Predicting March Madness with Machine Learning

Amie Corso Austen Kelly Erin McCarthy

Abstract—**Having the perfect bracket for March Madness is an elusive basketball fanatic's dream, and one that could be quite lucrative. There are many cash prizes for having the perfect bracket, or even just finishing with the best bracket. If one chose to flip a coin for each game the odds of a perfect bracket would be 1 in 9.2 quintillion (1). In this work we apply machine learning algorithms to predict the outcome of games in the March Madness tournament. We compare the results of four classifiers: Linear Regression, Logistic Regression, K-Nearest Neighbors and Neural Networks. Our results demonstrate that these models could not consistently outperform of the baseline of predicting the outcome of the game based on the team with the higher seed.**

I. Introduction

Each year the top 64 college men's basketball teams face off in the NCAA tournament, and each year billions of people enter their predictions of how it will all go down into office and national bracket pools. And so begins March Madness. Notoriously filled with dramatic upsets and game-winning buzzer-beaters, the dream of one day filling out a perfect bracket is dim but enticing. In fact, no person has ever been able to prove that they filled out a perfect bracket (2). As the years pass the prizes being offered for the perfect bracket increase. Before even two rounds of the 2018 March Madness tournament were over, there were no perfect brackets left (3). Thus raises the question: in the age of machine learning, can computers do what no other human has done and predict the perfect bracket? Our project takes a look at four different learning models and two different approaches to training and tuning to see if a simple classifier can provide an edge over picking all the favorites in a bracket.

II. Background

The idea to predict the March Madness tournaments came from a Kaggle competition offering $50,000 to a team that could best predict the 2018 tournament (4). This competition provided a variety of datasets ranging from 2003 to 2017 with statistics from past regular-season games and NCAA tournaments.

The NCAA tournament is formatted as a single-elimination bracket of 64 of the top teams in college basketball (along with 1 or 4 'play-in' games prior to the first full round). Thus for any given year, the NCAA tournament consists of 61 (or 64) games to predict. Each team in the four initial regions of the tournament is given a seed from 1-16 based on their overall performance in the regular season, and the tournament begins with 1-16, 2-15, etc. matchups.

The most granular data available on a game-by-game basis included the following statistics for both the winning and losing teams: season, day number, location (home, away, or neutral), score, number of overtimes, field goals made/attempted, three pointers made/attempted, free throws made/attempted, offensive rebounds, defensive rebounds, assists, turnovers committed, steals, blocks, and personal fouls committed. This information is provided for all regular seasons and tournaments played since 2003. Additionally, the Kaggle dataset includes information such as the seed of each team in the tournament, rankings of teams throughout the season, team coaches, and conference information.

III. Methods & Experiments

A. Data Selection

Of the possible features we could use in our predictions, we chose to omit several. Despite the fact that the home-court advantage is a real phenomenon (7), we chose to omit location as a feature because the success of our models was narrowly defined as success in predicting only the men's NCAA tournament results. The NCAA tournament is specifically designed to eliminate home-court advantage, thus all games are considered away. Even if we incorporated location into our training data, it would be uninformative given tournament games that are uniformly "away." We also omitted the number of overtimes and the day number of the season because these statistics are the same for both teams in a given game and seemed unlikely to provide additional information gain. For each of the remaining statistics, we generated an average value for each team for each season based on all games played by that team in that season. Additionally, we chose to engineer two additional features from the game-by-game data for inclusion in the team/season averages: average points allowed, and win/loss ratio. From this, we ended up with a total of 16 features to use in our analysis.

Finally, we min/max normalized (fit to the range [0, 1]) the vectors after all data points were generated. In other words, we did not normalize the average-statistic vectors that represented each team/season, but rather normalized each of the difference vectors that represented particular games. All regular-season game data was normalized with respect to itself, and all tournament game data with respect to itself.

B. Prediction Formulation

Our models attempt to predict the outcome of any particular basketball game. Since a game is actually a matchup between two distinct teams, we chose to represent a game as a single data point by generating a vector of differences between the relevant season-average statistics for Team 1 and Team 2. This difference vector contains the same set of features as those for each team, and is created by subtracting the value for Team 2 from the value for Team 1, for each feature. The model is then making a statement about whether Team 1 will win or not (a binary classification), or about the predicted score difference between Teams 1 and 2, depending on the model.

C. Models

We chose to compare four models: K-nearest neighbors (KNN), linear regression, logistic regression, and neural networks. All of these algorithms were implemented using SciKit Learn and normalized using SciKit Learn's preprocessing library. Since our goal was to accurately predict only NCAA tournament games, we were interested in whether there seemed to be a difference in relevance between regular-season games and tournament games when used as training data. We chose to run all experiments using these two different sets of data as training data. In one case, our training data comprised all 15 years of regular-season games, and our models were then validated and tested on individual tournament years. In the other case, we used tournament-only games as our training data, selectively omitting years to be used as validation and test sets. There is a huge difference in the volume of training data used in these two approaches: there are approximately $76,636$ regular-season games and only 981 tournament games. As a baseline from which to assess our accuracies, we computed the accuracy for each tournament that would have been achieved using only the value of the team's tournament seed to predict a winner. This is an interesting baseline because tournament seeds are rankings that theoretically take into account the variety of features that our models incorporate, as well as more nebulous assessments such as professional opinion and strength of seasons.

D. KNN

Our KNN algorithm is a typical k-nearest-neighbors learning algorithm (as opposed to a radius-based learning algorithm). Since KNN is a categorical classifier, our KNN model is simply attempting to make a binary classification: will Team 1 win or lose? The accuracy is calculated as the percentage of test points correctly classified.
Since KNN is an instance-based learning model, there are only a few hyper-parameters to tune. Most obviously is the choice of k, the number of nearest neighbors considered while classifying a new point. The choice and number of features to include is also particularly important in a KNN classifier. All features are weighted equally, therefore useless features quickly add noise to the "distances" between points and are likely to reduce accuracy.
One way that we attempted to make an informed decision about feature selection was by attempting to deduce the relevance of individual features, examined in isolation. We examined the accuracy achieved for a KNN-classification using each feature individually. We used the SciKit Learn default k-value of 5 for this experiment. Perhaps unsurprisingly, the most relevant features were average score, field goals made, win/loss ratio, and points allowed, each providing accuracies of around 60% when assessed on the pool of all regular-season games from the 2016 and 2017 seasons. This result also agreed with the results of (6) , for the features that we were using. The other features provided individual accuracies between 50% and 60%, so it seems they do add some information. This preliminary examination isn't enough to make conclusions about how these features might perform given more complex interactions among them. However, considering that there is an enormous number of potential sub-selections to make of features, it is unrealistic to test them all while seeking an optimal combination. This creates a basis for decision-making. Given the features ordered from most to least informative in the individual experiments, we assessed average accuracy of the model beginning with only one feature (the most informative), and progressively adding in subsequent features for consideration. As seen in Fig. 1, the results of this experiment showed consistent trends for values of k = 5, 20, 100 and 200, suggesting that the effect of feature inclusion has a mostly independent effect from the choice of k. Perhaps unsurprisingly, the best accuracies were achieved using a small set of only the 5 most informative features.
After settling on the 5 most informative features, our approach

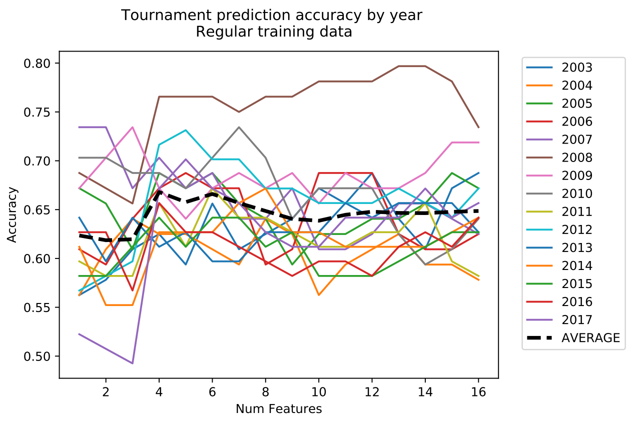

Fig. 1: KNN Average accuracy v.s. number of features included (k = 5)

to selecting the optimal value for k was a similar brute-force experiment. We calculated validation accuracies across a wide range of k-values and selected the final value for k that provided the best average accuracy across all tournament years. This experiment was performed using both the regular-season games as training data and the tournament-only games as training data. Initially, we chose values for k that spanned the entire availability of the respective training set. As expected, the optimum was clearly closer to 1, with a steady decline in accuracy as k approached the size of the training set, and finally a precipitous decline toward 50% accuracy when all training points were considered. In the case of regular season data, Fig. 2, the average accuracy was maximized for a k-value of approximately 40, whereas for tournament-only training data, Fig. 3, the optimal k was smaller, approximately 20. This reflects the large difference in the number of data points available for each scenario.

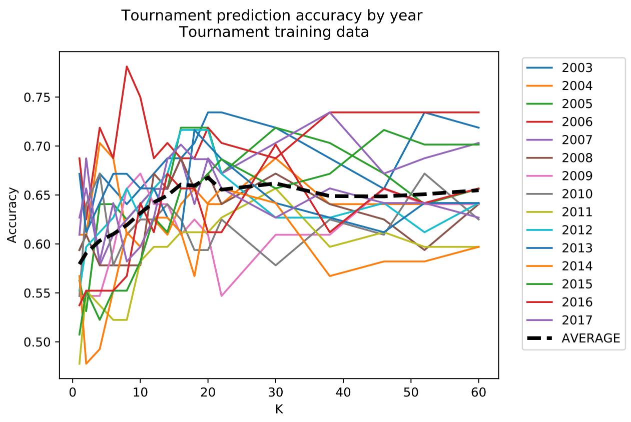

Fig. 2: KNN Average accuracy v.s. values of k, with regular-season training data

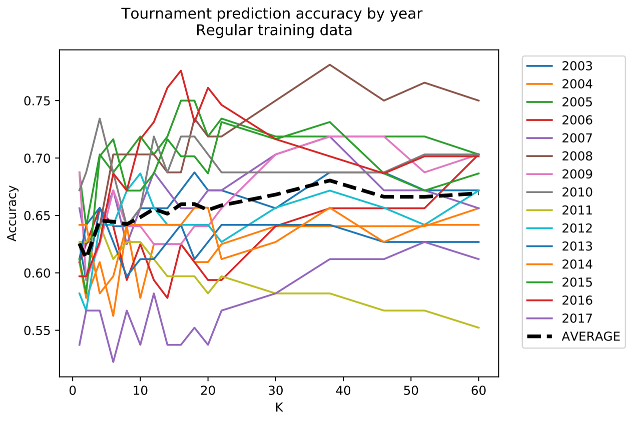

Fig. 3: KNN Average accuracy v.s. values of k, with tournament training data

E. Linear Regression

The goal of linear regression (linR) was to utilize score differences in a game in the learning process of the model. Instead of being presented with 0/1 representing win or loss it was presented with the score difference of the two teams playing. The implementation of this model was done by using SciKit Learn's (sklearn) algorithm with normalized input data. Other than regularizing sklearn implementation of linR does not allow for more customization.

F. Logistic Regression

Logistic regression (logR) was used to explore the accuracy of a simple linear classifier. It was given 0/1 as its labels to represent a loss (0) and win (1). It was again implemented using sklearn's algorithm with normalized input data. The hyper-parameters that could be tuned consisted of penalty, tolerance for stopping criteria, the inverse of regularization strength (C), the optimization algorithm, and the max iterations. The tuning was used based off of the 3-fold cross validation using sklearn's cross validation score. The final hyper-parameters were chosen based on the lowest difference between the accuracy of the three sections of training or validation data to prevent overfitting. While training on season data, the tournament data was used as validation data. Testing different values of the hyper-parameters did not make a difference, even when using a large C there was no significant change in accuracy on the tuning data. We then examined the impact of tuning the hyper-parameters on the tournament model, that was trained and tuned on tournament data. The optimization algorithm 'newton-cg' was used for the final model because it converged faster than the other optimization algorithm. The model was run for 100 iterations. Next we examined the best choice for C, a higher C resulted in greater difference in the validation scores, as expected since a higher C gives a model more ability to overfit the data. The final C used was 0.5. The tolerance value was kept the same since the model converged quickly.

G. Neural Network

In the neural network model, the aim is to predict a binary label 1 or 0 representing which team won. For neural networks, it is important to tune both the number of training iterations and the shape and size of the hidden layers. In order to tune the number of iterations, we set a hidden layer dimension of (8, 2), the activation function as relu, used stochastic gradient descent, and the L2 regularization coefficient as 0.00001, which seemed from initial tests to be reasonable choices. We then compare the average accuracy across all years versus the maximum number of iterations, as shown in Fig. 4 and Fig. 5. From these plots, we want to find the smallest number of iterations such that the training and validation accuracy has peaked, so as not to train longer than necessary and risk over-fitting. As a rough estimate, we chose to cap the number of iterations for the regular season-based model at 250 and at 750 for the tournament-based model.

Next, we examine the impact of changing the dimensions of the hidden layers on the accuracy with the pre-tuned numbers of iterations. Fig. 6(a) shows that when training on regular season data, the dimension of the hidden nodes has little to no discernible impact on the accuracy. However, when training on only tournament data, in Fig. 7(b) we see that this hyper-parameter has more significant impact. Based on this figure, we chose to use one hidden layer of size 12. While having two layers of sizes 12 and 8 results in similar accuracy, we opt for one layer to limit overfitting and to maintain a simpler structure.

H. Comparison

Finally, we compared our models for predicting the outcomes of each tournament in the years 2003 to 2017. For all methods, we evaluate our performance as the percentage of games that happened in the tournament that the model correctly predicted. Since the accuracy of the neural network

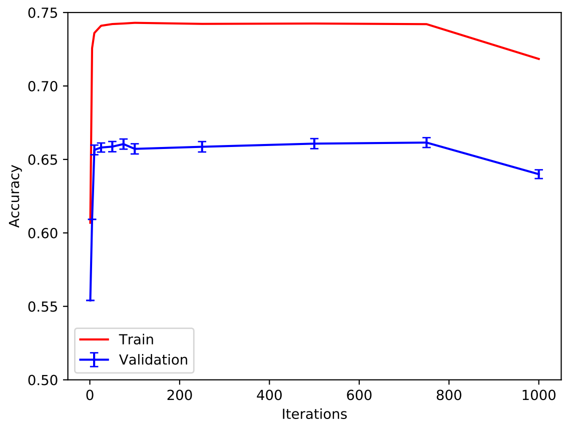

Fig. 4: NN Average accuracy v.s. number of iterations, with regular-season training data.

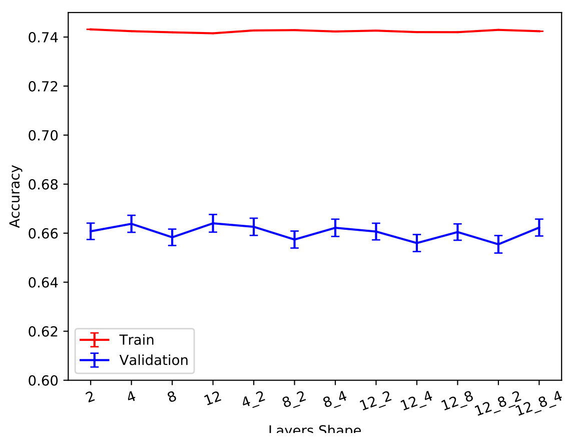

Fig. 6: NN Average accuracy v.s. hidden layer dimensions, with regular-season training data.

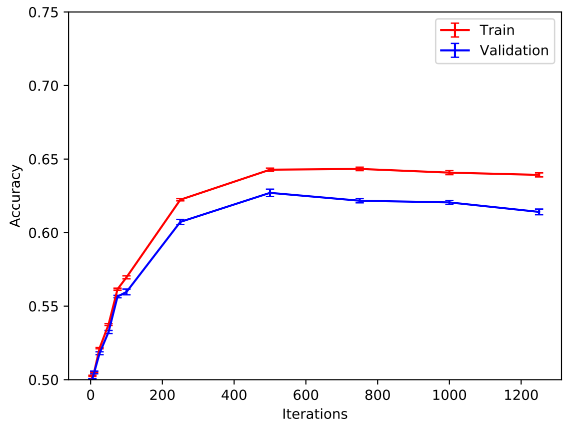

Fig. 5: NN Average accuracy v.s. number of iterations, with tournament training data.

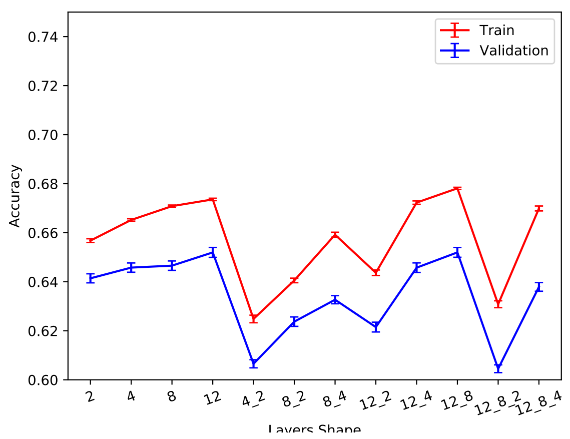

Fig. 7: NN Average accuracy v.s. hidden layer dimensions, with tournament training data.

model is impacted by the random initialization, it's accuracy is reported as an average over 10 trials. The results for training on only the regular season data versus training on only tournament data is shown in Fig. 8 and Fig. 9, respectively.

As shown in Fig 10 there was no model that performed consistently better on all or even most seasons. The baseline had the least amount of variation throughout the tournament predictions and on average performed better than the models we implemented. When taking into consideration which model was able to predict the most amount of games per tournament and ranked in that manner, the average ranking of KNN and linR slightly outperformed the baseline.

IV. Conclusion

The performance of a sports team is a complex phenomenon. Numeric statistics can capture certain aspects of a team's ability, but taking into account less concrete factors may be required to improve the accuracy of predictions. We had many ideas for future work that revolved around incorporating additional information. In particular, we did not incorporate tournament seed into our predictions, though this is one metric that may take into account expert opinions and other metrics such as "strength of season" that are not covered by the other features we used. Additionally, we considered whether time as a dimension could lend additional information. For example, does the win history of a team leading up to a game seem to affect their likelihood of winning? Is a team more likely to win if they have been showing consistent improvement across a season? Perhaps a recurrent neural network is a model that could allow us to incorporate some notion of time-based trending.

Ultimately, however, it seems based on all of the related works and our results that perhaps the extravagant prizes set in place for the person to finally create a perfect bracket are

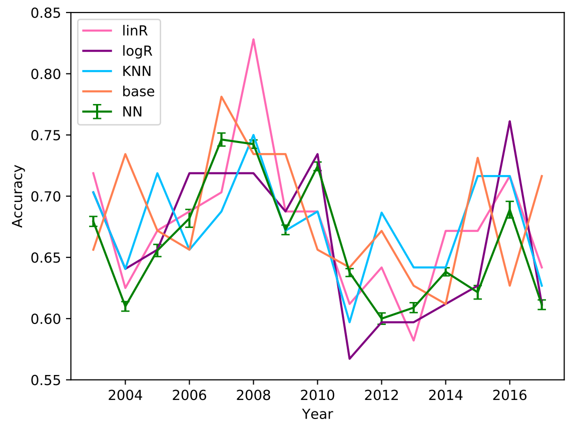

Fig. 8: Final comparison of accuracy for each year, with regular-season training data.

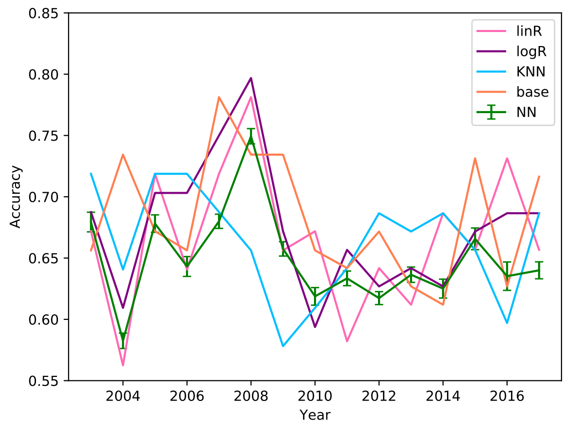

Fig. 9: Final comparison of accuracy for each year, with tournament training data.

not so extravagant after all. Few models exceed even 75% accuracy when predicting these tournaments, and each year some dramatic underdog upset comes out of nowhere to break even the best analyst's predictions. After all, it's not called March Madness for nothing.

	Base	KNN	linR	logR	NN
Reg. Season	**0.683458**	0.676166	0.676446	0.663417	0.661271
Tourneys	**0.683458**	0.663666	0.665936	0.674176	0.649459

Fig. 10: Average accuracy across years

V. References

1. Paine, N., & Boice, J. (2017, March 17). The Odds You'll Fill Out A Perfect Bracket. Retrieved March 18, 2018, from http://fivethirtyeight.com/features/the-odds-youll-fill-out-a-perfect-bracket/

2. Has anyone ever filled out a perfect bracket? (n.d.). Retrieved March 18, 2018, from https://www.si.com/college-basketball/2017/03/13/perfect-march-madness-bracket-possibility

3. Wilco, D. (2018, March 18). Last perfect bracket busts after UMBC pulls off biggest upset in NCAA tournament history. Retrieved March 18, 2018, from https://www.ncaa.com/news/basketball-men/bracketiq/2018-03-17/last-perfect-bracket-busts-after-umbc-pulls-biggest-upset

4. Google Cloud & NCAA ML Competition 2018-Men's. (2018, February). Retrieved March 18, 2018, from https://www.kaggle.com/c/mens-machine-learning-competition-2018

5. Wilde, A., & Forsyth, J. (2018, February). A Machine Learning Approach to March Madness[Scholarly project]. Retrieved March 18, 2018, from http://axon.cs.byu.edu/ martinez/classes/478/stuff/Sample_Group_Project3.pdf

6. Deshpande, A. (2017, March 12). Applying Machine Learning To March Madness. Retrieved March 18, 2018, from https://adeshpande3.github.io/Applying-Machine-Learning-to-March-Madness

7. Jamieson, J. P. (2010), The Home Field Advantage in Athletics: A Meta?Analysis. Journal of Applied Social Psychology, 40: 1819-1848. doi:10.1111/j.1559-1816.2010.00641.x