTS_A_ (AWAY_LG_SHOTS_A_)	Number of shots home (away) team had in last 1, 2, 3 weeks in away games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_A_.py fetaure_generation_functions/AWAY_LG_SHOTS_A_.py
HOME_LG_SHOTS_OT_ (AWAY_LG_SHOTS_OT_)	Number of shots-on-target home (away) team had in last 1, 2, 3 weeks in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_OT_.py fetaure_generation_functions/AWAY_LG_SHOTS_OT_.py
HOME_LG_SHOTS_OT_H_ (AWAY_LG_SHOTS_H_)	Number of shots-on-target home (away) team had in last 1, 2, 3 weeks in home games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_OT_H.py fetaure_generation_functions/AWAY_LG_SHOTS_OT_H.py
HOME_LG_SHOTS_OT_A_ (AWAY_LG_SHOTS_OT_A_)	Number of shots-on-target home (away) team had in last 1, 2, 3 weeks in away games in the league and their combination (sum)	Int	fetaure_generation_functions/HOME_LG_SHOTS_OT_A.py fetaure_generation_functions/AWAY_LG_SHOTS_OT_A.py
H2H_TOT_SHOTS_HA_	Head-to-head number of shots when home team was the home team and	Int	feature_generation_functions/H2H_TOT_SHOTS_HA_.py

	away team was the away team in last 1, 2, and 3 games		
H2H_TOT_CORNERS_HA_	Head-to-head number of corners when home team was the home team and away team was the away team in last 1, 2, and 3 games	Int	feature_generation_functions/H2H_TOT_CORNERS_HA_.py
HOME_L0S_FIFA_SCR (AWAY_L0S_FIFA_SCR)	Home (away) team's last season's FIFA ratings – attack, defense, midfield, and overall.	Int	feature_generation_functions/HOME_L0S_FIFA_SCR.py feature_generation_functions/HOME_L0S_FIFA_SCR.py
MKT_VAL_	Teams market values, absolute differences, ratios etc.	Float	feature_generation_functions/MKT.VAL_.py

2.3 Finalizing Raw Data (./term_project/raw_training_data)

After generating the features, the only step left was to remove redundant columns from the initial data, such as team names, referee, game result etc. Also, I filled the Null data with category averages. The reason why I handled the Nulls like this was because (I) number of Null data points were so few and (ii) all of the data is consisted of continuous features, so there would be very little bias taking the average. At the end, I placed the raw data into the directory ./term_project/raw_training_data. The raw data consists of 96 columns, 1 for target column and the rest is for features.

3. Model Selection and Justification

3.1 Changing the Target Variable (./term_project/model_selection.ipynb)

Machine Learning model inputs – features, feature selection methods, model types – occasionally change during the project phase. At first, I had decided to build a regression model on my target but after some quick model trials I decided to move on with a classification model, as Mean Squared Error would not make it possible to interpret the model performance for this specific situation. I obtained 13.29 mean squared error which is equivalent of +- 3.7 corners distant from the truth value, but I wanted to measure the model's performance not by error amount but by success (See ./term_project/initial_regression_model.ipynb). Thus, I set a threshold of 9 corners which is the mean of raw data to be the new binary target. I defined the convert_to_binary function to convert earlier target to the new one.

3.2 Feature Selection (./term_project/model_selection.ipynb)

I have ranked the features in Single Factor Analysis part (Please check the related graphs and tables at ./term_project/single_factor_analysis/images/ and the rankings at ./term_project/feature_selection/feature_ranking.csv). Now is the phase where I first drop the correlated features and then select the best features to enter the model. Feature selection phase is done in this project because there were 95 initial features and including all of them in the model might cause overfit and unstable results.

At first, I selected first 50 features from feature ranking. Then I dropped the features if they correlate with any of the other features by at least 75 percent, using drop_correlated_features

function . Lastly, I used scikit-learn's SelectFromModel to choose 5 best features from LinearSVC (Support Vector Classifier). At the end, the following features are selected to enter the model:

AWAY_LG_GOALS_H_L2W: Away team goals in last second home game
HOME_LG_SHOTS_OT_H_L3W: Home team shots-on-target in last three home games
H2H_TOT_CORNERS_HA_L3G: Head-to-head total number of corners when home team was home team and away team was away team last three games
H2H_TOT_SHOTS_HA_L3G: Head-to-head total number of shots when home team was home team and away team was away team last three games
MKT_VAL_RATIO: Market value ratio of home team to the away team

3.3 Model Selection (./term_project/model_selection.ipynb)

I selected scikit-learn's RandomForestClassifier as my model, because it's (I) an ensemble method that combines base estimators and reach a conclusion in a bagging manner so that it prevents high variance and bias, and (ii) it is a convenient classifier in terms of interpretability of the results, i.e. from which features and which thresholds does the model reach a conclusion. My parameters are:

n_estimators=1800

criterion='gini'

max_depth=2

min_samples_split=4

min_samples_leaf=2

random_state=17

max_features=5

Since Random Forest's philosophy is to create a lot of trees and obtain the result by major vote, I choose that 1800 trees with at most 2 splits would suffice. Also, since there are not so many data point in my set, I selected minimum samples required for a node to split as 4 and minimum samples required in a leaf to be 2. I let all features could enter the model with max_features property.

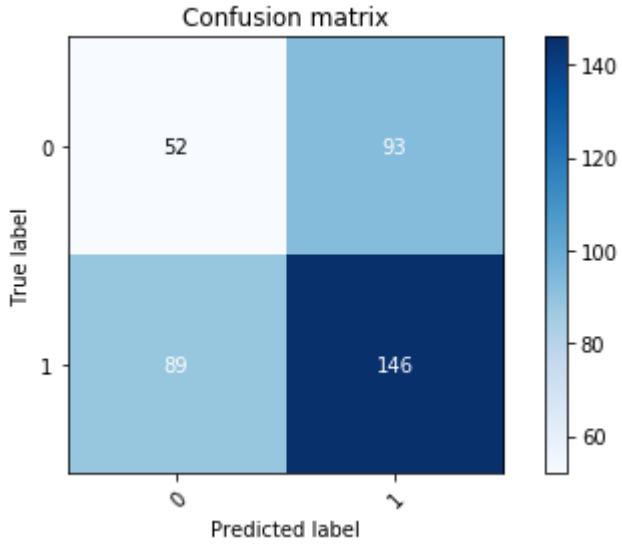
3.4 Results (./term_project/model_selection.ipynb)

In result, all of the features selected in Feature Selection phase entered into the model, with importances: 0.02487531, 0.08659659, 0.11092456, 0.19273614, 0.5848674. Thus, MKT_VAL_RATIO turns out to be the best contributing feature while H2H_TOT_SHOTS_HA_L3G is the second, which actually makes sense. The other features unfortunately contributed very little.

ROC Score (Train): 0.608937260660505

ROC Score (Test): 0.5178870139398385

Looking at the ROC scores, it is obvious that the model had some overfitting issues and features do not explain the outcome really well. Going on with the confusion matrix;



We see that we obtain relatively better results when we select 0.575 as the threshold. The model's precision is 0.61, which implies that 61% of the time when model says true, the outcome is actually true.

```
accuracy = (cm[1,1] + cm[0,0]) / cm.sum() # Overall, how often is the classifier correct?
precision = cm[1,1] / (cm[0,1] + cm[1,1]) # When it predicts yes, how often is it correct?
sensitivity = cm[1,1] / (cm[1,0] + cm[1,1]) # When it's actually yes, how often does it predict yes?
print('accuracy: {}, precision: {}, sensitivity: {}'.format(round(accuracy,2), round(precision,2), round(sensitivity,2)))
```

```
accuracy: 0.52, precision: 0.61, sensitivity: 0.62
```

4. Limitations and Challenges

The biggest limitation on a predictive model about sports is the randomness of the sports as nature. There are of course number of factors that affect the outcome (whatever the outcome is – the score, number of yellow cards etc.), but essentially sports are unpredictable. That is actually why we like them!

Secondly, I have thought of 95 features, but it could actually be thousands. I did not consider the weather, referee, squad's attributes for example. I created some ratios about past games shots, goals, but there are definitely a lot more to create in a non-linear manner. It requires more time and even group-work to think and create those features.

5. Conclusion

In conclusion, I reached a very poor-perfoming classification model. I learned, however, how to create time-dependent features, scrape the data from different sources, combine them, and try different ensemble models. I am positive that this model could be on board with some improvement!