Applying Machine Learning Methods to Document Arsenal's Success in the EPL

Johan Oakes

Final Project - CSC 322/622 – Spring 2016

Abstract

The overall goal of this report was to see the role that machine learning and data mining play in football betting specific to the English Premier League (EPL). Since I am a firm follower of The Arsenal, I wanted to tailor my findings towards their results so the dataset is split apart so that only games played by Arsenal are analyzed. Due to time constraints, the dataset was minimized to showcase important results in any sport (score, shots, etc.) and applied to one classifier: logistic regression. Predictions were based on one of three classes for each game: win, draw, or loss. The results were not expected since my classifier performed at an accuracy score of 75%, well above the normal 60-70% return rates of highly thought-out solutions. This ended in the conclusion that the preprocessing of my features needed a lot of work.

Keywords: machine learning, logistic regression, data mining, English premier league, prediction, betting

Applying Machine Learning Methods to Document Arsenal's Success in the EPL

Requirements Statement

The objective of this final project was to correctly create a model that predicts the outcome of Arsenal matches in the English Premier League close to industry standard: ~70% accuracy in terms of results of match sets, and to distinguish features that might be useful in evaluating the validity of results against market odds in order to profit off of bookmakers.

All datasets came primarily from an English source, Football-Data (football-data.co.uk/englandm.php) in comma-separated files, after reviewing other sources such as OpenFootball hosted on Github. Since the English Premier League is relatively young, the only available credible data spans from about 2000-2015. However, due to time constraints, I cut down on the amount of years and only worked with five seasons, 2000-2001 through to 2004-2005. Since each read dataset contained numerous values such as bookmaker odds to the total amount of free-kicks each team received, the features were trimmed down to each Arsenal game, fulltime result – 3 points for a win, 1 point for a tie, and 0 points for a loss as deemed standard by professional football leagues – fulltime goal difference (FTGD), halftime goal difference (HTGD), total shots taken on target difference (SOTD), and total shots taken difference (STD).

Analysis/Modeling Techniques

Since I was applying machine learning methods to the aforementioned dataset, I needed to separate the dataset into two categories: training data and testing data. For this I appointed the three earliest seasons to supplant as my training data with the latter two used for testing against the results learned from training the machine learning model. The overall goal is to predict the value of an outcome measure based on a number of input measures so by definition, supervised learning provided the path. In terms of supervised learning, I decided to implement the scikit-learn version of logistic regression since the problem is one of classification where the desired results lies in one of three values: win, tie, loss or {3.0, 1.0, 0.0}. There was no special reason that logistic regression was chosen as opposed to other classifiers (Naïve Bayes, Support Vector Machines, K nearest neighbors, etc.) other than the fact I was more experienced in the logic and mathematical demand of logistic regression.

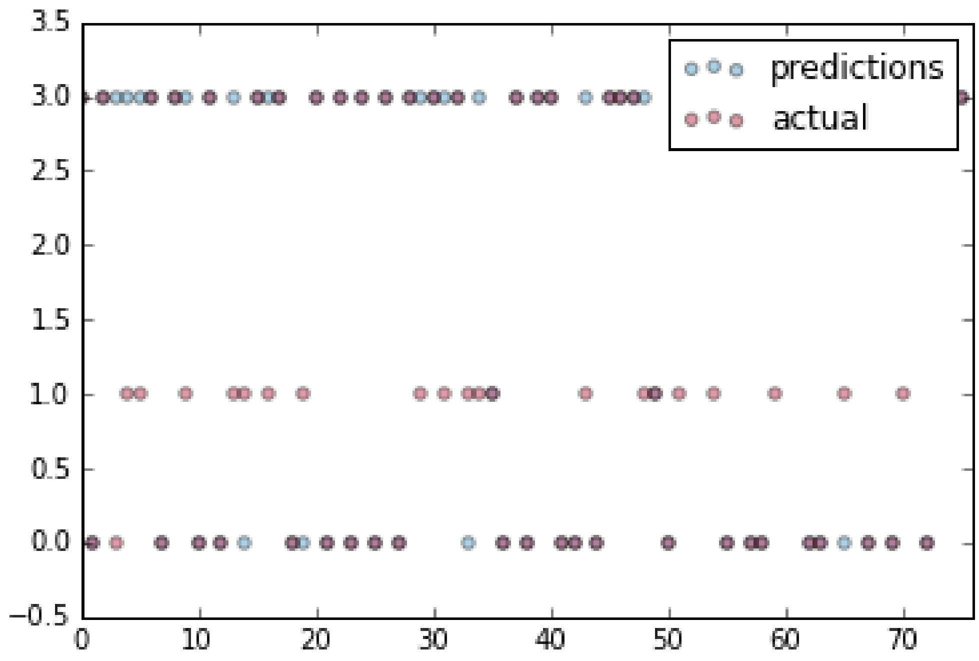
Other than using scikit-learn for logistic regression analysis, I used both the pandas and numpy packages for Python so that the entire numerical analysis could be carried out easily in Python. Pandas was used to read in the comma-separated values and to extract and minimize the important features for modeling and analysis. From there, numpy did the dirty work in numerical preprocessing beginning with normalizing the features in both the training and testing datasets so that the features rested on the interval [0, 1].

In order to validate the model against poor results and assure feature significance held true, I resorted to cross-validation in the form of Stratified K Fold since the classes are unbalanced and the dataset is not large enough to capture "rare" classes, hence, rearranging the data would ensure each fold is representative of all strata of the data instead of the bias prevalent in most classification algorithms. From the training sets I supplied, I got on average an accuracy score of ~90% when tested against the target values.

For better analysis of results, a major step in realizing the faults of the implemented model, I used the matplotlib package so that I could visually understand the results provided.

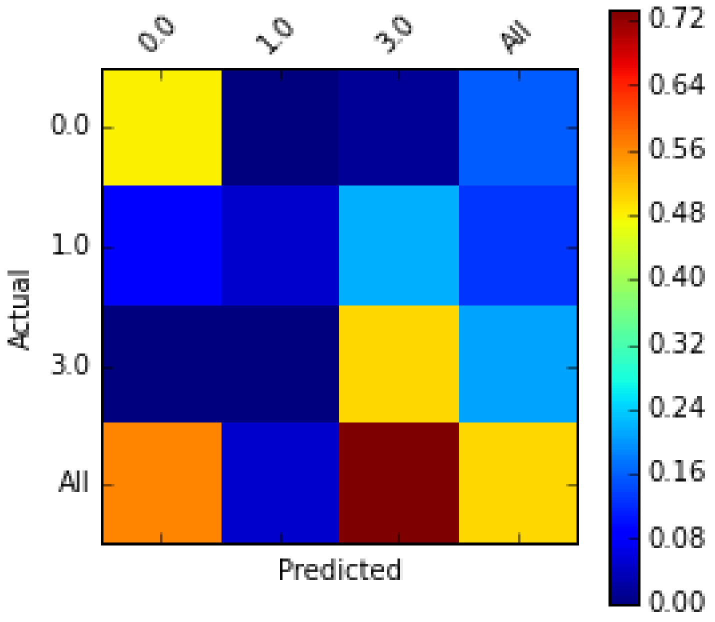
<u>**Results**</u>

Despite the effort done in cross-validation of my model and dataset, an accuracy score of 75% proved too good to be true since, as mentioned earlier, most experts and academics can only achieve around 70%. When I then tried to plot the errors using a scatter plot so that I could better understand the training error, the classifier seemed to drastically under-predict draws as shown below:



Thus, as explained earlier, since our classes are unbalanced, these classification algorithms are going to give bias towards wins and losses since win and loss classes have more examples in the training data set than the draw class. My prediction is that if we were to increase the training test size to fit a lot more seasons, our dataset would be large enough so that the linear classifier could capture the draw class and weight the classes appropriately.

The last step was to create a confusion matrix since these matrices are often used to describe the performance of a classification model. As discovered above, the fault in unbalanced classes proved to play a part in the inaccuracy of the model, so inspecting the confusion matrix would further explain the preference of classes over each other. As with many human problems, we want to minimize the possibility of false-negatives.



As can be seen again from a normalized confusion matrix, there is a stark difference between the weight of wins and losses to draws as seen by the concentration of the shades of blue. This is possible in the real world since you expect sports teams to go either way due to travel, form, motivation, and countless other factors, but in the case of the very unpredictable English Premier League, draws are about more common than losses, especially for a top team like Arsenal. In this case, we have a situation of too many false negatives such that prediction is learning towards losses, but ties or wins may actually occur.

<u>**Future Work**</u>

Future work will be primarily focused on the dataset since I believe that many of the errors generated in the project were due to poor preprocessing of the features, a relatively small training set, and ignorance of the implementations of the models on those datasets. Also, generating new features is of interest such as the aforementioned dynamic features of travel, form, player behavior, etc. The original dataset contains many unique statistics, such as overall team corner kicks, that could provide stronger results other than a simple ending score line. In terms of models, I would like to explore the implementation of Bayesian probability in order to successfully predict the winner of a future EPL season champion.