NCAA March Madness Predictions
STAT479 Project Report

Shaun Campbell scampbell22@wisc.edu
Bailey Russo brusso3@wisc.edu
Ruichao Ma rma57@wisc.edu

Abstract

The NCAA March Madness tournament is an annual college basketball tournament featuring sixty-four teams. It is common to compete to predict the winners of each game before the tournament starts, often for money. Kaggle has hosted a competition for the last several years, and several researchers such as Luke Benz, Mark Bahuk, Bunker and Thabtah, and Moorthy and Zimmermann have developed algorithms designed for sports prediction. The goal of this research is to expand on prior research and develop a machine learning model to predict the probability of a team winning an NCAA tournament game. Doing so would create an edge in making predictions and could lead to profit.

Performance is judged by a combination of accuracy (correct predictions) and log-loss (accurate probabilities). Several algorithms were explored, and only the best features for each algorithm were used to fit the models. Naive bayes, logistic regression, bagging with KNN, and random forest had the best, and similar, results. These classifiers were combined in a stacking classifier for the final model. Running the model on the test set resulted in an accuracy of 72.4%, but a log-loss of 0.568 which is only a slight improvement over random guessing (0.693).

The results show that machine learning can improve on random guessing when it comes to college basketball matchups. The most important feature was seed, meaning that experts are credible and going by seeding alone is a reasonable prediction method. Above all, the research showed the difficulty of sports predictions due to the high random component and the wealth of unquantifiable features.

1. Introduction

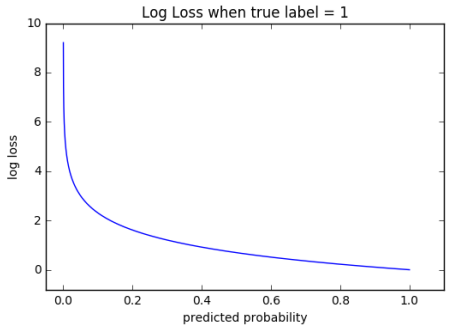
NCAA (National Collegiate Athletic Association) is a nonprofit organization, which regulates American student athletes. Each year, it holds several sports tournaments and basketball is one of most popular ones. The NCAA March Madness basketball tournament is one of most exciting times of year in the sports world, with sixty-four teams competing to win the national championship. It is a tradition to predict winners of the tournaments by filling out mock brackets with how one thinks the tournament will go. Those who have the highest prediction accuracy can win money in money pools. Once, Warren Buffet offered $ 1 billion to anyone who made a perfect bracket [11]. This project is a perfect target for machine learning because tournament outcomes can be predicted based on teams' past performance, each player's shooting averages, and other factors. Based on these features, the probabilistic nature of predicting wins and losses can hopefully be quantified. Machine learning concepts will be applied to predict which team will win each game of tournament.

Over the past few years, Kaggle has started to host "Machine Learning Madness" as a competition. Therefore, there is a large body of related work. The competition is joined to access the large datasets that are provided by Kaggle [7] and see how the machine learning models developed in this report will fare with the competition.

This project is meaningful to us for several reasons. We can combine all knowledge we learned in this course to this project. Thus, we can build our foundational knowledge of machine learning by doing this project. Moreover, we are all interested in basketball. Combining our personal interests with practical uses in our major will create a satisfying experience. Finally, this project also has certain societal influences because there is no widely-accepted algorithm to predict basketball game outcomes.

The goal of this project is to predict the probability that a team will beat another team. According to Kaggle's evaluation guideline, the predictive model will be based on log-loss. A visualization and equation of the log loss function can be seen in Figure 1.

The original Kaggle dataset contains over 20 features. Feature selection methods were used to select the best features for each model. Next, the data was divided into



$$H_p(q) = -\frac{1}{N}\sum_{i=1}^{N} y_i \cdot log(p(y_i)) + (1 - y_i) \cdot log(1 - p(y_i))$$

Figure 1. Log-Loss Function and Equations

a training set and test set, and 10-fold cross validation was used to fit data into the four best statistical models: Logistics regression, naïve Bayes, bagging with KNN, and random forest. Finally, a stacking method was used by combining the selected four models to produce the final predictive model, which should have highest predictive accuracy and lowest log-loss.

2. Related Work

Predicting winning probabilities in sports is a common topic and many previous researchers have done substantial amounts of work. Prediction work basically involves two general forms: The first method uses teams' final ranks and corresponding coefficients. Typically, this is done via logistic regressions and was popularized by Luke Benz [4]. However, by using this method, accuracy and final scores are limited because it is focused only on the final score of the competitions, not how players get their final rankings. To address this problem, subsequent researcher Mark Bahuk [3], a director of business analytics for the Oakland Athletics, proposed a new method that works on cumulative win probabilities based on both team and individual performances. However, this method also has its drawbacks, including missing data, incomprehensive features and inefficiency of generating accurate model to unseen data. As a result, researchers have turned to machine learning to better handle unseen data. Machine learning is a mature classification learning system, which bases predictions on numerous features.

With respect to the use of machine learning in winning probabilities prediction, Bunker and Thabtah [5] provide a good overview. They underscore the importance of classification. Sports predictions need to be treated as a classification problems, and the outcome of win or loss can be interpreted as binary numbers. Binary functions do not need too many computational resources and are highly interpretable, and do not require input features to be scaled. Thus, binary classification is used in this project. Before starting this project, previous papers about machine learning used in sport predicting were investigated. Shi, Moorthy and Zimmermann [10] published a paper related to predicting sports outcomes based on machine learning. However, in this paper, the final model's accuracy is not very high and it does not conduct feature selection, making it computationally expensive. Thus, this paper gives room for expansion and learning about what causes such problems and how to avoid them: By using stacking methods to combine several models with high prediction accuracies and selecting only the most important features to be computationally less expensive. To some extent, we stand on giants' shoulders and seek to improve the precedent method gradually.

3. Proposed Method

Before even attempting to run various learning algorithms on a given dataset, we need to create such a dataset in the first place. Every good machine learning model, at its core, will contain data or feature variables that are predictive. In other words, in order to make an accurate machine learning model, we need to find feature variables that provide us a lot of information about what we are trying to predict.

In our case, we want to predict the probability that one basketball team will beat another basketball team. Thus, we need to find feature variables that are indicative of a team winning or losing a given game. To find such feature variables, we observed a variety of different team statistics and analyzed how these statistics correlated with winning and losing. Some examples of team statistics we used in our final model that provided us useful information included turnovers per game, three-point field goals made per game, and defensive rebounds per game. In Figure 2, the winning team distribution of the given team statistic is plotted against the losing team distribution of the same team statistic.

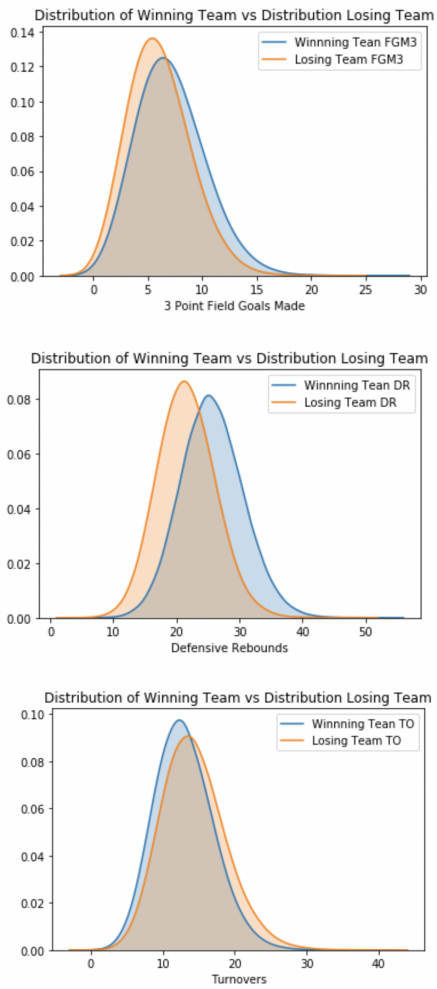


Figure 2. Distributions for Various Stats, Winning Team versus Losing Team

As we can see from the figures above, the winning team distribution for defensive rebounds per game and three-point field goals made per game is much higher, on average, than the losing team distribution. Alternatively, the winning team distribution for turnovers per game is lower, on average, than the losing team distribution.

These results make intuitive sense as more three-point field goals made per game and more defensive rebounds per game will likely result in a greater chance of a team winning. Teams that have a high number of defensive rebounds per game will give a team more possessions and in turn, more chances to score more points. Teams that make more three-point field goals per game will likely end up scoring more points since these kinds of shots are worth more. Alternatively, teams that turn the ball over a lot give the other team more possessions and more chances to score more points. Thus, it makes sense that winning teams will likely not turn the ball over that much.

Another important factor we wanted our model to take into account were team matchups. In other words, we not only wanted to measure how good a certain team was, but also how good the team's opponent was. In order for our model to capture these "matchups", we took certain team statistics and then subtracted these team statistics from the corresponding statistic for the team's opponent. We used this difference in team and opponent statistics as feature variables in our final model. For example, we took a team's seed and subtracted it from the opponent's seed, and we would use this seed difference as a feature variable in our final model. If a team's seed was 1 and the opponent's seed was 8, we would input -7 in our model, indicating that a team was ranked 7 seeds higher than their opponent.

After choosing feature variables that we observed to be good predictors, we ran numerous binary classification algorithms on our dataset to predict the probability that a given team beats another team. Our target variable was whether or not the team won or lost. We represented a 1 as a team winning and a 0 as a team losing. Some binary classification algorithms we ran on our dataset included logistic regression, naive bayes, k-nearest neighbors, and decision trees. We also utilized various ensemble methods in order to reduce bias as well as variance. These methods will be discussed in more detail in the sections that follow.

4. Experiments

4.1. Software

This report made use of Jupyter Notebook [6] and the Python packages scikit-learn [8] and mlxtend [9].

4.2. Dataset

The dataset obtained from Kaggle is very large, and regular season data may not be very useful in predicting tournament winning probability because of players' physical conditions, pressure, varying coaching strategies, etc. Thus only data from past tournament games was used for class labels. The dataset features 1,048 tournament games and numerous features such as final scores, turnovers, and field goals made.

In the original dataset, game stats are labeled by winning team and losing team. To adjust this, winning team and losing team are randomized into team and opponent so that the class label vector is balanced with 1's and 0's, and not all 1's or all 0's. Then the individual game statistics are changed to be average statistics across the team's regular season prior to the tournament. Stats for win percentage, win percentage against common opponents, and tournament seeds were added. The difference between the team and opponent stats were taken to capture both team strength and opponent strength while

halving the number of features.

4.3. Feature Selection

The data was then split into a training and a test set. The training set consisted of games from 2003 to 2014, and the test set was games from 2015 to 2018 (this is the test set that Kaggle evaluates on). The test set and training set were 268 and 780 observations respectively. Since the training set is small, an independent validation set is not used. Instead, 10-fold cross-validation is utilized for algorithm and model selection, while the test set was retained for final testing. The algorithms that were explored were:

Logistic Regression	Naive Bayes
KNN	Decision Tree
Random Forest	Bagging
Boosting	Stacking

For each algorithm, exploration into the best features was conducted. Each algorithm that was explored has a method via SciKit-Learn or MLxtend that can extract feature importances from the model. The best features were sequentially added for each algorithm and evaluated with a 5-fold cross-validation until performance started to decline due to too many features. Then the subset of the best features for the algorithm was selected for further evaluation. The five features that were consistently important regardless of the algorithm were seed, win percentage, win percentage against common opponents, and field goals made. These features will each be summarized briefly.

Seed Seed in basketball refers to a rank given to a team prior to the tournament. Figure 3 shows the correlation between number of seeds and number of wins. Number of wins gradually decays as seed increases. Thus, for two teams who have a large seed differential, the team with a lower seed has a higher probability to win.

Win Percentage The second most important feature is win percentage difference. This refers to the difference in win percentages in the regular season before the tournament. Figure 4 shows plots between tournament wins and regular season wins. From these two plots, it can be seen that teams who win more in the regular season win more in the tournament, which is reasonable.

Win Percentage Against Common Opponents Third most important feature is differences of winning percentage against common opponents. Figure 5 is a plot between tournament win percentage and differences in winning percentage against common opponents. All else equal, when difference in winning percentage is zero, each team has equal chance to win. However, when a team has a better record against common opponents, that team wins more often.

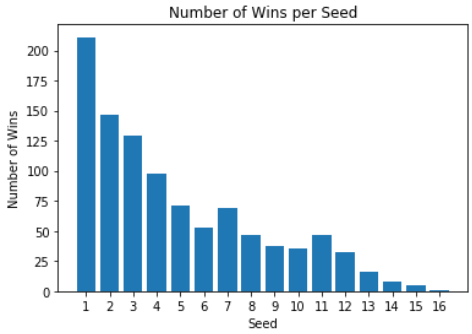


Figure 3. Relationship Between Seed and Number of Tournament Wins

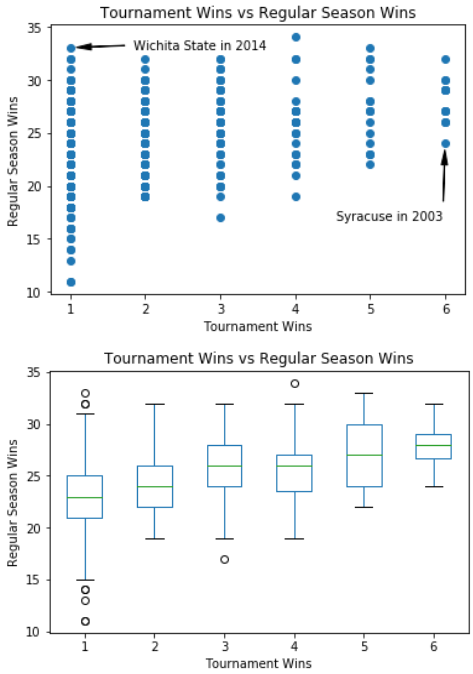


Figure 4. Regular Season Wins versus Tournament Wins

Average Score The fourth important feature is difference in average score. This is the difference in average score in the regular season before tournament. Figure 6 shows distribution plots of winning teams' average scores and losing teams' average score. As indicated by the plots, the winning teams' score distribution has a higher mean and median than the losing teams' score distribution.

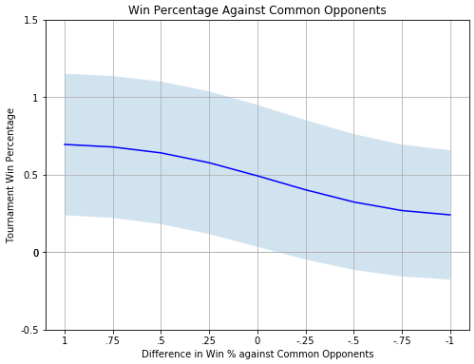


Figure 5. Tournament Win Percentage versus Difference in Win Percentage Against Common Opponents

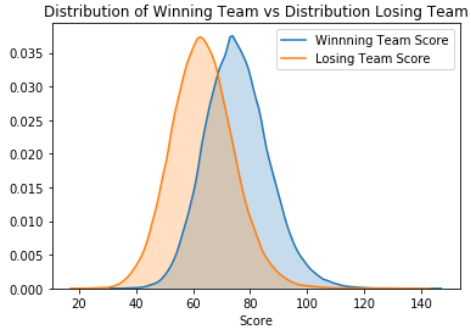


Figure 6. Distribution of Average Regular Season Score Between Winning and Losing Teams

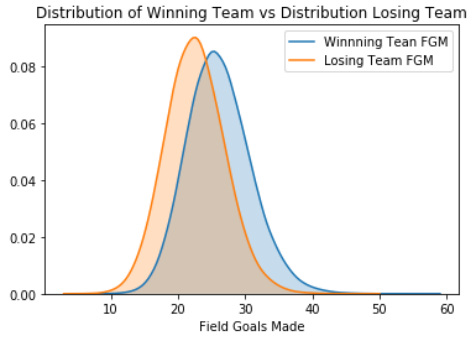


Figure 7. Distribution of Average Regular Season Field Goals Made Between Winning and Losing Teams

Field Goals Made The fifth important feature is difference in field goals made, which means difference in average field goals made per game in the regular season before tournament. High field goals would suggests excellent shooting skills. Figure 7 shows distribution plots of field goals made by winning teams and losing teams. The mean and median are higher for winning team, which makes sense since the team with higher shooting skills should have a higher chance of winning.

4.4. Algorithm and Model Selection

Model selection is performed to define the best set of parameters for each algorithms. Various sets of values for important parameters are tested for each algorithm and the best-performing parameter values are retained. After testing the different optimized algorithms, four performed well and had similar results: Random Forest, Naive Bayes, Bagging with KNN, and Logistic Regression. The algorithms each had similar mean log-losses and accuracies in the cross-fold validations. Each algorithm and model will be summarized briefly.

Random Forest Random forest is an ensemble method which usually uses a bagging method to create individual and uncorrelated decision trees. The prediction result is more accurate than that of any individual trees since bias reduced. Random forests can also avoid overfitting problems and are effective to identify important features in training dataset according to their feature importance weights. Features and select parameters of the random forest model are shown below in Table 1.

Table 1. Random Forest Features and Parameters

Features	Seed, OR, DR, AST, TO, STL, PF, Rank
Parameters	n_estimators = 100, criterion = gini, max depth = None

Naive Bayes Naïve Bayes uses Bayes's theorem to check independence of features. Generally, Naïve Bayes assumes features in the training dataset are conditionally independent. To illustrate, given random variables X, Y and Z, if and only if the probability distribution of X is independent of the value of Y given Z can we infer that X is conditionally independent of Y given Z. Generally, it is easy and fast to predict classifications since Naïve Bayes algorithms are simple and computationally cheap. They also work well on multi-class problems. Moreover, as long as independence assumptions hold, Naïve Bayes performs better than other classification models since less training data needed. Features used in the naive Bayes model are shown below in Table 2.

Table 2. Naive Bayes Features

Features	Seed, FGA3, FTA, TO, STL

Bagging Bagging is an ensemble method, which is a good tool to promote stability of the model and reduce variance. Generally, the first step in bagging is to bootstrap data with replacement and generate multiple subsample datasets, which require the same baseline predictor. After that, these training models will be run in-

dividually to unseen test data. Finally, the individual model performances are averaged to get the final decision. Since we use multiple sets of different data, it is a good way to reduce variance and loss. Features and select parameters of the bagging model are shown below in Table 3.

Table 3. Bagging Features and Paramaters

Features	Seed, Score, FGM, STL, PF
Parameters	Base estimator = KNN, n_estimators = 20, max_samples = 0.75
Base Parameters	n_neighbors = 20, distance = minkowski

Logistic Regression Logistic regression is also frequently used in binary classification. "Logistic regression is basically a supervised classification algorithm. In a classification problem, the target variable (or output), y, can take only discrete values for given set of features(or inputs), X."[2]. It usually builds a regression model to predict whether given data belong category "1" or "0". The following sigmoid function is used: $g(z) = \frac{1}{(1+e^{-z})}$[2]. Generally, it implements easily, and the result is more robust since independent variables are not necessarily normally distributed. The result of logistic regression can be easily interpreted. Moreover, it gives measures of how relevant predictors are and the corresponding moving direction (positive and negative). Features used in the logistic regression model are shown below in Table 4.

Table 4. Logistic Regression Features

Features	Seed, CommOpp, Score, DR, AST, TO, BLK, Rank

The resulting performance metrics of the models on the 10-fold cross validation are shown below in Table 5. The mean of the 10 folds are used to judge the model log-loss and accuracy. The four models will be combined into a stacking classifier for the final model. In stacking, multiple learners can be built and these models can be used to build intermediate prediction. Each model fits one prediction, new models are continually added which learn from the intermediate predictions[1]. Consequently, the final model prediction accuracy should be the highest.

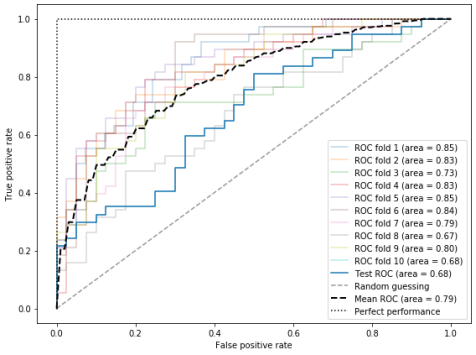


Figure 8. ROC Curve for Stacking Classifier

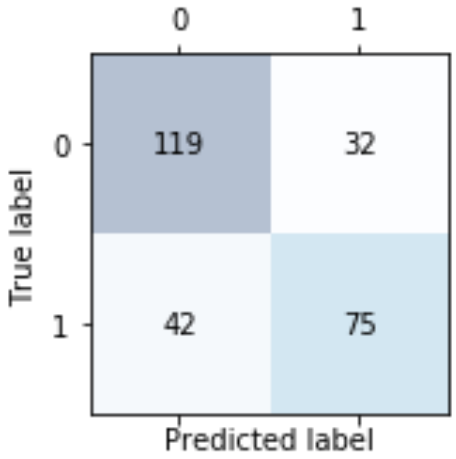


Figure 9. Confusion Matrix for Test Set Results

Algorithm	Log-Loss	Accuracy
Naive Bayes	0.5515	70.88%
Bagging with KNN	0.5521	71.26%
Random Forest	0.5864	68.18%
Logistic Regression	0.5999	68.94%

Table 5. Mean Performance Metrics of Algorithms, 10-fold CV

4.5. Results

The stacking classifier is run on the test data. The classifier is also run with 10-fold cross validation to produce an ROC curve, shown in Figure 8. The mean area under the curve is 0.79, which is better than any individual of the four models. The test set shows a decline in performance, with an area under the curve of 0.68.

A confusion matrix for the results of the test are shown in Figure 9. The results are fairly balanced but show a slight bias towards predicting losses. The evaluation statistics for the test are shown below in Table 6.

Algorithm	Log-Loss	Accuracy
Stacking	0.568	72.39%

Table 6. Test Evaluation Statistics

5. Discussion

Our final stacking model predicts historical games with an accuracy of about 72.39 percent, which is a considerably high rate. Our log loss of 0.568 however, is higher than we would have liked it to be. Our log loss indicates that we predicted the outcome of games that we were confident in as being either a win or a loss incorrectly. In other words, it appears that we had a large number of confident but wrong predictions.

Our model also tells us that the most important variables when determining whether or not a given team will win a tournament game is largely dependent on the two teams' seed difference, regular season win percentage difference, regular season win percentage against common opponents difference, regular season point per game difference, and regular season field goals made difference. These five factors alone account for a large majority of the variance in our data and prove to be good predictors in whether or not a given team beats another team in the March Madness tournament.

Our model also tells us that certain team trends in the regular season will often continue into the tournament season. For example, teams that have high regular season win percentages will on average, continue their winning trends into the tournament season. Teams that score a lot of points per game and have high-powered offenses in the regular season, will continue to exhibit such offenses going into the tournament season. Thus, regular team statistics seem to be positively correlated with corresponding tournament team statistics.

While we are generally pleased with our results, there are a couple of drawbacks as well as further improvements we believe we can make to our model. For one, our model does not take into account injuries. If a team's best player was injured throughout the entirety of the regular season but came back healed just in time before the start of the tournament, that team's regular stats may not be representative of how good that team actually is at full strength. Alternatively, if a team's best player got injured just before the tournament started, that team's regular season stats will only take into account how good the team is at full strength. The model will not factor in this team's injury to their best player and will perform assuming the team is at full strength like it was during the regular season.

Also, one can make the argument that our training data is outdated. Our data is trained on games during the 2003 season and the game has obviously changed considerably since then. Perhaps a lot of the data our model is trained on is not representative of how the game is played today. Going hand-in-hand with this idea is the fact that rules change every year in the NCAA. For example, the shot clock was reduced from 35 seconds to 30 seconds a couple of years ago. Minor changes to the rules such as this can influence and skew results from year to year, and in turn, influence our predictions.

Our model also does not take into account the intangible skills of a team such as experience, the ability to perform under pressure, mental mood, etc. While all of these are large factors in determining whether or not a given team will win or lose, these factors were not included in our model mainly due to the difficulty in measuring such factors.

6. Conclusions

The results of the experiment show that it is possible to improve on random guessing when it comes to making college basketball game predictions. The test results displayed a high accuracy, significantly improved over random guessing. However, the log-loss was only a slight improvement over random guessing. This is because going by seed differential alone will result in a greatly increased prediction accuracy over random guessing. Therefore, the slight increase of accuracy above what one would get using only seeds is on par with the slight decrease in log-loss.

The importance of seed differential in prediction accuracy shows that the experts doing the rankings are credible. It also suggests that the seeding by experts can perhaps encapsulate unquantifiable variables that only humans can identify. Going off seed differential alone would prove to be a simple and valid prediction method. The model developed in this research improved slightly on using only seeds but is much more computationally expensive. Furthermore, the stacking classifier is computationally expensive and only improves slightly on simpler methods such as naïve Bayes and logistic regression.

The research showed the inherent difficulty in making sports predictions. Sports have a very high random component and any team can win or lose on any given day. There is a great deal of unquantifiable features such as players' physical condition, mental pressure, and coaching strategies to name a few. However, this research showed that utilizing machine learning techniques can lead to slightly improved probability predictions for college basketball tournament games.

7. Acknowledgements

The dataset used in this report was retrieved from Kaggle, and was provided to Kaggle in large part by

Kenneth Massey and assembled in large part by Jeff Sonas of Sonas Consulting [7].

8. Contributions

Data cleaning and algorithm coding was divided evenly among the group members. We met approximately weekly and worked on the code together, and decided what coding tasks each member would complete before the next meeting. The presentation was also created together in-person. For writing the report, the tasks were divided:

- **Ruichao:** Introduction, previous work, experiment, results.
- **Shaun:** Latex file, abstract, experiment, conclusion.
- **Bailey:** Proposed model, experiment, discussion.

References

[1] Stacking in machine learning. *GeeksforGeeks*.

[2] Understanding logistic regression. *GeeksforGeeks*.

[3] M. Bahuk. Using cumulative win probabilities to predict ncaa basketball performance. *roceedings of the MIT Sloan Sports Analytics Conference*, 2012.

[4] L. Benz. Ncaa basketball win probability model. 2017.

[5] Bunker and Thabtah. A machine learning framework for sport result prediction. *Applied computing and informatics*, 2017.

[6] T. Kluyver, B. Ragan-Kelley, F. Pérez, B. Granger, M. Bussonnier, J. Frederic, K. Kelley, J. Hamrick, J. Grout, S. Corlay, P. Ivanov, D. Avila, S. Abdalla, C. Willing, and J. D. Team. Jupyter notebooks – a publishing format for reproducible computational workflows. *Positioning and Power in Academic Publishing: Players, Agents and Agendas*, 2016.

[7] K. Massey and J. Sonas. Apply machine learning to ncaa® march madness®. Data retrieved from Kaggle, `https://www.kaggle.com/c/mens-machine-learning-competition-2019/data`.

[8] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay. Scikit-learn: Machine Learning in Python . *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[9] S. Raschka. Mlxtend: Providing machine learning and data science utilities and extensions to python's scientific computing stack. *The Journal of Open Source Software*, 3(24), Apr. 2018.

[10] Shi, Moorthy, and Zimmermann.

[11] R. Wile. Warren buffett will give you $1 billion if you fill out a perfect 'march madness' bracket. *Business Insider*, 2014.