Rafeeq Shodeinde

Machine Learning Semester Project

5/6/2019

CSCI 4750/5750

Football (Soccer) Match Prediction

Problem Definition:

The objective of this project was to predict the outcome(scores) of soccer matches using match statistics such as, home team shots, away team shots, home team fouls and many more features. To accomplish this task, this project was modelled as a supervised, batch, model-based learning system, implementing two different machine learning algorithms. The algorithms used were Random Forest Regression algorithm & SoftMax Regression

The training data was scrapped from the data-hub website: https://datahub.io/sports-data/english-premier-league#resource-season-1819. When considering the process of cleaning the data before training, this was not a tedious or hard process. Columns in the training file, such as Date, Home team, Away team, Referee name, Full time result & Half time result, containing unwanted or unimportant features were simply deleted to clean up the data. The data was split into Home and Away data which will help make computation and prediction for the different goals easier. The model will be trained to perform predictions for both the home and away teams' goal. Since in this project a supervised learning model was used, in the training data, the column containing the label to be trained for are separated from other features used for training. The data was split using the train_test_split function of ScikitLearn. The entire data set was split into 20% test data and 80% training data.

Random Forest Regression:

To solve this problem, the random forest regression algorithm of Scikit-Learn library was implemented. The random forest algorithm is a form of ensemble method which uses a collection of trees to perform prediction. After the train data was cleaned leaving only the important features: Home Team Shots, Home Team Shots on Target, Home Team Fouls Committed, Home Team Corners, Home Team Yellow Cards, Home Team Red Cards, Half Time Home Team Goals. The data set was run through the Random Forest algorithm to train the model. The key parameter which played a major role in the performance of the model was the "n_estimators" parameter which represents the number of trees the model uses to train.

For this project, we began with a value of "n_estimators = 20" for the experimentation to improve the training model. The bootstrap parameter was set to "True" and random_state = 0. Bootstrap was set to "True" so the trees will be trained on random subsets of the training data set. With this parameter setting, the model was trained, and the predictions of the model and the test labels were compared. The mean squared error and root mean squared error calculated were 0.1448 and 0.3806. The root mean square error measures the error the system makes during its prediction showing that these parameters have 38% error. The n_estimators was varied from 20 to 100, 500 & 1000. It was noticed that the error generated by the model diminishes as n_estimators is increase. However, when set to 500 the decrease in error is seen to be significantly reduce. The values generated in the training were 0.3574, 0.3522 & 0.3516.

For further experimentation, the bootstrap parameter was set to False. The False setting implies that each tree would be trained on the whole train data each time. Again n_estimators was varied between 20, 100, 500 &1000. On training with n_estimators = 20, the root mean squared error was calculated to be 0.1171. Simply comparing this result to the results acquired when bootstrap = True, a 69% improvement on the training model can be seen. After training

with the various n_estimators, it was noticed that there was no improvement in the training model as the number of trees in the model increased.

Finally using the test data set aside to perform prediction, we notice very poor performances from the training model. Using the same parameter as at when training, n_estimators = 20, bootstrap = True, random_state = 0, the root mean squared error was calculated to be 0.9351. Increasing n_estimators respectively results in RMSE values of 0.9252, 0.9223 & 0.9210. We see similar trends of the error decreasing as n_estimators increased. A possible reason for the result of the difference in prediction between the training data and test data will be as a result of the model overfitting the data.

For better understanding of the error, the accuracy of the model for both the training and test set was calculated using the score() function. For the test data, when bootstrap = True and n_estimators = 20 about a 48% accuracy was measured and a 91% accuracy for the training data. When bootstrap = False, the training data had an accuracy of 98% and the test data had an accuracy of 13%. As stated in the previous paragraph the model looks to be overfitting when training. The accuracy for the other n_estimators values were also computed and seen to improve the accuracy. Regardless of the improvement the model was still overfitting and generating bad prediction for the test data.

To try and improve the result of the model, the data used for training was scaled using the StandardScaler() function of ScikitLearn. Although in theory the training data did not need to be scaled because the values of the features had similar scale, the values in the training data ranged from 0-25. This change in the setup of the model did not actually make a significant difference in the predictive outcome of the model and performed the same as when unscaled.

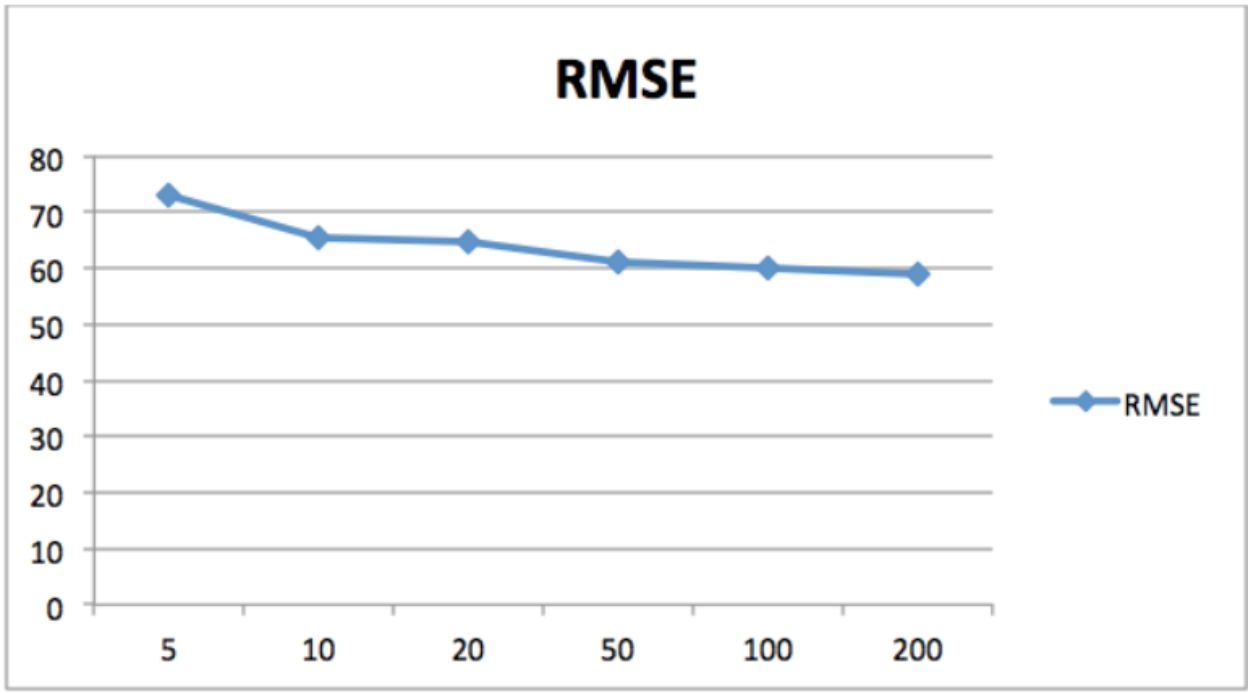

Figure 1. RMSE vs n_estimators

In the above graph, it is seen that the error values of the model decrease with the increase in number of estimators. After 200 the rate of decrease in error diminishes, so therefore 200 is a good number for n_estimators.

SoftMax Regression:

A variation of the logistic regression algorithm was also used to train the predictive model for the project. This algorithm works by computing a weighted score on the features then estimating the probability by applying the SoftMax function to the scores. The data set was run through the SoftMax regression algorithm to train the model. To set up the model, the parameters (multi_class = "multinomial", solver = "saga", C = 10, max_iter = 1000) were set. Prior to setting a value of the max_iter = 1000 the model was generating a convergence error stating the model was unable to converge. Something noticed during the experiment was that the default value for the parameter “solver = lbfgs” had to be tweaked to “saga” for better performance of the model. During experimentation, it was noticed that max_iter values above 1000 did not have any other effect in decreasing the error generated by the model. The value calculated for the root mean squared error was 0.9710. The parameters of the model were tweaked further to look for trends and decrease in error but neither occurred no matter how much, C, max_iter & solver were varied. Simply from training, it is evident that this machine learning algorithm will not be suitable for accomplishing the project goal of predicting soccer match scores. When run against the test data set, the value generated for root mean squared error was above 1. The poor performance of the model may be due to insufficient feature columns when training.