MGT 6203 | Spring 2023 |Team 20
Mina Weldon, William Warner
Heather Connelly, Alexander Reiners

Predicting the Outcome of College Basketball Games

Overview of Project

Introduction

A basketball upset is defined as a game in which the team popularly predicted to win (the "favorite") loses to a team the majority expects to lose (the “underdog”), defying expectations. What or when upsets will occur is an age-old question, despite basketball pundits and betting analysts constantly making their own predictions. College basketball upsets are often unpredictable, and can have financial, reputational, and even emotional impact. Team, player, and game attributes, and play types that drive upsets need to be identified.

Business Justification

“Upsets make the college basketball world go round,” 247sports asserts.[1] Fans, from current college students to lifelong fans, have a huge emotional investment in their team’s basketball season performance. Come every March, millions of fans fill out a bracket, trying their hand at predicting upsets and champions (even though according to the NCAA, the odds of filling out a perfect bracket are 1 in 9,223,372,036,854,775,808!)[2]. Gamblers have a clear financial interest in games. Nearly $10 billion was gambled on March Madness in 2021, with that number expected to grow in successive tournaments (American Gaming Association (AGA)[3]). TV executives also have a financial interest in ensuring the most exciting games, many of which are upsets, are shown at high viewership times and could be used to help sell advertising at a higher rate for exciting games. Furthermore, March Madness (the NCAA Division I men’s basketball tournament) is the NCAA’s largest money-maker. The tournament typically brings in over $1 billion in revenue, and a portion of this is distributed back to teams based on their tournament performance. The further their teams advance in the tournament, the more a school receives from the NCAA.

Determining the most important predictors of an upset win has the potential to make fans, gamblers, TV executives, and college athletic programs alike a lot of money.

Problem Statement

The goal of this project was to identify the attributes that most accurately predict a basketball upset by analyzing college basketball game data over a 5-year period.

Hypothesis

Through our analysis we expected to identify key attributes in predicting when a college basketball upset will occur, whether that be due to attributes directly related to the teams’ scoring or due to less scoring-related factors such as flagrant fouls or turnovers. We expected that rather than a single attribute or two having overweighted significance in predicting an upset, that several factors interacting would most accurately predict an upset.

Methodology

Data for this project was compiled by scraping from two data sources ranging from 2014-2019. We chose this timeline to avoid the skewed data due to COVID disruption of the basketball seasons. The dataset was merged based on team name and was focused on D1 teams due to limited data from KenPom. Given that each data source provided numerous attributes, the project team visually selected appropriate attributes from each data source for the analysis. The project team explored variable selection using forward, backwards, and both forward and backward stepwise regression. Based on the variables selected, the team explored various propensity models considering logistic regression, random forest, and XGBoost. The output of these models were probabilities [0,1] of a certain game being an upset

game, and therefore the team had to calculate and set a threshold between 0 and 1 that would optimize desired performance of the propensity model. Assumptions were made on the desired performance of the propensity model (i.e., higher recall and precision are more important than high accuracy/low error, false negatives are more costly than false positives, etc.).

Overview of Data

Data Sources

The project team leveraged two data sources, KenPom [4] and ESPN [5], for this analysis. KenPom provides NCAA basketball statistics starting in 2002, with over 100 attributes ranging from efficiency statistics, offense and defense statistics, team ratings, home court ratings, to player height and experience. The ESPN dataset provides over 50 play-by-play attributes including score and time at the time of the play. After we removed columns that were not relevant to our project, the cleaned and merged dataset had 63 attributes with a portion of the dataset shown in the figure below.

	X	game_id	home_team	home_season	home_team_abbreviation	home_field_goal_pct	home_three_point_field_goal_pct	home_free_throw_pct
1	4339	400596493	Hartford	2015	HART	38.5	15.6	60.0
2	6059	400817367	Bryant	2016	BRY	34.3	23.1	66.7
3	18856	400989727	Northern Iowa	2018	UNI	50.0	45.5	65.2
4	6707	400827784	Georgia Tech	2016	GT	40.7	29.4	86.4
5	12785	400916461	Southern Utah	2017	SUU	39.3	34.5	80.8

Figure 1: Merged Dataset

Data Cleaning and Challenges

The biggest challenge in the project was scraping, cleaning, and merging the datasets. A readily available python package was used to scrape KenPom website for its data and the hoopR R package was used to scrape the play-by-play data. However, troubleshooting the codes in the packages was challenging. We learned that the developer had made updates and changes, but those changes were not reflected in the packages we were referencing. By tracking down the latest updates and the updated packages, we were able to successfully scrape the data from our sources without much time lost.

As for data cleaning and merging the data sets, the ESPN data stored short code for each college basketball team while the KenPom data stored the full college name making it difficult to merge by college team names. The short code in the ESPN data had to be translated into appropriate college names to map the data across the two sources to merge into one final dataset. General rules, such as replacing “St.” or “St” with “State”, was helpful in narrowing down the number of team names we had to manually change.

The rest of the data cleaning was smooth. There was no missing data that we had to impute, but we found that home_team_rebounds was populated with all 0 values, so we removed this column in addition to the other irrelevant columns. We also split our dataset into train (80%) and test (20%) sets, which allowed us to calculate model performance as we developed the model and compared different model types and parameters.

Exploratory Data Analysis (EDA)

EDA revealed a highly imbalanced dataset, with about 1.5% of all games being classified as an upset. This helped to set expectations as we began model development and made us aware that using a simple accuracy or error metric would not be sufficient.

We also explored the relationships between the input variables so that as we began the model development process, determining which variables should not be included in logistic regression models together. For example, a few predictors are included in the example correlation plot below. As seen in

Figure 2, besides the strong positive relationship between home_total_rebounds and home_defensive_rebounds or home_offensive_rebounds (which is intuitive), the plot shows that there are not many high correlations between these predictors.

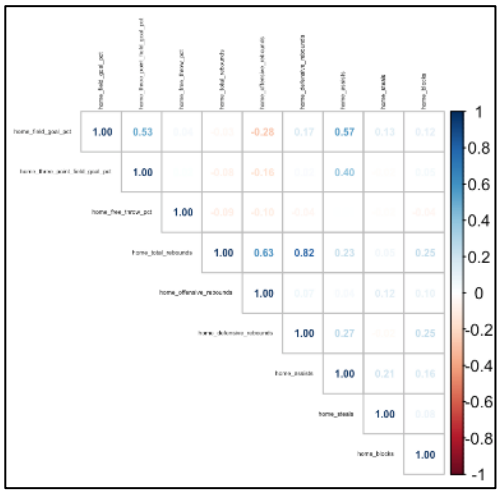

Figure 2. Predictor Variable Correlation Plot

Key Features

Some of the most important features in our final model were the number of points scored in a game by both the home and away teams, two-point field goal percentages for both the home and away teams, and the away team's count of steals. As hypothesized, there were many "key" features predicting an upset of a game. More detail on the key features is covered in the XGBoost Model Development section, including a detailed figure showing the ranked feature importance (Figure 16).

The team did not pursue feature engineering, as most of the features were clean and already primed for model development. Engineering historical performance features could be an interesting next step (e.g., does a team's count of steals during their last game have any impact on whether their current game will be an upset? Or does summing the number of a team's upsets over the past couple years help predict upsets?).

Overview of Modeling

Variable Selection

The team used stepwise regression (backward elimination, forward selection, and bidirectional elimination) as well as the LASSO method for variable selection. Since we wanted to estimate the probability of an upset occurring, we used logistic regression within all of these variable selection methods. As seen below, the stepwise regression using both directions had the lowest cross validated mean squared error (MSE).

model	CV MSE
backward	0.01473
forward	0.01474
step	0.01472
lasso	0.02952

Figure 3: Cross Validated Mean Squared Error per Variable Selection Method

This method selected 23 variables/features, which was very helpful in narrowing down from the original 51 independent variables. This allowed for faster computation in the next model development steps.

Logistic Regression

We explored logistic regression to predict the probability of a basketball game being an upset. Because the output of the logistic regression model is a decimal and not necessarily 0 or 1, we created a threshold to define what we consider an upset. We found that a threshold of 0.15 was optimal if we optimized our total cost function, which defined false negatives as twice as costly as false positives. This threshold gave us an alarmingly high accuracy rate of 98.3%, and a low recall rate of 44.4%, as seen in the confusion matrix below.

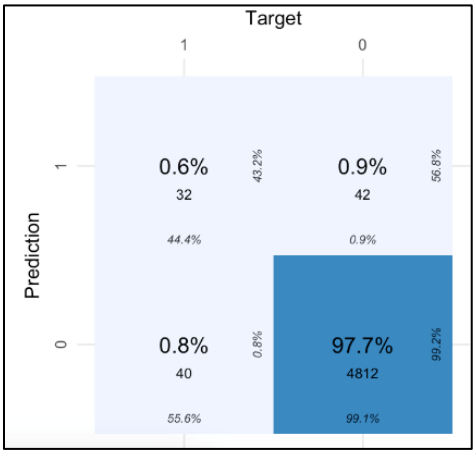

Figure 4: Initial Logistic Regression Confusion Matrix

During the model development process, we discovered that we were including a feature, game_spread, which was originally used to create our target variable, upset. Game spread was highly correlated to our target, which was intuitive because the game spread had to be high (absolute value > 10) in order for a game to be considered an upset. This feature was removed from our model development going forward and observed the model performance decline.

Next steps included refining our threshold. Using a default threshold of 0.5 (model output probabilities greater than 0.5 would be considered an upset) resulted in the model predicting no upsets. To address this issue and to find the optimal threshold, we tested different thresholds and tracked several performance metrics (cost, accuracy, recall, precision, and f1 score) at each threshold value. We again defined our cost function by making false negatives as twice as costly as false positives, which resulted in the below total cost plot.

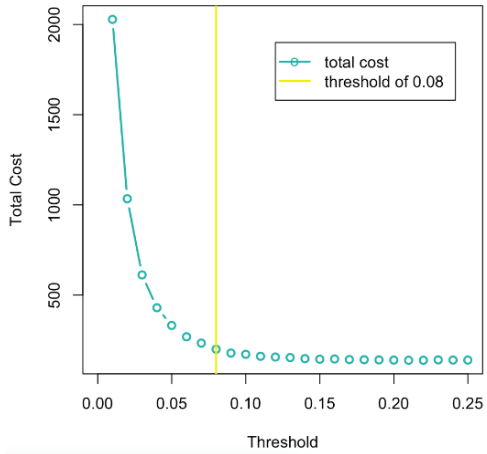

Figure 5: Logistic Regression Total Cost per Threshold

The plot was not particularly useful since if minimum cost was desired the threshold near 0, which would predict almost all games to be an upset. Note that our final chosen threshold for logistic regression occurs at the elbow of the graph of 0.08, which is not the minimum cost but still acceptable.

Because over 98% of the dataset was the negative class (not an upset), using accuracy to determine the threshold would be inadequate as any highly imbalanced dataset could easily yield a high accuracy. The F1 score would be an optimal metric for our use case as it is known to work well for imbalanced classification problems. We liked how it incorporated both recall and precision, which both focus on the positive class, upsets. Recall measures how well true upsets were predicted, while precision measures the accuracy of upset predictions. As seen in the figure below, we decided to set our final logistic regression threshold at the threshold of the maximum f1 score: 0.08.

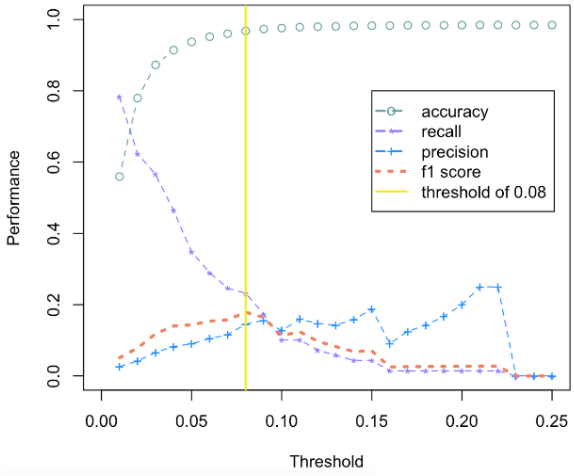

Figure 6: Logistic Regression Performance per Threshold

The final logistic regression confusion matrix and Receiver Operating Characteristic (ROC) curve can be seen below. Although we were still at a low recall rate of 23% and low precision rate of 15%, the results were acceptable due to the imbalanced class issue.

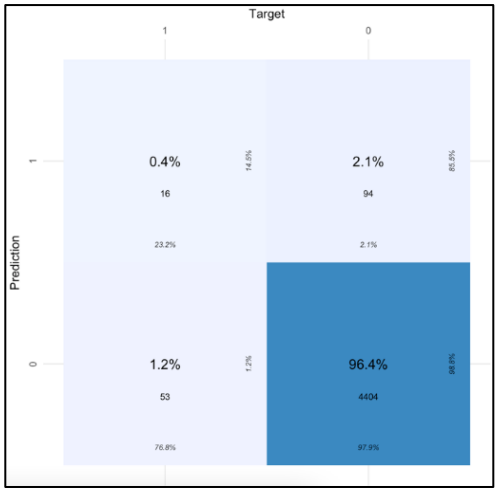

Figure 7: Final Logistic Regression Confusion Matrix

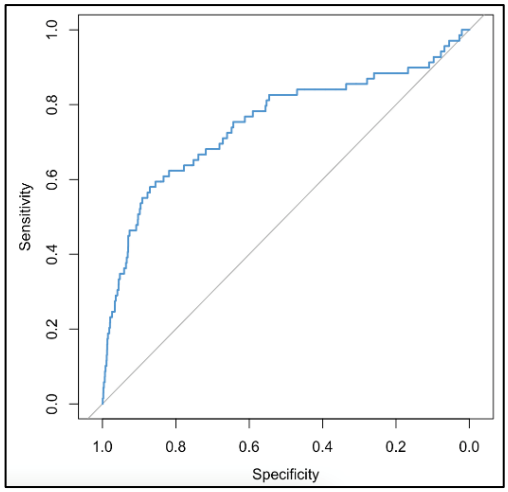

Figure 8: Logistic Regression ROC Curve

Random Forest

For the random forest approach, the R package randomForest was used. To set up the model, the

response variable 'upset' was set to type 'factor' to trigger the classification type of random forest and then identical columns and columns not relevant to the modeling, such as away_team_abbreviation, were dropped from the data. The first model was run using the default number of trees and a 'mtry' value, which is the number of variables randomly sampled as candidates at each tree split, being based on the square root of the number of columns. All variables in the dataset that could potentially be relevant were used. After running the code, it was seen that the random forest model predicted that no games would end in an upset, as shown below in **Figure 9**.

```
        OOB estimate of  error rate: 1.54%
Confusion matrix:
      0 1 class.error
0 22484 0           0
1   352 0           1
```

Figure 9: Random Forest Confusion Matrix

After considering how unlikely upsets are in terms of probability, and since there was a large imbalance between the number of '1' and '0' responses in the dataset, 352 vs. 22,484, the random forest classification modelling was moved to a more probabilistic model by using the predict function with *type = 'prob'* to get the relative probabilities of each binary response. This way the classification threshold could be altered between 0 and 1 to try and improve the resultant confusion matrix of the random forest model. This resulted in the confusion matrix seen below when using a 0.10 threshold value.

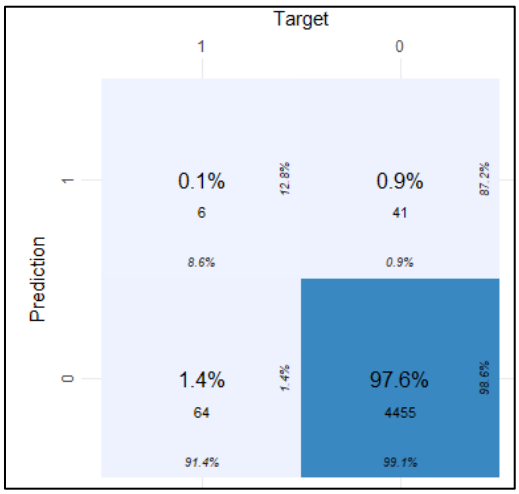

Figure 10: Confusion Matrix with 0.10 Threshold Value

As seen in the logistic regression model, a tradeoff exists between the accuracy and recall of the model. If a higher threshold value was used, the accuracy increased but the ability of the model to correctly predict an upset decreased.

A random forest model using as predictors the variables found during the variable selection modeling was examined, however no improvement was seen in the overall predictive ability of the model and a similar tradeoff was seen when adjusting the threshold value in terms of accuracy and recall. In an attempt to improve the predictive power of the random forest model, the 'mtry' input value was tuned, however no improvement was seen compared to the method of using the square root of the number of data columns to determine this value. Other random forest models investigated included only using the KenPom variables, only using ESPN variables, and modelling variable interaction, but with little to no improvement to the overall predictive power of the model.

The last piece of information pulled from the random forest model was the variable importance, which shows which variables had the highest predictive power. The mean decrease accuracy plot shows how much the accuracy of the model decreases by excluding each variable, and the mean decrease in gini plot essentially shows how useful a variable is in gaining homogeneity of the tree nodes when splitting the tree on the variable. The figure below shows the variable importance for the original random forest model of using all variables.

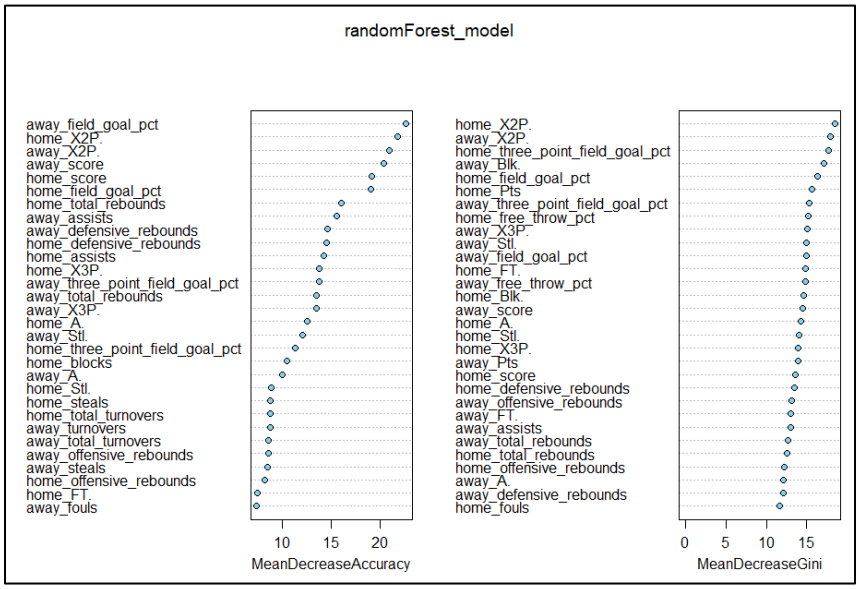

Figure 11: Variable Importance Plot

XGBoost

The last model development approach was XGBoost, an implementation of the gradient boosting decision tree algorithm. This approach took a few extra steps to get started, as XGBoost requires data to be in a matrix, and there were numerous input arguments to investigate (e.g., max.depth of trees, nround (maximum number of iterations), early_stopping_rounds (used if improvement not observed), scale_pos_weight (to control for imbalanced classes), and gamma (a regularization term)). We started with the default value of the many input variables and a threshold of 0.5, which resulted in poor results for the upset class as seen in the figure below.

Figure 12: Initial XGBoost Confusion Matrix

The team then used R package mlr (Machine Learning in R) to tune the hyperparameters. We used a "random search" optimization strategy to find the best combination of our parameters, which was much more efficient than looping through all possible hyperparameter values. We chose to move forward with the hyperparameters that yielded the highest value for our evaluation metric: balanced accuracy. It was not feasible to use all evaluation metrics in some of our tuning functions, but balanced accuracy seemed to be appropriate. Similar to the f1 metric, balanced accuracy is helpful for imbalanced classes because it weights the accuracy from the positive class the same as the accuracy from the negative class, no matter the volumes.

The XGBoost model was trained with the optimized hyperparameters and we chose a few key metrics to output at every threshold value: recall (true positive rate), precision (positive predictive value), and f1 (harmonic mean of recall and precision).

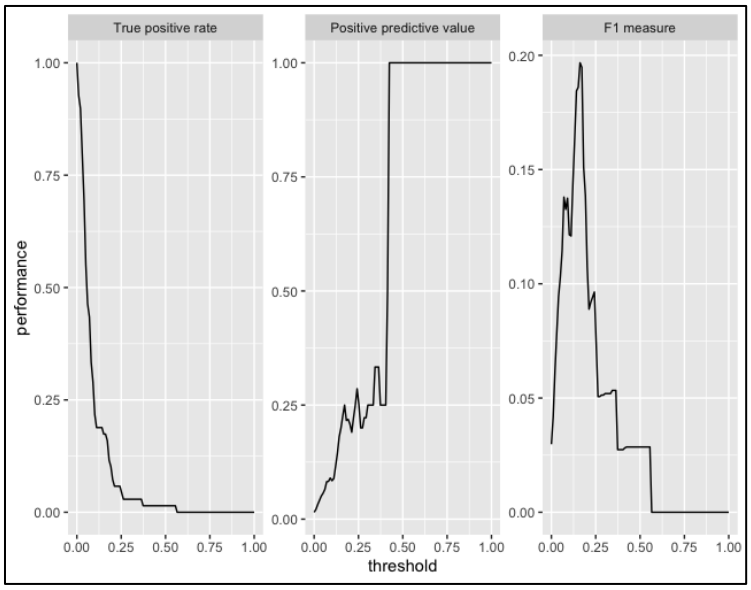

Figure 13: XGBoost Performance per Threshold

Similar to the logistic regression model development, we decided to set our final XGBoost threshold at the maximum f1 score: 0.16.

The final XGBoost confusion matrix and Receiver Operating Characteristic (ROC) curve can be seen below. Although it resulted in a low recall rate of 17% and low precision rate of 23%, most of the XGBoost evaluation metrics beat out both the simple logistic regression and the random forest approaches. We recommend XGBoost as the final model to predict the upset of a basketball game.

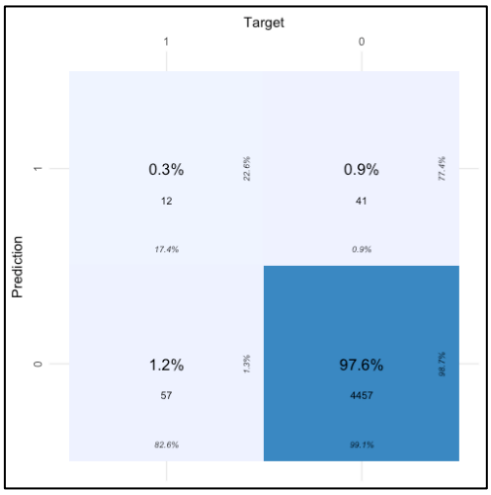

Figure 14: XGBoost Confusion Matrix

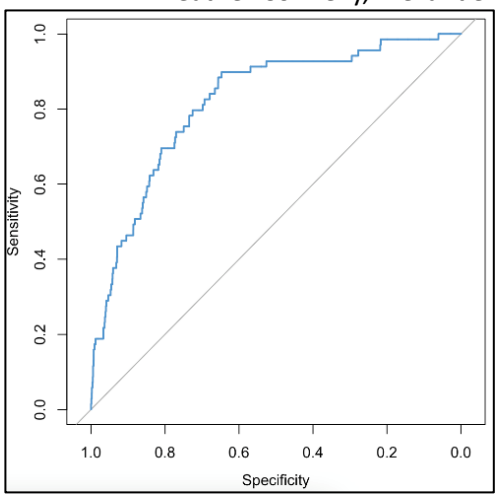

Figure 15: XGBoost ROC Curve

The feature importance plot below orders the final features by importance, which is calculated using the "Gain" metric. Gain estimates the relative contribution of the respective feature to the model by calculating the feature's contribution for each tree. From these results, we can observe that the number of points scored during a game, the two-point field goal percentage, the away team's steal count, and offensive rebounds, among others, all have high predictive power when predicting a college basketball upset.

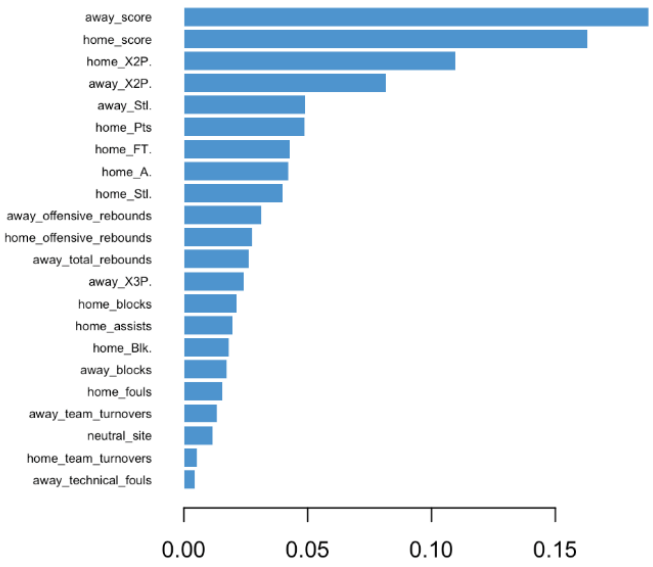

Figure 16: XGBoost Feature Importance*

*Note that most features are self-explanatory. The two features ending in _A represent the team's assists.

Conclusion

The team recommends developing the XGBoost model further beyond this study. The next step is to add additional variables back into the model and completing feature engineering. Variables such as the average experience of a team can be added and feature engineering can be completed by calculating the difference in the number of attempted three-point shots between the two teams, for instance.

In this study the game was marked as an upset if the pregame spread was greater than 10 points. While eliminating the spread did produce a better model, it would be worth exploring a different spread of 5 points to determine how the model behavior changes. By decreasing the game spread threshold for a positive upset classification, the percentage of upsets in the data set will increase, which may increase classification threshold resulting in a model with higher sensitivity compared to the models developed in this study.

Additionally, the models produced in this study would be useful to coaches of the team, as they are able to be used to identify areas for the underdog team to focus on as they are more likely to lead to upset. TV executives and gamblers, however, may want to make decisions in advance by building a model that solely relies on statistics for a team over the course of the season for predicting the likelihood of an upset.

Finally, the result of our study has shown that predicting an upset is possible and we feel with some additional variables and further refinement of the XGBoost model, it is possible to predict an upset with some degree of confidence.

Works Cited and References

[1] https://247sports.com/LongFormArticle/College-basketballs-10-biggest-upsets-of-the-2022-23-season-so-far-203714619/#203714619_1
[2] https://www.ncaa.com/news/basketball-men/bracketiq/2022-03-10/perfect-ncaa-bracket-absurd-odds-march-madness-dream
[3] https://www.oddsshark.com/ncaab/march-madness/how-much-is-bet
[4] KenPom: https://kenpom.com/
[5] ESPN: https://www.espn.com/
[6] Hoegh, A., Carzolio, M., Crandell, I., Hu, X., Roberts, L., Song, Y. & Leman, S. (2015). Nearest-neighbor matchup effects: accounting for team matchups for predicting March Madness. *Journal of Quantitative Analysis in Sports*, *11*(1), 29-37. https://doi.org/10.1515/jqas-2014-0054
[7] Brown, M. & Sokol, J. (2010). An Improved LRMC Method for NCAA Basketball Prediction. *Journal of Quantitative Analysis in Sports*, *6*(3). https://doi.org/10.2202/1559-0410.1202
[8] Brown, Bryce. (2019). Predictive Analytics for College Basketball: Using Logistic Regression for Determining the Outcome of a Game. Honors Theses and Capstones. 475. https://scholars.unh.edu/honors/475