Abe Woycke

Final Report:
National Basketball Association (NBA)
Game Outcome Classifier

Problem Statement

I set out to create a classifier that would take inputs of both the away team and home team's statistics entering a game, and predict the game winner. The hypothetical client for this is a company trying to see which team they should contract with to advertise on their jerseys.

Other potential applications of this forecasting are sports gambling, informing a team's internal play style valuations, or adding information to an assessment of the future financial value of an NBA franchise.

When I set out to start the project, I deemed an overall accuracy over 70% would be very successful. Using the eventual model to classify NBA game outcomes against a held out test set resulted in a 66% classification accuracy. This was disappointing, but still a worthwhile result, as 66% accuracy is still 7% better than expert consensus game pick accuracy of 59%.

Data Wrangling

There is an excellent CSV file containing game data from 2004-2020 available at https://www.kaggle.com/nathanlauga/nba-games. It was originally pulled from the NBA.com API.

From that CSV file, I did a lot of data cleaning and transformation. I verified some individual player stats that had been missing to make sure that missing values should be filled with 0. This was accurate.

I wrote several functions to transform the data into the form of observations that would be most useful. I wrote a function that compiles a team's individual games in a season up to the game in question and returns their stats coming into that game. The resultant data frame has rows that are both the home and away team's stats entering the game, along with the game result. This format for the data is nice and tidy for our use case. The predictor variables are all present with the response variable in the same form we would expect to have at time of prediction, and each row is a single observation in that desired format.

From that tidy data set, I did a lot of manual feature transformations based on my domain knowledge. My main goal was to remove redundancies/collinearity in the predictor variables (i.e. transforming 2-pt FGM and 2-pt FGA into 2-pt FG%). I also used a Bayesian prior for each team's win percentage so that early season win percentages (that would normally be at 0% or 100%) wouldn't be outliers. The hope is that this would make for more meaningful predictor features, and that I wouldn't have to limit the range of available classifier models to those that deal with multicollinearity well.

I also combined the home and away team statistics into a dataframe of relative differences to see if this would be more predictive than keeping them separate. These manually extracted features had much stronger correlations with the response variable than the raw stats for each team, so I proceeded with that data frame as the source of our predictor variables.

Exploratory Data Analysis

The project’s goal was to classify NBA game outcomes based off of each team’s statistics entering the game. The main things I looked for in the EDA were:

a) Verifying the transformed data didn’t have any outliers, and was distributed as expected
b) Checking predictor variable correlations with the response variable, home_win

Some visualizations from those portions of the EDA are available on the pages that follow.

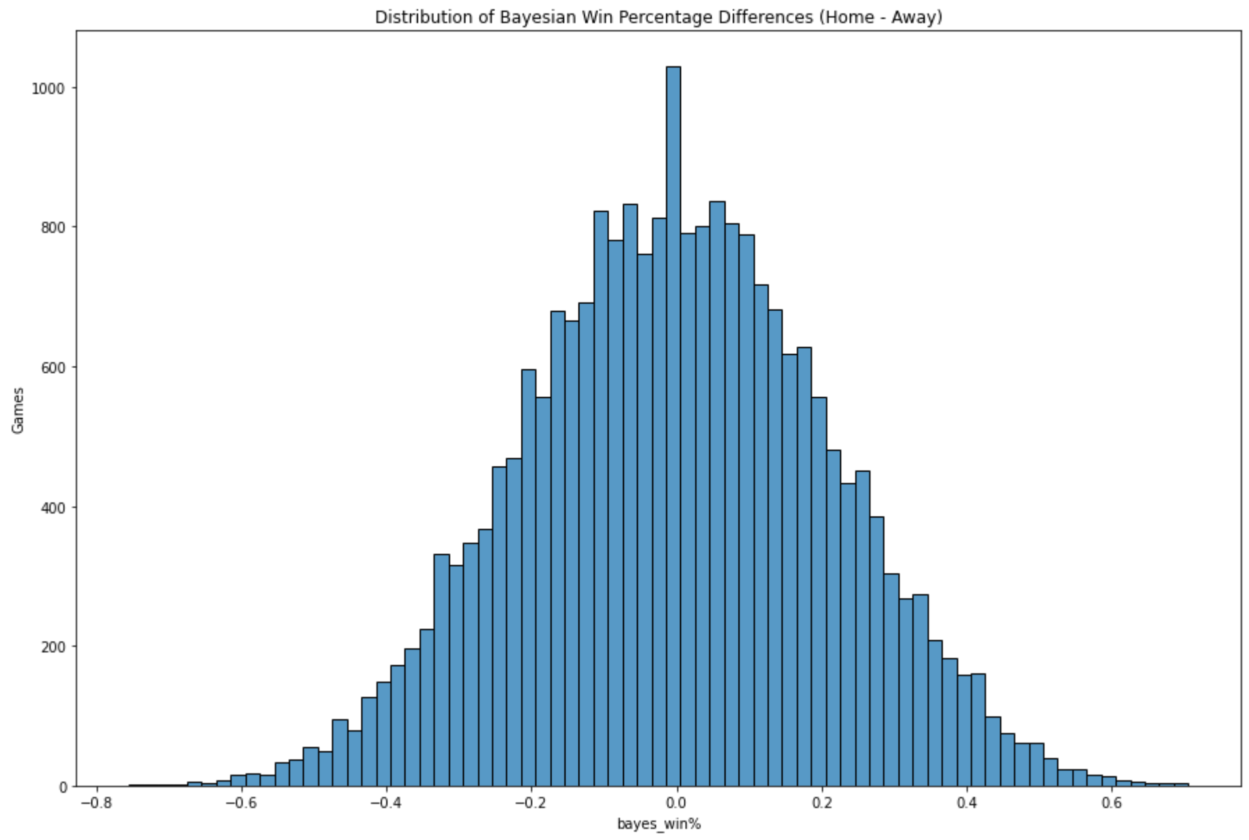


Figure 1: a histogram of Win Percentage Differences

The distribution of our transformed variables all took approximately normal shapes, with averages near zero, and scale for each stat that made sense based on my knowledge of the game.

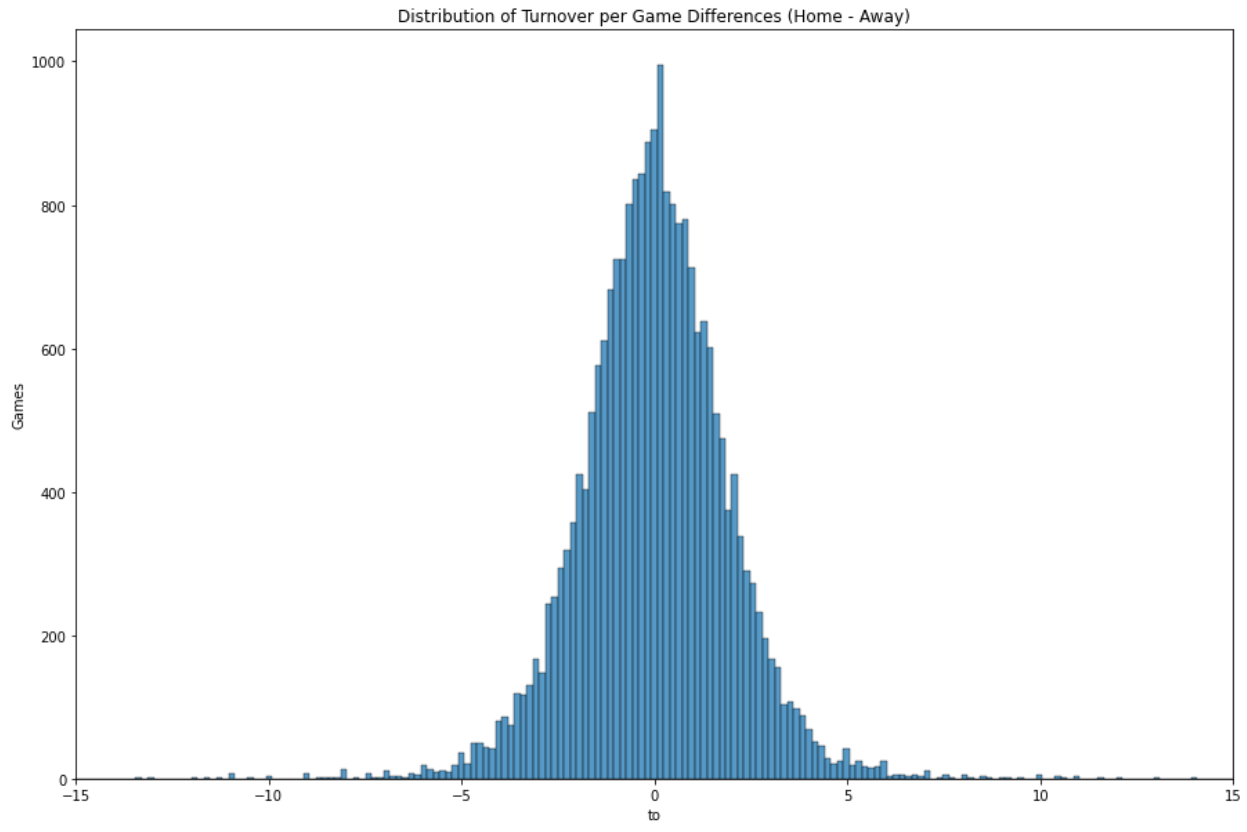
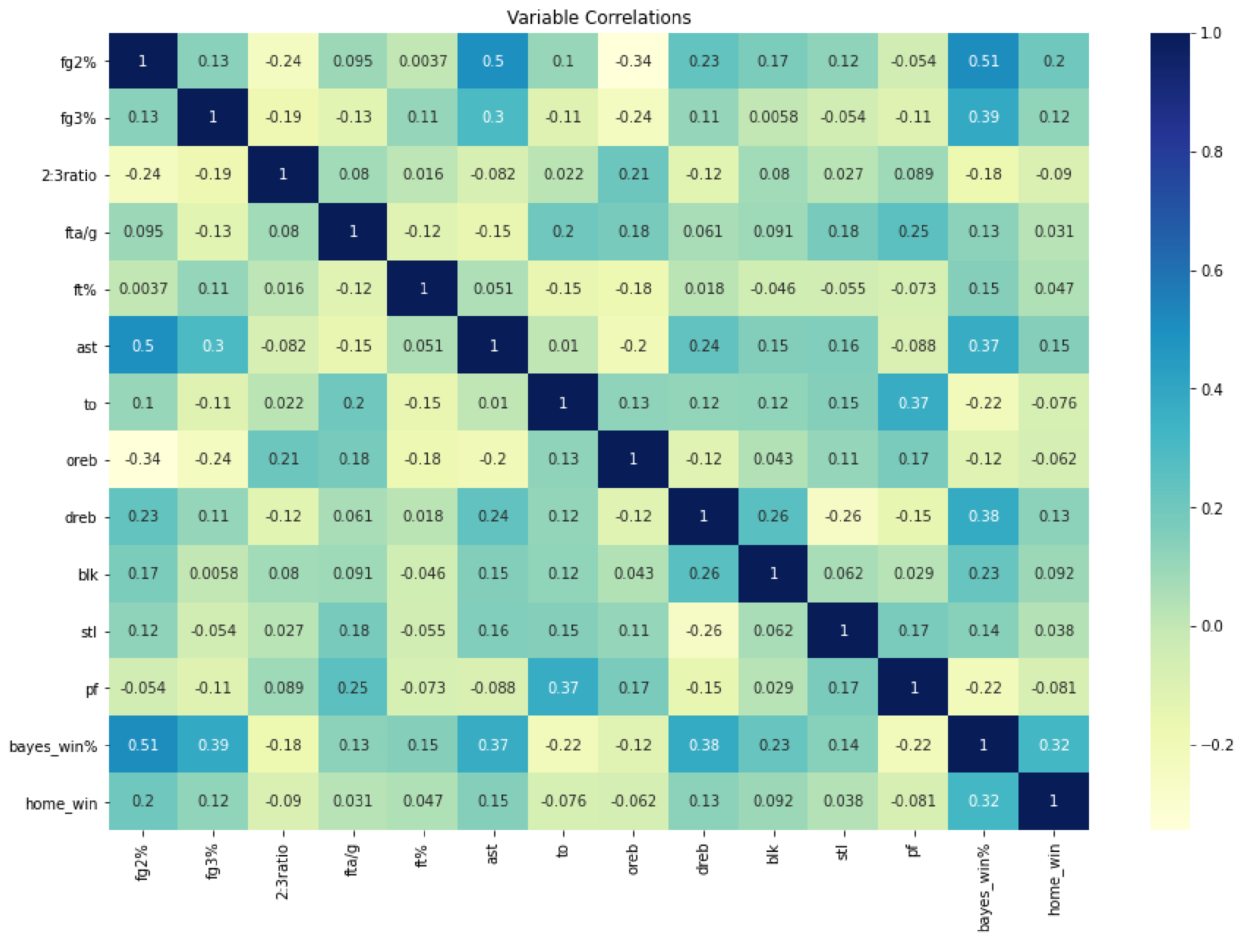


Figure 2: a histogram of Turnovers/Game Differences

Outliers in the dataset are early season games when both team’s statistics have higher variability



	fg2%	fg3%	2:3ratio	fta/g	ft%	ast	to	oreb	dreb	blk	stl	pf	bayes_win%	home_win
fg2%	1	0.13	-0.24	0.095	0.0037	0.5	0.1	-0.34	0.23	0.17	0.12	-0.054	0.51	0.2
fg3%	0.13	1	-0.19	-0.13	0.11	0.3	-0.11	-0.24	0.11	0.0058	-0.054	-0.11	0.39	0.12
2:3ratio	-0.24	-0.19	1	0.08	0.016	-0.082	0.022	0.21	-0.12	0.08	0.027	0.089	-0.18	-0.09
fta/g	0.095	-0.13	0.08	1	-0.12	-0.15	0.2	0.18	0.061	0.091	0.18	0.25	0.13	0.031
ft%	0.0037	0.11	0.016	-0.12	1	0.051	-0.15	-0.18	0.018	-0.046	-0.055	-0.073	0.15	0.047
ast	0.5	0.3	-0.082	-0.15	0.051	1	0.01	-0.2	0.24	0.15	0.16	-0.088	0.37	0.15
to	0.1	-0.11	0.022	0.2	-0.15	0.01	1	0.13	0.12	0.12	0.15	0.37	-0.22	-0.076
oreb	-0.34	-0.24	0.21	0.18	-0.18	-0.2	0.13	1	-0.12	0.043	0.11	0.17	-0.12	-0.062
dreb	0.23	0.11	-0.12	0.061	0.018	0.24	0.12	-0.12	1	0.26	-0.26	-0.15	0.38	0.13
blk	0.17	0.0058	0.08	0.091	-0.046	0.15	0.12	0.043	0.26	1	0.062	0.029	0.23	0.092
stl	0.12	-0.054	0.027	0.18	-0.055	0.16	0.15	0.11	-0.26	0.062	1	0.17	0.14	0.038
pf	-0.054	-0.11	0.089	0.25	-0.073	-0.088	0.37	0.17	-0.15	0.029	0.17	1	-0.22	-0.081
bayes_win%	0.51	0.39	-0.18	0.13	0.15	0.37	-0.22	-0.12	0.38	0.23	0.14	-0.22	1	0.32
home_win	0.2	0.12	-0.09	0.031	0.047	0.15	-0.076	-0.062	0.13	0.092	0.038	-0.081	0.32	1

Figure 3: a heat map of our variable correlations

The variables generally made sense in their direction of correlation with home_win, although the magnitudes did surprise me a bit. The top six variables for correlation with a home team winning were:

1. Winning %
2. 2-point FG%
3. Assists per Game
4. Defensive Rebounds per Game
5. 3-point FG%
6. Blocks per Game

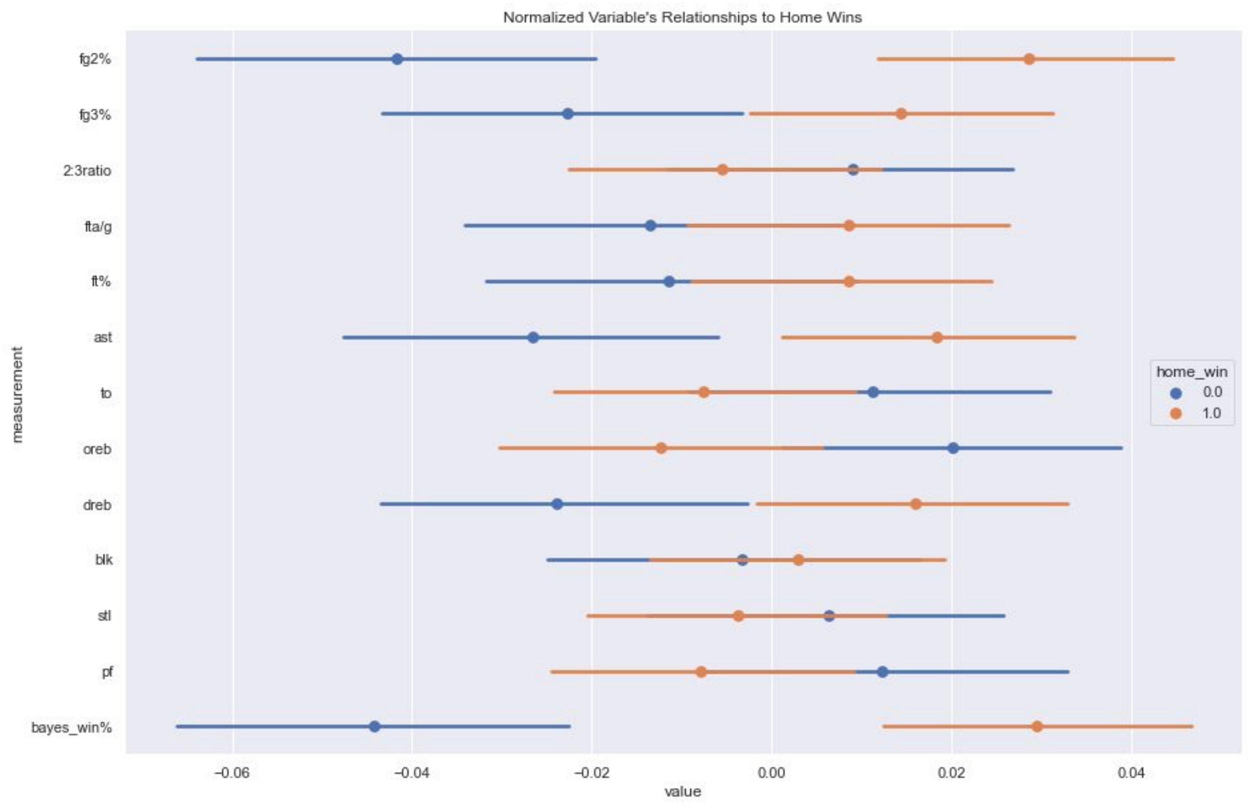


Figure 4: the strength of normalized variable's correlations with the response variable

The above plot also shows the variable relationships. The more the orange line is to the right of the blue line, the more the home team being better in that statistic entering the game impacts their expected win %. These results mostly align with a normal basketball viewer's expectations. For example, if the home team has more assists per game than the away team, the home team is more likely to win the game.

Conversely, if the blue line is to the right of the orange line, the home team having a higher value in this category is negatively correlated with the home team's probability of winning the game. These statistics are things normally thought to be bad for a team. including turnovers and personal fouls.

One variable I was surprised by was teams having higher offensive rebounds are more likely to *lose*. This goes against my intuition. But perhaps you only have more chances for offensive rebounds if you miss shots!

Preprocessing and Model Selection

In terms of preprocessing, I used scikit-learn's TrainTestSplit to hold out 20% of our dataset for testing, to account for potential overfitting to our training set. As a matter of curiosity, I tried a lot of the out-of-the-box versions of different classifiers, including most of the ensemble classifiers in scikit-learn, as well as XGBoost. I also tried some neural network classifiers. Those initial exploratory results narrowed my search to building and tuning three models: a GradientBoostingClassifier, a Logistic Regression, and a Neural Network Classifier.

Before proceeding with tuning, I selected area under the receiver operator curve (AUROC) as a suitable performance metric for model comparison. In cases with imbalanced classes, AUROC can be a more valuable metric than pure accuracy. Additionally, in this use case, I don't place a particularly higher or lower value on either Type 1 or Type 2 errors, so focusing on Precision/Recall is less necessary.

Because it is computationally intensive, I tuned the GradientBoostingClassifier using 1,000 iterations of RandomSearch over a wide parameter space. Out of curiosity, I also used HyperOpt, and then did a narrowed GridSearch on the set of parameters in and around the successful results from the prior two searches. The final narrowed GridSearch performed the best of the three GradientBoostingClassifiers, with an AUROC of 0.698 when predicting on the hold out set. I don't necessarily attribute this success to the GridSearch, as the differences between the three GBCs performance in AUROC were small enough to be the result of chance with regards to the TrainTestSplit.

Because the Logistic Regression is not computationally intensive, I used a wide GridSearch for tuning. The resultant model had an AUROC of 0.696 when predicting the test set. For the neural network classifier's tuning parameters, I used a GridSearch. The resulting NN had an AUROC of 0.691 when predicting the test set. I also created some voting classifiers to combine the other three models' predictions, and they performed comparably to the individual models.

For the purpose of most accurately predicting game outcomes, I would probably opt to use the GradientBoostingClassifier. The soft voting classifier might also be worth considering.

Takeaways

The best performing of the three individual models by AUROC is the GradientBoostingClassifier, with an AUROC of 0.698. The best performing of the three individual models by classification accuracy is also the GradientBoostingClassifier, with an accuracy of 0.661.

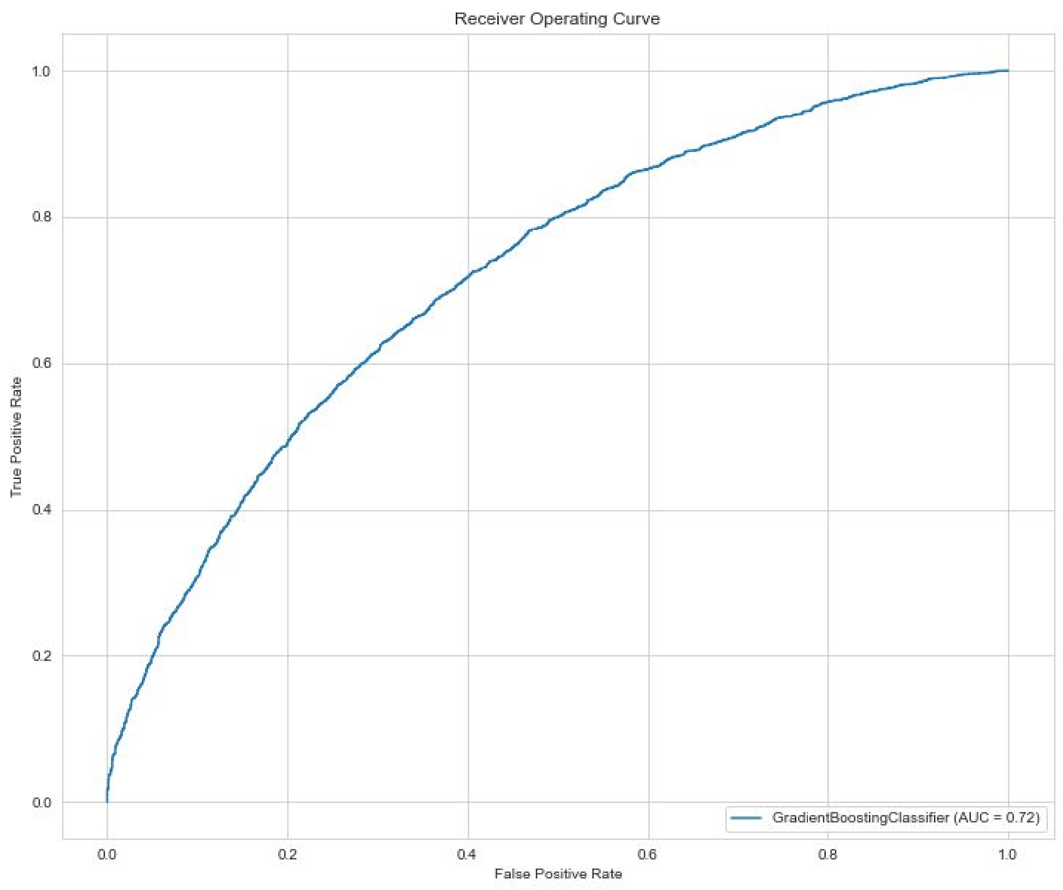


Figure 5: the final model’s Receiver Operating Curve
(a different test set split was used for final viz, thus the different AUC score)

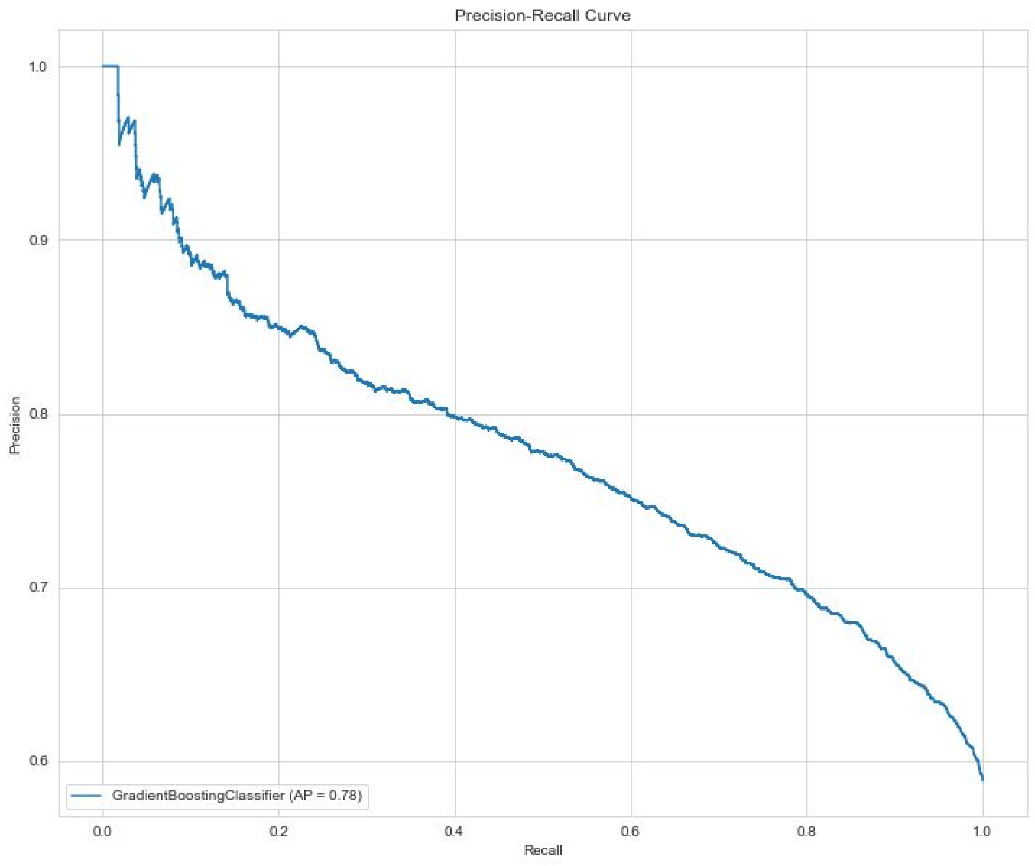


Figure 6: the final model's Precision-Recall Curve

In this case, I don't place a different value on Type 1 or Type 2 errors, so don't have a strong opinion on maximizing precision or recall.

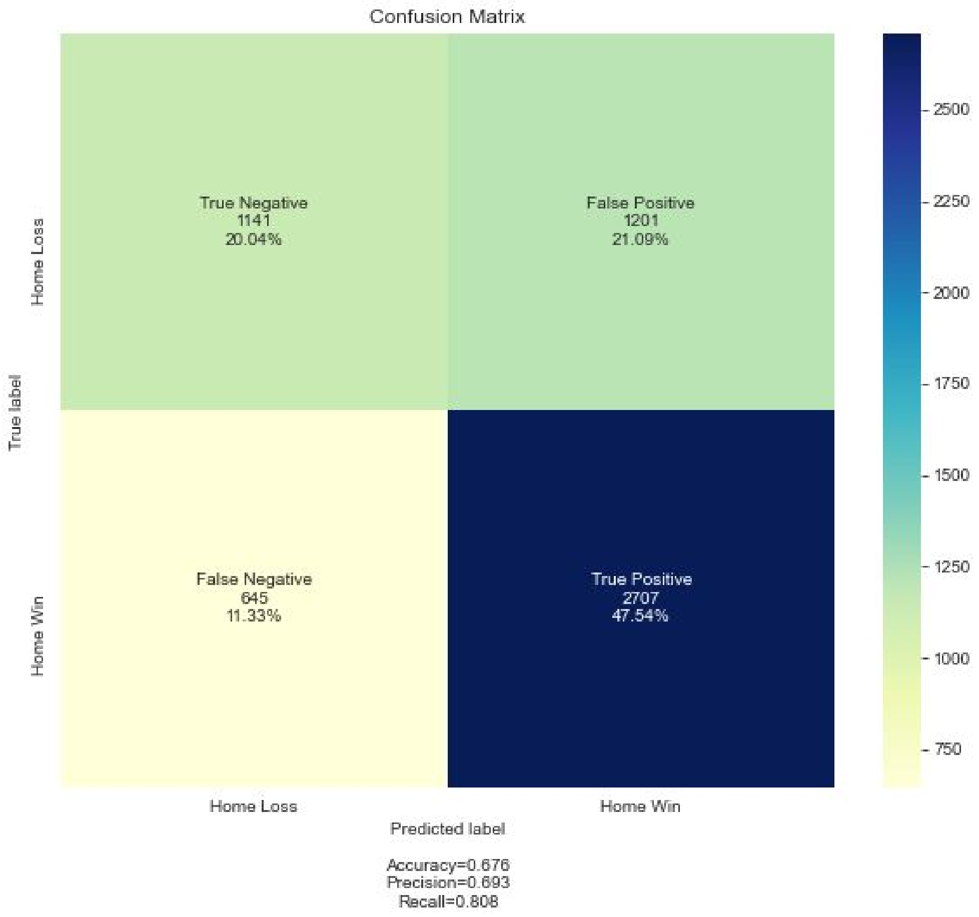


Figure 7: the final model's Classification Matrix

I would be interested in seeing if there are tweaks that could be made that would decrease the false positive rate. I attribute the high false positives to class imbalance (home teams win 59% of the time).

All of these differences in model performance metrics are very small, and are also partially influenced by the random chance from the selection of our sample. Even more hyperparameter tuning could perhaps find an incrementally better model.

Future Research

If the data is available, I would be interested in expanding the dataset and looking at data before 2004. I would also be interested in building a pipeline to simulate full seasons using the dataset. Also, I would be interested in seeing how the model performs against truly unseen data with pre-2004 backtesting, and how it will perform in the upcoming 2020-2021 season (although that might be a weird sample due to limited home fan attendance).

Credits

Scikit-learn: Machine Learning in Python, Pedregosa et al., JMLR 12, pp. 2825-2830, 2011.