Predicting expected goals in English Premier League

Introduction

Expected goals (xG) is the most trending metrics in the world of football right now. Since it surfaced, it has been a constant topic of conversation when evaluating players' performance. Suddenly a player might not be as good anymore even if he or she scores 30 goals per season, if he should have scored 40 according to his/her xG value. In this project, I aim to construct a predictive model that can determine the likelihood of a field event resulting in a goal and quantify it as xG. I will begin by discussing the problem formulation and the methods used to build the models, followed by the presentation of results, and ultimately, the conclusion.

Problem formulation

Datasets for this project are collected originally by Italian sport company Wyscout and made public by Luca Pappalardo and Emanuele Massucco in their article "A public data set of spatio-temporal match events in soccer competitions" [1]. The data consists of one season's worth of matches from seven different competitions. In this project, I use the data only from English Premier League.

Each data point in the used dataset corresponds to an event that transpired on the pitch, such as a pass, shot, or foul. Data points hold a number or informative features such as event time, player identification, did it happen on the first- or the second half, coordinates corresponding to a position on the field etc. Data consists of a total of 16 features and 8450 data points. The data that ultimately is used in the model is both categorical and continuous.

I approached the task as a simple binary classification problem. Each event has two possible outcomes: a value of 0 indicating no goal or a value of 1 signifying a goal. Model then tries to predict the outcome thus making it a supervised learning task.

Methods

The methods can be divided into two different parts. Before diving into the modeling part, the data must be discussed. Some preprocessing had to be done and the features in their original form did not fulfill the prediction potential. After a brief introduction to the data, I will discuss the models in more detail.

Data preprocessing and feature engineering

Most of the features hold no valuable information for the model and I started by dropping most of them. I focus only on the 'shot' event since it is the only one that has the outcome the model is trying to predict. From that I extracted the predicted variable i.e., the label which is the outcome of the event 'is_goal' as the label column. Then I used the existing features to build a few new features based on the distance from the goal and the player's attributes who took the shot, totaling in 7 predictive features. The features were chosen based on three factors: domain knowledge, correlation matrix and finally information about more refined xG models.

The data were then split into three parts: training, validation, testing (70/15/15). The validation dataset ensures that the possible hyperparameter tuning can be done without any data leakage. After some closer inspection it is evident that the negative class (no goal) is highly overrepresented in the data, so I decided to balance the training data using SMOTE method for more efficient training of the models and to avoid overfitting. SMOTE oversamples the minority class synthetically to balance the difference between the classes.

Modeling

For the first model I chose to use Random Forest. Random Forest is an ensemble method that utilizes multiple Decision Trees with randomized subsets of the data. It then combines the predictions from individual trees through majority voting to come up with the final prediction. The choice of model is based on personal preferences and experience with Random Forest and also on the fact that it is known to work well with unbalanced data and can handle complex relationships between features.

My second choice of model is SVC which is known for being well-suited for binary classification and good generalization ability. It seeks to find optimal hyperplane to separate classes in the dataset, maximizing the margin between them.

I aim to measure the performance and effectiveness of the models with various metrics. The Random Forest uses gini impurity when constructing individual trees, which is a common approach for tree models. For evaluation, multiple metrics are utilized such as precision, recall, accuracy, 0/1 error and F1-score. The loss function for SVC is hinge loss since it is the mainly used loss function for Support Vector Machines like SVC. The evaluation is made using the same metrics as in the case of Random Forest.

Results

The results are divided into three distinct sections: one for each of the models and one for comparison. The metrics and visualizations presented in this section are based on both validation and test data. Both datasets are 15% of the overall data (1267 and 1268 data points, respectively). Since the dataset is known to be imbalanced towards the negative class, I will be more focused on other metrics than accuracy or 0/1-error. I evaluate both of the models by using test data and discuss the choice of the final method in the conclusion section.

Random Forest

An integral aspect of our evaluation is the determination of the optimal classification threshold, considering the precision-recall trade-off, but in almost every scenario precision and recall are < 0.50. The F1-score is the highest with a threshold of 0.3. Accuracy tends to rise with the threshold, but this is mainly due to the class imbalance mentioned earlier. More negative predictions inevitably lead to a bigger portion of correct predictions.

Since I am using non-weighted 0/1 Error that is simply $1 - accuracy$, the same principals apply that were discussed with the accuracy. Error decreases when the threshold is lifted so more predictions are classified to the negative class.

Table 1. Metrics for Random Forest

	Threshold	Precision	Recall	F1-score	Accuracy	0/1 Error
Validation	0.3	0.23	0.51	0.32	0.71	0.29
	0.5	0.31	0.32	0.31	0.81	0.19
	0.7	0.31	0.12	0.17	0.85	0.15
Test	0.3	0.21	0.56	0.31	0.74	0.26
	0.5	0.26	0.34	0.29	0.83	0.17
	0.7	0.32	0.18	0.22	0.87	0.13

Overall, the model performs well with the validation data and could be even more improved without limiting the depth of the trees. However, this would lead to a more complex model that is almost guaranteed to overfit and would not improve the performance with unseen test data.

SVC

From table 2 we can see that SVC tends to make predictions that are overconfident towards the positive class. With low threshold the model correctly classifies over 80% of the actual positives, but at the same time, most of its positive predictions are false.

With SVC the error evaluation is the same as with Random Forest. Impact of the error can be seen in the accuracy.

Table 2. Metrics for SVC

	Threshold	Precision	Recall	F1-score	Accuracy	0/1 Error
Validation	0.3	0.19	0.82	0.31	0.51	0.49
	0.5	0.26	0.62	0.36	0.71	0.29
	0.7	0.39	0.33	0.36	0.84	0.16
Test	0.3	0.14	0.80	0.25	0.49	0.51
	0.5	0.20	0.61	0.30	0.70	0.30
	0.7	0.28	0.32	0.30	0.84	0.16

What SVC does especially well is adapt to unseen data. The model does not suffer performance loss with test data practically at all. Accuracy can be affected by threshold but the cost of trade-off between precision and recall needs to be considered.

Comparison

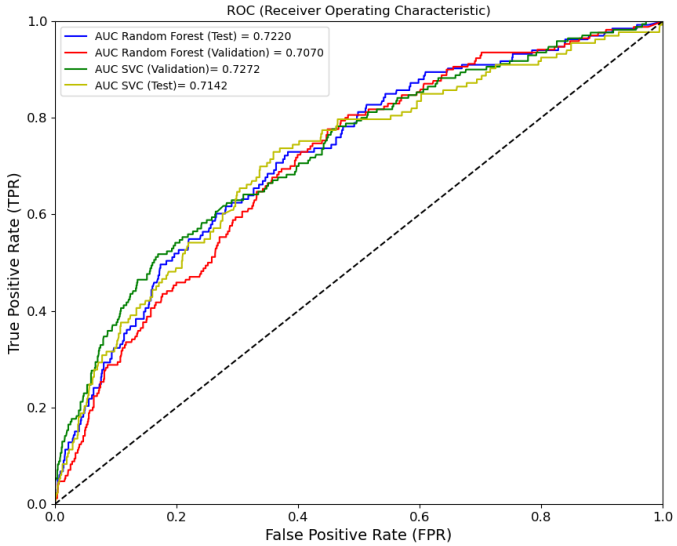

Figure 1. ROC Curve with AUC scores.

Both models share the same problem of not recognizing positive class well enough. Precision and recall move around a lot when changing thresholds, but it is more about preference than performance since the change in the latter does not get much better. Accuracy or 0/1 error is not effective or valid metric for this problem since it is so much affected by the amount of correctly predicted and widely overrepresented negative class. The ROC/AUC -curve presented in figure 1 shows how similar performance both models have on this task. The ROC/AUC curve plots the relationship between the true positives and the false positives. One thing worth mentioning from the figure is how robust the models are to unseen test data. Random Forest even seems to perform better with test data, which might be due to randomness in the split or coding error from my part.

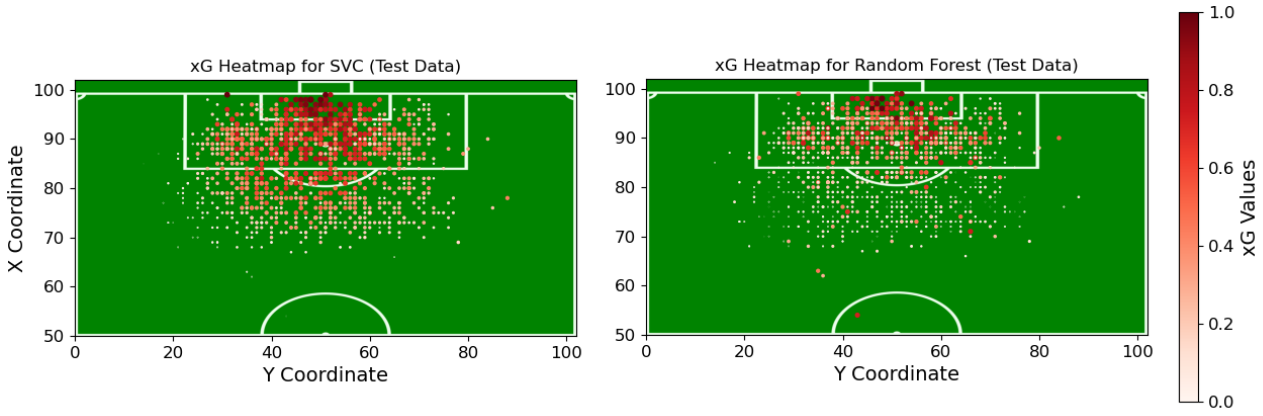

Figure 2. Heatmap that shows the probability of a prediction belonging to positive class (shot leading to a goal i.e., xG value). On the left we have SVC and Random Forest on the right. Both are from test data. Higher values are represented with larger circles and darker red.

From figure 2 we can see how both models are essentially predicting the same principle that probability of scoring a goal is greater the closer the shot is taken. Random Forest does have a smaller radius of high probabilities than SVC which predicts around 0.6 probabilities even outside the box. However, plot shows that Random Forest has more clearly visible outliers that do not really fit the pattern and the probabilities are not as clearly decreasing outwards from the goal compared to SVC. This and the further inspection of the probability distributions (available in the appendices) shows that Random Forest tends to predict much more often near zero probabilities, which explains why it has better accuracy throughout the project. In the case of SVC, the probabilities are much more evenly distributed, and it predicts higher probabilities in average as seen in figure 2.

Conclusion

Although both models do a solid job in terms of accuracy, I cannot consider these results successful. Models have considerable problems in recognizing the positive classes. This makes it so that predicting the positive class correctly without exploding the amount of false positives becomes very challenging. In this kind of task where the classes are heavily imbalanced, the value of true positives is greater than the value of true negatives. If the change in threshold does not change the outcome, then the models do not perform well in general.

Out of the tested two models, I would choose SVC. The models both perform very evenly with the unseen data but SVC does manage to predict positive class slightly better. The visualization also shows that it does not have as many outliers. It yielded a concerning test error of 0.30 with default threshold, but as stated before, the error is not the most prominent metric in this task.

For future applications, I would suggest a larger dataset and some more advanced feature engineering and selection. With larger dataset and more features, the data might turn out to be too complex for model like SVC because of its computational complexity, so I would additionally suggest exploring other models and maybe revisit Random Forest with a larger dataset.

References

[1] L. Pappalardo and E. Massucco, "A public data set of spatio-temporal match events in soccer competitions," 2019. https://www.nature.com/articles/s41597-019-0247-7