Basketball Prediction: Milestone Report 2

Introduction

Every year, many people attempt to predict the outcome of the NCAA basketball tournament. My goal is to use machine learning to predict the outcomes of the tournament games based on data from the current season games as well as data from past years. I will consider every possible matchup of teams, and predict the winning team based on the data. I will consider many different variables, such as field goal rate, free throw rate, turnovers, and rebound rates to determine what factors affect the probability of a team winning.

Many basketball enthusiasts and sports analysts are interested in this data, and how to predict the winners of the tournament. Knowing which factors are strongly correlated to winning would be very useful. Even a basketball fan who has very little understanding of data analysis could make better predictions if they knew which factors have the greatest impact on a team's likelihood of winning.

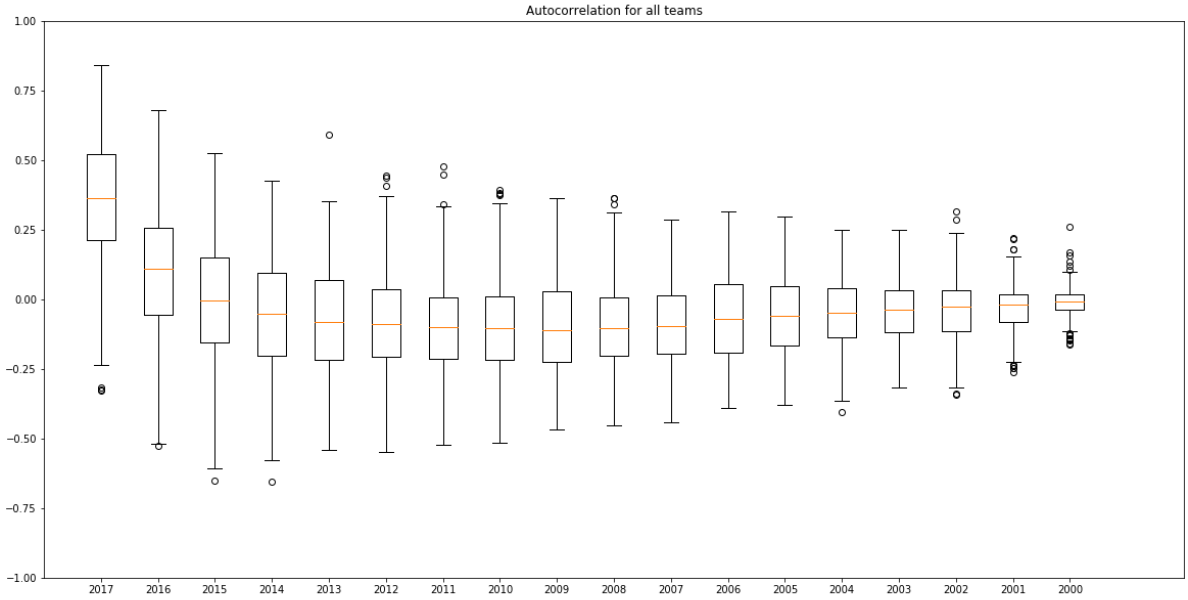
If possible, I would like to use a Bayesian approach, calculating the conditional probabilities of either team winning a game, given data from previous games. This would be different from the traditional approach, where the winner of each game is predicted based only on the regular season data, and data from previous tournaments.

Data Collection and Cleaning

Most of the data was obtained from Kaggle, and is available in multiple csv files. Some of the features that I would like to use are the field goal rates, free-throw rates, etc. The raw data does not contain these rates, but they can be easily calculated. For example, to find the field goal rate for a team, I can divide total field goals made by total field goals attempted. I calculated these rates and constructed a dataframe containing three columns: season, team, and field goal rate. I noticed that there were some null values in the field goal rate. I did some investigation, and discovered that these teams were not Division I teams in those years.

I was able to find additional data from sports-reference.com. This website has the overall ratings for the teams, which was not available in the other datasets. The data from sports-reference.com was in html tables, so I had to do some web scraping to access the data in a format that I could use for analysis. I exported the data for each year in a csv file, extracted the columns that I needed, then merged it into a single dataframe. I had to do some cleaning to remove rows with null values, and also some rows that contained additional information that was not needed. I then calculated the correlation between the 2018 ratings and the ratings for each previous year, going back to 2008. There was still a fairly strong correlation between the 2018 ratings and the 2008 ratings, so I included the data for a few more years, going back to 2000. For the earlier years, especially prior to 2003, the correlation was not as strong.

I then looked at the autocorrelation of individual teams' rankings. I used the autocorrelation function from statsmodels to do this, and used matplotlib to construct boxplots displaying this data. First, I constructed a dataframe where each column was a year, starting at 2018 and going back to 2000, and each row was the overall rankings for a team. I used several for loops to access the data I needed, first calculating the autocorrelation values for each team, then accessing all the values for 2018, all values for 2017, and so on. I constructed a boxplot for each year to get an overall picture of the autocorrelation. The plots are shown below. For the year 2018, all the values were equal to one, because it is the correlation of that year's ranking with itself, so I did not include the 2018 correlation values.



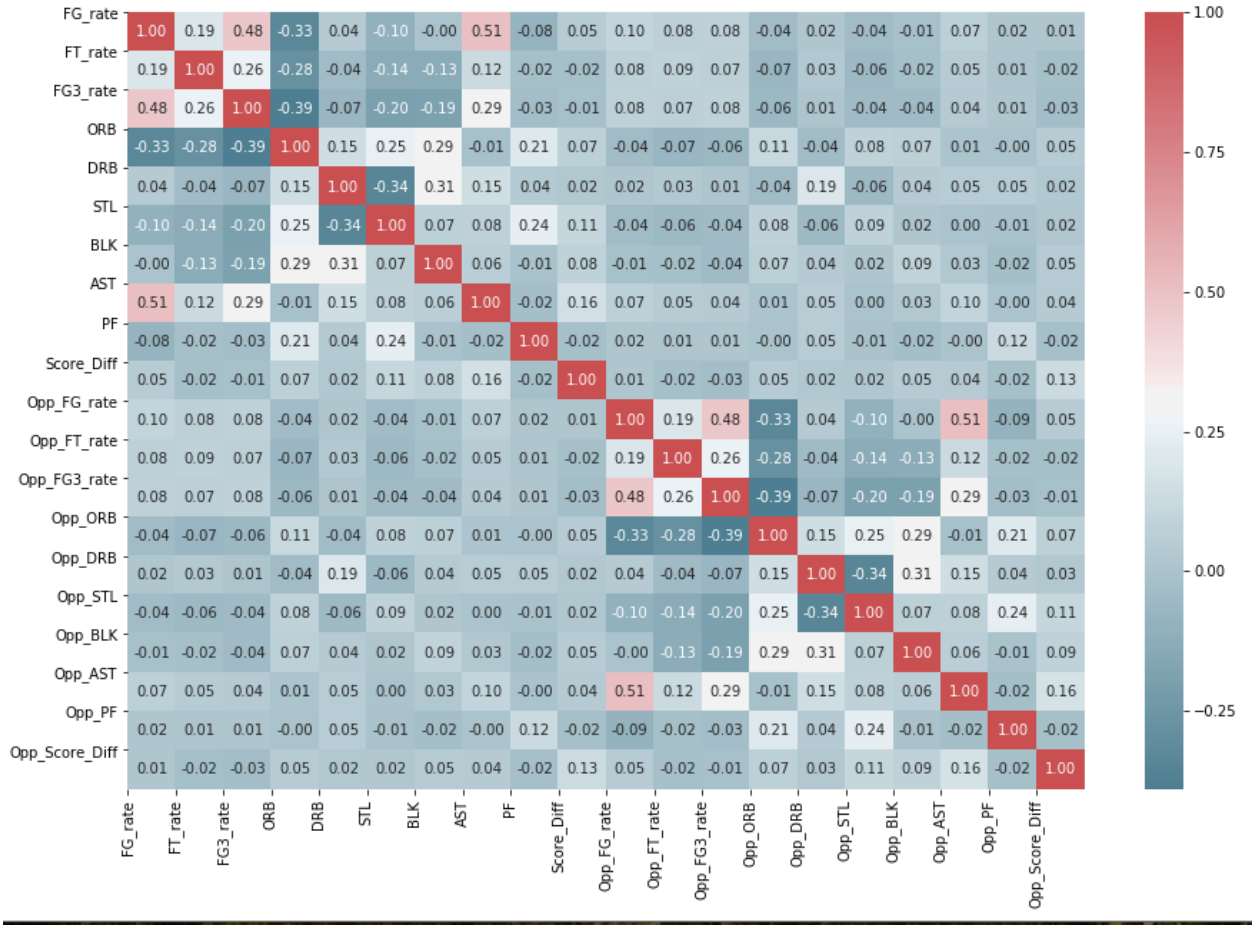
Before I can build a model to predict the outcome of a game, I need to determine which variables should be used in the model. The data from Kaggle contained the season and day of the game, and lots of raw data for both teams, including points scored, number of assists, personal fouls, free throws attempted, free throws made, steals, and many others. Rather than use these values directly, I wanted to create features using this data. For example, I calculated the field goal rate for each team by taking the number of field goals made divided by number of field goals attempted. Similarly, I calculated the free-throw rate and three-point shot rate. I also calculated the average number of assists, steals, blocks, rebounds, and personal fouls per game for each season, along with the average difference in scores. I constructed a column of each of these values.

In the dataframe I had constructed, each row represented a game, and contained data for both the winning and losing team. To begin the process of feature selection, I wanted a dataframe where each row contains data for one team and an indicator of whether or not they won the game. This would allow me to determine which features were related to a team winning. I used the Pandas stack() function to combine the winning team ID and the losing team

ID into a single column. Then I add a column of ones and zeros, indicating whether the team won or lost. I then used the Pandas merge function to combine this data with all the features I created earlier. Now I had a dataframe where each row contained the team ID, the opposing team ID, and the data for both teams. The field goal rate, number of steals, etc. was always for the previous season. So to predict the outcome of a game in 2018, I will use the rates from 2017, and so on. I exported this data as a csv file so that it could be used for further analysis.

Exploratory Data Analysis

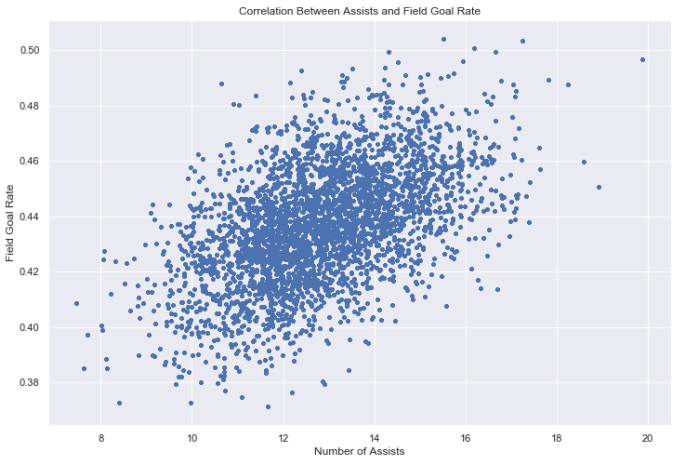
To conduct further analysis, I read in the file containing all the data, and selected all the quantitative features. I wanted to see whether any of these features were highly correlated with each other. The correlation heatmap, shown below, displays the correlations between all these features. More red color indicates a stronger correlation, and light blue indicates little or no correlation. This plot also displays the correlation coefficient for each pair of features.

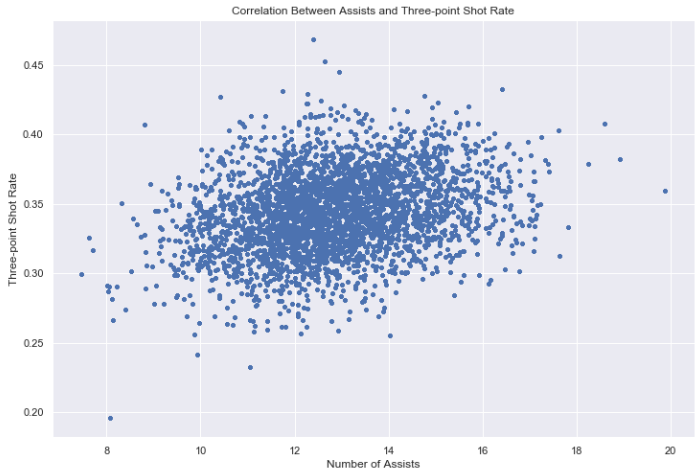


The plot indicates that most of the correlations are very weak. The only variables that appear to have a moderate positive correlation are field goal rate vs. three-point rate, and field goal rate vs. assists.

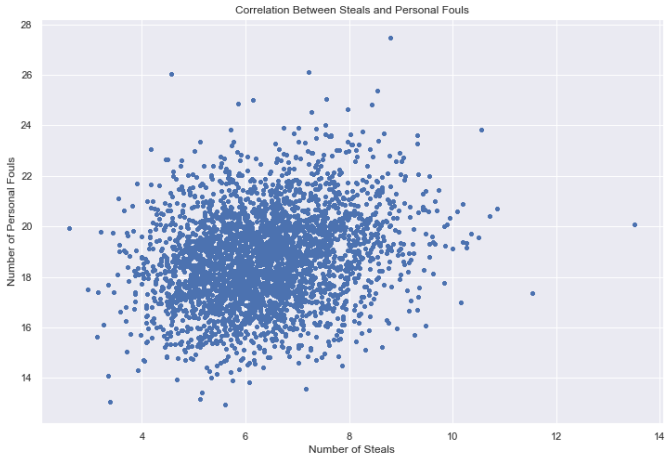
After merging all the data and adding the features, I noticed that there were some null values in the dataframe. Upon further inspection, I noticed that there were several rows where the team ID was missing. I dropped these rows from the dataframe, since they would not be useful for prediction. There were also some rows where some of the feature columns had null values. I filled these using the column mean. I now had a dataframe with just over 100,000 rows and 25 columns. There were a few more features that I wanted to add, using external data that I had saved as csv files. First, I used data from the NCAA tournaments to add a column indicating whether or not each team had gone to the tournament in the previous season. Then I used the team rankings data (from sports-reference.com) to add a column with each team's ranking from the previous year.

I wanted to look more at the correlations between some of the variables, so I constructed a few scatterplots. Two of these plots are shown below. There is a clear positive correlation between number of assists per game and the field goal rate. So in general, teams that have more assists per game tend to have a higher field goal rate. There is also a positive correlation between number of assists and three-point shot rate, but it is not as strong.

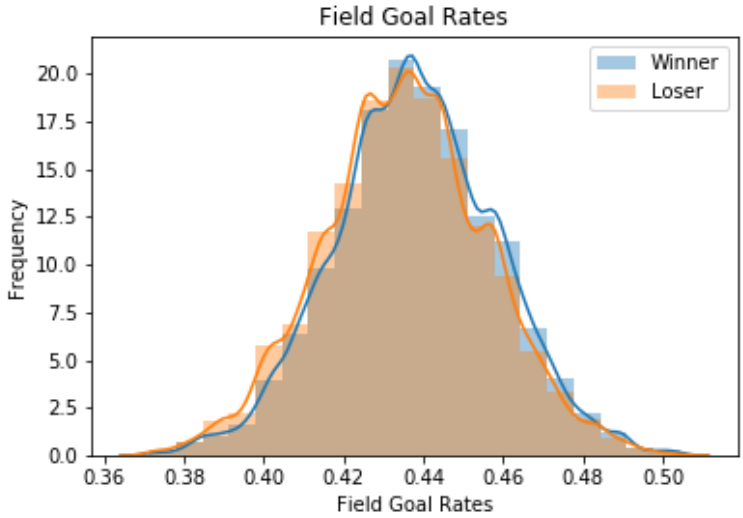




On the correlation heatmap, there also appeared to be a slight positive correlation between number of steals and number of personal fouls. The plot below shows this correlation. Although the previous plots did not surprise me, this correlation is not one that I would have expected. However, it does make sense that there could be a correlation here. If a team is playing more aggressively, they will tend to steal the ball more, and they will also tend to have more personal fouls.



To determine which features might be good predictors of winning or losing, I constructed a few plots comparing the winner's data to the loser's data. The plot below shows the distribution of field goal rates for winning teams versus losing teams. We can see that the field goal rates are approximately normally distributed for both teams, but the winner's plot is shifted slightly to the right. This indicates that there is a difference between field goal rates for winning teams and field goal rates for losing teams, so this may be a good predictor to use in the model.



I then conducted hypothesis tests using all the features, to determine whether there is a difference between winners and losers. For most of the features, the p-value was very small, which indicates that there is a difference in that particular feature between the winning team and the losing team.

The next step was to fit a model to predict whether a team will win or lose. The dataset had some null values for the rankings, since some of this data was not available. It did not make sense to impute this missing data, so I decided to use lightgbm, a machine learning library that can handle null values. I split the data into training and test sets, and fit a model using the training data. I then used the model to predict values using the test set, and compared the predicted values to the observed values. The accuracy was about 62%. I decided to add data from 2 years prior to try to improve the model. After adding these features, the accuracy improved to just over 63%. I then added the Season, the team ID, and the opposing team ID as categorical variables, which improved the accuracy to about 69%.

Conclusion

In this project, we have discovered which features can be used to predict the outcome of a basketball game. By looking at the autocorrelation, and determining which features are correlated with each other, we have selected the best predictors and built a model that predicts the outcome of a game with almost 70% accuracy. The next step will be to tune the hyperparameters, and possibly improve the performance of the model a little more. I will also look at the feature importance, and determine whether there are features that are not needed, and can be dropped from the model.