Reid Buzby & Michael O'Herron
12/7/2018
CSCI 451

Predicting Outcomes of European Soccer Matches Using Linear Regression

Introduction:
The goal of this project was to develop a model that could predict the outcomes of European soccer matches based on the characteristics of the teams involved in each match. We used a stochastic gradient descent linear regression algorithm to predict the number of goals scored by each team, which informed a prediction which team would win. We also used an SVM to classify matches into home team wins, losses, and draws directly. Lastly, we experimented with using an elementary machine learning pipeline by first using our regression model to predict the number of goals each team would score in the match, and then use those predictions as features to help the performance of the SVM classifier.

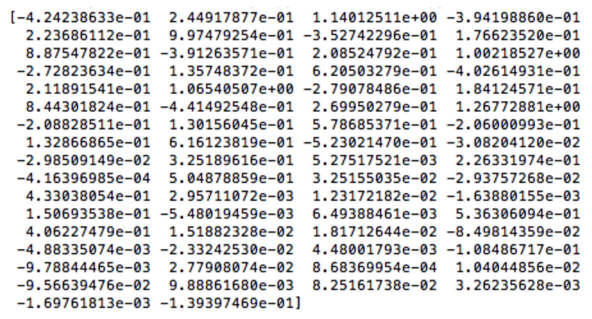
Experimental Setup:
Given a dataset of nearly 26,000 European soccer matches and statistics on almost 300 teams, we set out to develop a model that was able to accurately predict the outcome of matches. The dataset we use, "European Soccer Database", comes from Kaggle, and includes information about each of the matches, including the teams that were playing, the league and date of the match, the three-way money lines (win, tie, loss) set for the match by various bookies, and specific attributes about the teams, such as how quickly the team builds up its play, how well it creates crossing chances, how compactly it defends, etc.. One issue with the data set was a large amount of missing values for some features. We found that about 12% of our total data set was null. We had two main thresholds to help us judge how accurate our predictions were: First, in our dataset, the home team won approximately 46% of the games. Therefore, if our model were to predict a home win for each match, it would automatically get 46% correct. The second threshold is the bookies' accuracy - they are able to predict the outcome of a given match approximately 53% of the time. The most optimal outcome would be to be to create a model that outperforms the bookies' accuracy rate; however, given that we are going off significantly less information than what the bookies were using, a model that matches or slightly underperforms the bookies' prediction rate would also be considered successful.

Data Setup
An important element of this project was formatting the data into a structure that was conducive to machine learning operations. This involved merging two distinct datasets, the matches and the teams, into one dataset. Since the team dataset took into account the changing characteristics and playing styles of the teams over the eight year span of the match data, we had to pair each match with the two teams' corresponding characteristics at the time that the match was played. Additionally, several of the team attributes are given as qualitative descriptions, which we translated into numbers so that they could be learned. For example, each team's defensive aggression tendencies are classified into either "Press", "Double", or "Contain", which we translated into 1, 2, and 3 respectively. We also normalized all of the features in order to perform linear regression. Our data were divided into three groups: training (~65%), validation (~17%), and test (~17%). Each time we trained our model and checked it using the validation set we shuffled the training and validation sets together before separating them to avoid overfitting on one single training set.

Linear Regression

Once the data was formatted, we ran the Stochastic Gradient Descent Regression algorithm from SKlearn's linear model package on the training data to predict the number of goals the home and away team would score in each match. The output for the number of goals a team score was always some real number, which we were trying to get as close to the integer number of goals that were actually scored in the game.

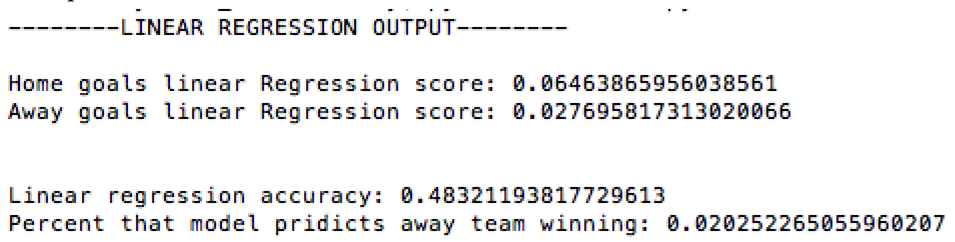
Checking the accuracy of each model using the validation set, we tweaked many of the parameters that were available in the model to first minimize the bias and then variance. Among the things we tried was changing the learning rate (α), changing the type of regularization that we were using (from the standard L2), and creating polynomial features. The polynomial features we created were the product of the different playing style scores for attack, defense and build up play. For example, we multiplied the "Defensive Pressure" score and the "Defensive Aggression" score to make an overall defense score. We also changed the cost function from the default "squared loss" to the "huber" loss function, which is less sensitive to outliers. We did not have the option to alter the number of iterations or epochs, because the regression would automatically continue to run until the cost function no longer improved after each iteration. For this reason, we do not have any graph showing the relationship between the number of iterations and the cost function - each time the model was fit, the cost was only available at the end. We checked that the algorithm was converging to the same thing every time simply by looking at the coefficient matrix for the features and checking that they were close to identical after each fit. Because it was running on a different training set each time, the coefficients were not always perfectly identical, but they were very close every time. Below is an example coefficient matrix generated by the model where each entry is the weight on a given feature:



```
[-4.24238633e-01  2.44917877e-01  1.14012511e+00 -3.94198860e-01
  2.23686112e-01  9.97479254e-01 -3.52742296e-01  1.76623520e-01
  8.87547822e-01 -3.91263571e-01  2.08524792e-01  1.00218527e+00
 -2.72823634e-01  1.35748372e-01  6.20503279e-01 -4.02614931e-01
  2.11891541e-01  1.06540507e+00 -2.79078486e-01  1.84124571e-01
  8.44301824e-01 -4.41492548e-01  2.69950279e-01  1.26772881e+00
 -2.08828511e-01  1.30156045e-01  5.78685371e-01 -2.06000993e-01
  1.32866865e-01  6.16123819e-01 -5.23021470e-01 -3.08204120e-02
 -2.98509149e-02  3.25189616e-01  5.27517521e-03  2.26331974e-01
 -4.16396985e-04  5.04878859e-01  3.25155035e-02 -2.93757268e-02
  4.33038054e-01  2.95711072e-03  1.23172182e-02 -1.63880155e-03
  1.50693538e-01 -5.48019459e-03  6.49388461e-03  5.36306094e-01
  4.06227479e-01  1.51882328e-02  1.81712644e-02 -8.49814359e-02
 -4.88335074e-03 -2.33242530e-02  4.48001793e-03 -1.08486717e-01
 -9.78844465e-03  2.77908074e-02  8.68369954e-04  1.04044856e-02
 -9.56639476e-02  9.88861680e-03  8.25161738e-02  3.26235628e-03
 -1.69761813e-03 -1.39397469e-01]
```

We used the regression model's built-in "score" metric to determine how accurate our regression is. This score is defined as $(1 - \frac{u}{v})$where $u = \sum_{i=1}^{m} (h_\theta(x^i) - y^i)^2$and $v = \sum_{i=1}^{m} (y^i - \mu)^2$. The best score is 1, and scores can be infinitely negative. Originally our score was hovering around -65, but with all of our adjustments we were able to get it to about 0.07 for the home goals and 0.03 for the away goals.

Once we were satisfied with regression for the home and away team goals, we compared the two to see how well the linear regression was predicting the results of the games. If the number of goals for the home team was predicted to be greater than that of the away team, we would classify it as a home win, otherwise it was an away win. We ignored the possibility of a tie because the data showed that ties are predicted about 0.004% of the time by bookies. With this method, our accuracy was about 48.1%. This basically meant that in nearly every game, the linear regression predicted the home team to score more goals than the away team, with the exception of only the most unbalanced matchups. This is an example of the output of our linear regression on a test set with the huber loss function and epsilon = 5:



```
--------LINEAR REGRESSION OUTPUT--------

Home goals linear Regression score: 0.06463865956038561
Away goals linear Regression score: 0.027695817313020066


Linear regression accuracy: 0.48321193817729613
Percent that model pridicts away team winning: 0.020252265055960207
```

Thus, using linear regression to predict the outcome of soccer matches was demonstrated to be relatively ineffective.

SVM Classification

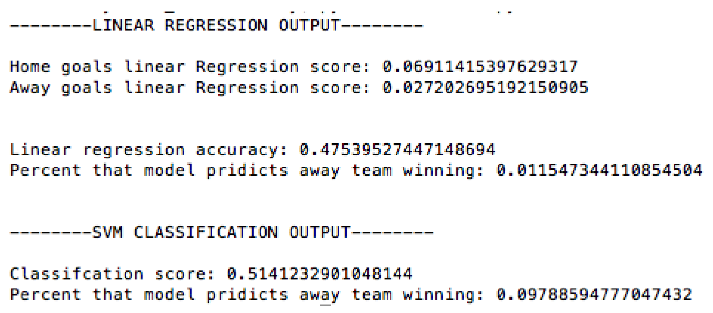
To better predict the results of games we used SKLearn's "support vector classifier" SVM to do a more traditional classification problem. We used the same data that we used in linear regression (including the polynomial features we created), and classified a home win to be equal to 1, a draw equal to 2, and a loss equal to 3. We also used the same technique of randomizing the training and validation data to minimize the variance, and with the same ratios of training, validation and test data. Again, the default parameter for the number of iterations was set so that it would continue iterating until the cost function reached a minimum.

The kernel that yielded the most accurate predictions turned out to be the default, Radial Basis Function kernel. This kernel has two main parameters, C and γ, where C is the regularization term and γinfluences the weight of each datapoint. With the default value of C = 1, the SVM classified every match as a home win, resulting in a 46% accuracy. In order to lead the model away from this (at the expense of potential overfitting), we increased the value of C to be much larger. An analysis of the bookies' predictions showed that they predicted the away team to win about 13% of the time. With each test we checked to see if any away losses were being predicted, and if so, at what level of accuracy. We settled on C = 5000, because this was the point at which the accuracy seemed to be the highest, and the model was willing to classify games as away losses. However, with a C value this high, overfitting was certainly a problem, so the validation accuracy. We ended up getting the model to predict an away win about 10% of the time, which was as high as we got without getting significant variance.

The highest accuracy we were able to achieve on the training and validation sets was 52.6%, which is very close to the bookies' average prediction rate of 53%. Given the very high value of C that was necessary to push the model away from predicting all home wins, there was a non-zero variance. The accuracy for the test set came out to be 51.5%, just under the bookies' rate of prediction.

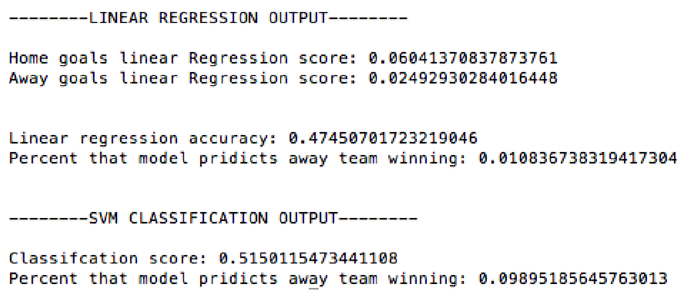
Experimental Pipeline

One idea we had to improve our SVM model's accuracy was to feed the output of the linear regression model to the SVM model as new features. We took the predicted values of home and aways goals scored from the linear regression and appended them onto the training and test sets for the SVM model. We were hoping these new features would increase the classification accuracy, but the accuracy barely changed. Here is an example output of our model when the linear regression output was added as features:



```
--------LINEAR REGRESSION OUTPUT--------

Home goals linear Regression score: 0.06911415397629317
Away goals linear Regression score: 0.027202695192150905


Linear regression accuracy: 0.47539527447148694
Percent that model pridicts away team winning: 0.011547344110854504


--------SVM CLASSIFICATION OUTPUT--------

Classifcation score: 0.5141232901048144
Percent that model pridicts away team winning: 0.09788594777047432
```

The classification score for the SVM model shows 51.4%. Now here is an example output without the linear regression output added as features:



```
--------LINEAR REGRESSION OUTPUT--------

Home goals linear Regression score: 0.06041370837873761
Away goals linear Regression score: 0.02492930284016448


Linear regression accuracy: 0.47450701723219046
Percent that model pridicts away team winning: 0.010836738319417304


--------SVM CLASSIFICATION OUTPUT--------

Classifcation score: 0.5150115473441108
Percent that model pridicts away team winning: 0.09895185645763013
```

This output shows a classification score for the SVM model to be 51.5%, so virtually no difference between the model ran above. While we thought creating these new features would be a good idea, it did not affect our output.

Conclusion

In our analysis, there were several things we did not do, even though we had the option to. One of the things we decided against implementing was a form of PCA dimensionality reduction. Although we intended to do this at the outset, we opted not to because we had no problem with the runtimes for any of these algorithms, and we did not have too many features to begin with. Also, we tried several different ways to do an iteration analysis - that is, check performance of the algorithms in relation to the number of iterations. However, both of the models we used set the number of iterations automatically, and only had a score once the model was finished fitting. Thus, we employed different techniques to ensure that the algorithms were converging, and spent more time looking at the other parameters.

In conclusion, we were satisfied with our results. Our ultimate goal was to get close to, or beat, the bookmakers prediction accuracy of 53%. While we never achieved a value higher than this, we were consistently getting results in the 51-53% range. Had we been given a more complete data set and more features regarding each team, 53% potentially could have been broken. However, we believe we were very successful given the data provided.