ACCURACY OF MACHINE LEARNING ALGORITHMS IN PREDICTING COLLEGE BASKETBALL GAMES

A Research Paper submitted to the Department of Computer Science
In Partial Fulfillment of the Requirements for the Degree
Bachelor of Science in Computer Science

By

Sindhura N. Mente

April 25, 2022

On my honor as a University student, I have neither given nor received unauthorized aid on this assignment as defined by the Honor Guidelines for Thesis-Related Assignments.

ADVISOR
Haiying Shen, Department of Computer Science

INTRODUCTION

College basketball is an extremely popular sport that has become a lucrative industry in terms of producing revenue, especially in the realm of sports betting. For the 2022 NCAA men's basketball tournament an estimated 3 billion dollars would have been spent on betting (Korpar, 2022). Developing a project that could potentially provide better insight into the behavior of the game was determined to be a valuable endeavor.

We reasoned that with the expansive datasets that exist online on the records of the performances and outcomes of NCAA men's basketball games we could apply Machine learning techniques to develop a model that could accurately predict the outcomes of games. College basketball is a sport that involves a large range of statistics in order to measure individual and team performance. Figure 1 below is a list of the range of statistics that were available and we used as features in our model.

Four Factors

Offensive	Defensive
Offensive Efficiency	Defense Efficiency
eFG%	eFG%
TO%	TO%
OR%	OR%
FTRate	FTRate

Team Stats

Avg Height
Eff Height
C Hgt
PF Hgt
SF Hgt
SG Hgt
PG Hgt
Experience
Bench
Continuity

Figure 1: Four Factors. This figure shows the range of basketball features used to build the models. (Kim & Mente, 2022).

RESULTS FROM SIMILAR RESEARCH

A team of researchers from KU Leuven approached the problem by using J48, Random Forest, Naive Bayes, and Multilayer perceptron. The team of researchers used similar features to our own including the Four Factors. One observation that was made by the team was that the

simpler model Naive Bayes performed the best out of all the approaches and also that using additional features, outside of the four factors, lead to worse results (Moorthy et al., 2013, p.8).

Another team of researchers from the University of Pittsburgh tried predicting the outcomes of NCAA basketball games using Adaptive Boosting, K-nearest neighbors, Naive Bayes, Neural Networks, Logistic Regression, Support Vector Machine, and Random Forest. The team discovered that Logistic Regression had the best performance in terms of scoring points when used to fill out a bracket for the men's basketball tournament (Levandoski & Lobo, 2017, p. 13).

Both teams felt that they encountered a limit to which any model would be able to predict a game. The researchers at KU Leuven had a limit of around 74-75%. The researchers from the University of Pittsburgh found that the models had a better performance than an average person but luck/volatility still made it difficult.

PROCESS AND METHODOLOGY

INITIAL PROCESS AND PROOF OF CONCEPT

Initially the focus was on data from the Athletic Coast Conference (ACC) during the 2020-2021 season. The reason for this was that college basketball is unique due to the high turnover rosters have from season to season, and it would be difficult to extract the most significant features from data across multiple seasons. Additionally, first fitting a variety of models on a small subset of the data and observing the outcomes to ensure that the data cleaning step was performed correctly would be more prudent than modifying all of the data sets and then coming to the conclusion that the data were being modified incorrectly. First, we gathered all of the necessary data on the ACC from the KenPom website (kenpom.com) which includes data regarding each collegiate team including parameters such as free throw rate, point differentials

(the difference in final points scored for a particular game between the chosen team and the opposing team), and bench, which is the percentage of the minutes played where none of the original starters are playing. Each of the features, and how they were calculated, for each team is listed in Figure 2 below.

Statistical Category	Meaning
Offensive eFG%	(Field Goals Made + 0.5 * 3 Pointers Made) / Field Goals Attempted.
Offensive TOP%	TO% = TO / Possessions
Offensive ORB%	OR% = Offensive Rebound / (Offensive Rebound + Opponent Defensive Rebound)
Offensive FTR%	Free Throw Attempt / Field Goal Attempt
Defensive eFG%	(Opponent Field Goals Made + 0.5 * Opponent 3 Pointers Made) / Opponent Field Goals Attempted.

Defensive TOP%	Opponent Turn Over / Opponent Possessions
Defensive ORB%	Defensive Rebound / Defensive Rebound + Opponent Offensive Rebound
Defensive FTR%	Opponent Free Throw Attempt / Opponent Field Goal Attempt
Exp	The average years that roster spent in college. Freshman - 0 Sophmore -1 Junior- 2 Senior 3
Bench	Percentage of the minutes played where none of the original starters are playing
Size	It's the average of the heights of all the players multiplied by the total number of minutes each player played
Point Differential	The feature we were trying to predict. tm - opp

Figure 2: Formulas for Features. This figure shows the calculations done to create the features. (Kim & Mente, 2022).

We pulled every team's records and game scores from the ACC during the 2020-2021 season from https://www.sports-reference.com/cbb/ and cross-referenced this data with that extracted from KenPom to ensure that there weren't discrepancies, and then created a final data frame for each team in the ACC with all of the information regarding the games each team played in the chosen season. An example, for the University of Louisville, is pictured below in Figure 3.

Louisville_2020_2021_FINAL

	Tm	Opp	Opp Off eFG%	Opp Off TOP%	Opp Off ORB%	Opp Off FTR%	Opp Def eFG%	Opp Def TOP%	Opp Def ORB%	Opp Def FTR%	Opp Exp	Opp Bench	Opp Size	tm Off eFG%	tm Off TOP%	tm Off ORB%	tm Off FTR%
0	79	44	53.194765	17.317249	19.029374	25.404157	57.370518	18.328475	24.163028	34.342629	2.30	21.12	76.36	47.774869	17.23554	32.00569	32.111693
1	71	70	50.032787	18.859537	29.613734	35.672131	49.428934	19.182843	29.778247	27.284264	2.38	24.24	79.19	47.774869	17.23554	32.00569	32.111693
3	75	54	50.520196	20.453518	31.770833	35.495716	49.385965	18.868742	24.207493	22.397661	2.12	30.64	76.31	47.774869	17.23554	32.00569	32.111693
4	48	85	49.944598	13.640654	23.791822	26.260388	47.856315	17.343117	25.166826	28.447277	2.38	28.77	77.97	47.774869	17.23554	32.00569	32.111693
5	64	54	48.829953	18.321827	34.487021	36.895476	50.000000	17.458842	26.143791	33.226581	1.27	33.54	77.63	47.774869	17.23554	32.00569	32.111693
6	62	59	47.086721	19.756581	32.644178	34.485095	46.679816	17.945089	30.522946	29.651545	1.00	31.72	78.95	47.774869	17.23554	32.00569	32.111693
7	76	64	49.789030	19.156790	25.373134	29.704641	55.463576	18.465209	28.656716	29.056291	1.76	33.86	76.95	47.774869	17.23554	32.00569	32.111693
8	73	71	52.258065	17.183288	27.830832	33.387097	48.842975	17.924528	24.353741	31.900826	1.55	28.51	76.42	47.774869	17.23554	32.00569	32.111693
9	77	65	48.975235	20.014976	26.158038	28.522630	54.451167	17.900718	25.909091	32.065687	1.70	38.13	77.51	47.774869	17.23554	32.00569	32.111693
10	72	78	47.375328	18.415458	27.032735	32.808399	52.546584	17.927990	26.259378	19.751553	1.80	30.00	78.61	47.774869	17.23554	32.00569	32.111693
11	65	78	54.004107	20.421042	34.367246	32.443532	46.143345	19.513440	31.663113	35.153584	1.96	39.17	79.62	47.774869	17.23554	32.00569	32.111693
12	70	65	53.583502	18.331368	32.227488	23.394185	51.316752	19.883862	28.957055	31.894660	0.86	24.72	77.18	47.774869	17.23554	32.00569	32.111693

Figure 3: Louisville Dataframe. This figure shows an example dataframe. (Kim & Mente, 2022).

To create these data sets, we started by cleaning the data. First, we filled all of NaN values in the raw datasets by filling them in with the mean value of that respective column, and if any of the categorical variables contained NaN values, then we dropped the row completely. We then modified the names of some variables for understanding and consistency across datasets for all colleges. Afterward, we proceeded to start the model building and training process.

To verify that our results were viable, we chose to train a variety of models on the University of Virginia's (UVA) ACC data from 2020 to 2021 to predict the point differentials relative to UVA.

The first model trained was a simple linear regression model that considers all of the aforementioned features when making predictions, and we chose to use the linear root mean square error (RMSE) to evaluate the utility of the model. The next model we trained was a decision tree regression model, and the error we chose to evaluate the algorithm was the tree RMSE. After this, we tried a random forest regressor and similarly used random forest RMSE to evaluate the error. The last model we implemented was a logistic regression model.

EXPANSION TO COMPLETE ACC DATASET

After verifying that the models were valid and provided believable results, we proceeded to expand the dataset to include the features of size, experience, and bench minutes shown in Figure 1, in order to expand the work of a previous paper. We also then expanded the dataset further to include data for the entire ACC as opposed to just UVA. Next, we trained the data for each of the schools using a linear regression model and obtained relatively low linear RMSEs.

EXPANSION TO ALL COLLEGIATE CONFERENCES

Since only teams in the Athletic Coast Conference (ACC) were included in the models created, it was determined that expanding the dataset to include a wider range of conferences and teams would assist in making the model's prediction more accurate given the large field of college basketball. The conferences added were the Big Ten, Big 12, PAC 12, and the Southeastern Conference, raising the total number of teams included from 14 to 63. Our preliminary findings showed that logistic regression was the best model since it yielded an error of 9.992007221626415e-16; however, we believe this may have been due to overfitting of the data.

REMOVAL OF BIAS

The errors of the models were deemed suspiciously small and we recognized that overfitting was occurring. We analyzed each of the features to attempt to identify which ones may have given rise to this problem, and were able to identify the “Tm“ and “Opp” features of the dataset being mainly responsible for the overfitting. The reason for this was that “Tm“ and “Opp” are directly used in the calculation of “Final Point Differential”. We then reran the models with those features removed from the datasets for each of the colleges, which yielded errors that were significantly larger than the errors we obtained initially, confirming that overfitting was present in the models. By removing the features that were causing this overfitting, we were able to reduce the inherent biases present in the data. This in turn allowed us to create models for each college participating in March Madness that would be able to be extrapolated to future seasons and beyond the scope of the training data to have meaningful impacts in the sports industry.

In doing so, we determined that logistic regression and not linear regression was the model best suited for the problem, since the goal of the capstone project is to accurately determine the outcome of a basketball game given parameters about the teams. Logistic regression encodes the variable being predicted, in this case final point differential, as a 0 or 1, where 0 indicates that the chosen team lost, and 1 indicates that the chosen team won. This appeared to be the best model for the problem it is intended to solve, since we obtained a log error of 0.596255892458005, so we proceeded with the logistic regression models for each school for the remainder of the project.

FINAL MODELS AND GAME PREDICTION

Once the overfitting and bias were removed from the data, and the final data frames for each college were made, we trained models respective to every college on the data associated

with that college, and then saved the models for each of the 63 colleges in a list. Since the models created, satisfied our initial goal was to develop an algorithm to successfully predict the outcome of a single basketball game given certain parameters regarding the teams and player, we decided to test the models we built on the NCAA March Madness bracket for the 2021-2022 basketball season, since all of the models we trained used data from the previous 2020-2021 season. After this, we considered different methods to create a March Madness bracket solver in order to test the power and robustness of our models.

In our research, we came across the bracketology python package, which provides a historical database of all previous NCAA tournament brackets (PyPI, 2020). The package is pertinent to our project due to its ability to provide an overview of model effectiveness in deciding the outcome of games, and by extension, the winners of the NCAA tournament. Figure 4 below shows an example output of running the bracketology March Madness simulator with the model used to predict the outcome of a game being a coin flip.

```
# Initialize simulation parameters
n_sims = 1000 # number of times to simulate through all years
brackets = [b19]
total_sims = (n_sims * len(brackets))
scores = []
correct_games = []

# Loop through a plethora of brackets
for i in range(n_sims):
    for bracket in brackets:

        # Run the algorithm on the bracket
        bracket.score(sim_func=pick_a_random_team, verbose=False)

        # Save the scoring results in a list
        scores.append(bracket.total_score)
        correct_games.append(bracket.n_games_correct)

# Calculate the average across all simulations
avg_score = round(sum(scores) / total_sims)
avg_correct = round(sum(correct_games) / total_sims)

# Print result
print(f"Average number total score {avg_score}/192")
print(f"Average number of games guessed correctly {avg_correct}/64")
```

```
Average number total score 30/192
Average number of games guessed correctly 21/64
```

Figure 4: Bracketology Example. This figure shows example code from the bracketology documentation. (Kim & Mente, 2022).

After examining the internals of the methods in the package and how they fit together to form the overall March Madness bracket solver, we supplied the models we created for each team to the package to have a better evaluation function to predict the outcome of individual games. We created two methods, named choose_team and toss_up to determine the outcome of a game and supply the result to the bracketology package to fill out a March Madness bracket for each round in the competition.

The choose_team function takes in the names of the two teams playing against each other in a game and runs the logistic regression model for the first team with the second team as the

opponent, and the model for the second team with the first team as an opponent. If only the first team's model or the second team's model returns a 1, then that team is predicted to have won the game. If both teams' models return either a 0 or 1, then this means that the game could be won by either team, and the toss_up function is called internally. Figure 5 displays the logic for the choose_team function.

```
if trial_1[0]==0 and trial_2[0]==1:
  print(team_one, " won!")
  return team_one
elif trial_2[0]==0 and trial_1[0]==1:
  print(team_two, " won!")
  return team_two
elif trial_1[0]==0 and trial_2[0]==0 or trial_1[0]==1 and trial_2[0]==1:
  print("This is a tossup")
  tossup_result = toss_up(team_one,team_two)
```

Figure 5: Choose Team Logic. This figure shows the logic behind how a team is picked. (Kim & Mente, 2022).

The toss_up function is called when the outcome of a game between two teams is unable to be determined by the models created, meaning that both models return that their respective team will win or lose the game. The toss_up function then creates models for each team in which it only considers the six most important factors in the model building process. We determined this by extracting the importance of each feature and the results are shown below in Figure 6.

```
for i,v in enumerate(importance_list):
  print('Feature: %0d, Score: %.5f' % (i,v))
```

```
Feature: 0, Score: -25.13313
Feature: 1, Score: 41.07369
Feature: 2, Score: -19.76595
Feature: 3, Score: -13.26620
Feature: 4, Score: 37.32968
Feature: 5, Score: -4.08651
Feature: 6, Score: 14.56971
Feature: 7, Score: 7.46394
Feature: 8, Score: 7.62890
Feature: 9, Score: 5.85040
Feature: 10, Score: -28.48853
Feature: 11, Score: 2.83601
Feature: 12, Score: 1.00098
Feature: 13, Score: 1.62019
Feature: 14, Score: 1.91999
Feature: 15, Score: 2.84510
Feature: 16, Score: 1.30950
Feature: 17, Score: 1.71680
Feature: 18, Score: 2.24447
Feature: 19, Score: 0.10174
Feature: 20, Score: 1.52894
Feature: 21, Score: 4.05288
```

Figure 6: Importance List. This figure shows the coefficients for the features. (Kim & Mente, 2022).

The new models are then run and if they still provide the same answer, this means that both teams are evenly matched and either could win the game, so one of the two teams is chosen at random and predicted to win the game. The internals of the toss_up function are provided in Figure 7 below.

```python
def toss_up(team_one, team_two):

  dataframe_1 = final_dict_schedules[team_one]
  dataframe_2 = final_dict_schedules[team_two]

  dataframe_1_select = dataframe_1
  dataframe_2_select = dataframe_2
  new_column_names_1  = []
  for names in dataframe_1_select.columns:
    if names == "Final Point Differential":
      new_column_names_1.append(names)
    else:
      names = names.replace("tm","Opp")
      new_column_names_1.append(names)

  dataframe_1_select.columns = new_column_names_1
  new_column_names_2  = []
  for names in dataframe_2_select.columns:
    if names == "Final Point Differential":
      new_column_names_2.append(names)
    else:
      new_column_names_2.append(names)
  dataframe_2_select.columns = new_column_names_2
  d_1 = dataframe_1_select.join(dataframe_2_select, on="Final Point Differential", rsuffix="Other ")
  column_means = d_1.mean()
  d_1 = d_1.fillna(column_means)
  d_1 = d_1.drop(['Final Point DifferentialOther ','Final Point Differential'],axis=1)
  trial_1 = perform_special_log_prediction(final_dict_schedules[team_two],d_1)


  dataframe_1 = final_dict_schedules[team_two]
  dataframe_2 = final_dict_schedules[team_one]

  dataframe_1_select = dataframe_1
  dataframe_2_select = dataframe_2

  new_column_names_1  = []
  for names in dataframe_1_select.columns:
    if names == "Final Point Differential":
      new_column_names_1.append(names)
    else:
      names = names.replace("tm","Opp")
      new_column_names_1.append(names)

  dataframe_1_select.columns = new_column_names_1
  new_column_names_2  = []
  for names in dataframe_2_select.columns:
    if names == "Final Point Differential":
      new_column_names_2.append(names)
    else:
      new_column_names_2.append(names)
  dataframe_2_select.columns = new_column_names_2
  d_2 = dataframe_1_select.join(dataframe_2_select, on="Final Point Differential", rsuffix="Other ")
  column_means = d_2.mean()
  d_2 = d_2.fillna(column_means)
  d_2 = d_2.drop(['Final Point DifferentialOther ','Final Point Differential'],axis=1)
  trial_2 = perform_special_log_prediction(final_dict_schedules[team_one],d_2)

  if trial_1[0]==0 and trial_2[0]==1:
    print(team_one, " won!")
    return team_one
  elif trial_2[0]==0 and trial_1[0]==1:
    print(team_two, " won!")
    return team_two
  elif trial_1[0]==0 and trial_2[0]==0 or trial_1[0]==1 and trial_2[0]==1:
    print("This is a double tossup")
    winner = random.randint(0, 1)
    if winner==0:
      print(team_one, " won!")
      return team_one
    else:
      print(team_two, " won!")
      return team_two
```

Figure 7: Toss-up Code. This figure shows the code for the toss-up function. (Kim & Mente, 2022).

To complete the testing of our models and March Madness predictor, we ran the models and filled out the bracket for the 2022 March Madness season and compared the results we obtained to the actual outcome of the NCAA tournament.

RESULTS

RESULTS OF INITIAL MODELS

After creating the models based off of the UVA dataset, we calculated the error of the predictions based on the model training, and these results are discussed below.

For the linear regression model, we obtained a linear RMSE of roughly 1.4552158858602253e-14, which shows that the model was able to predict the point differentials between UVA and other schools, and by extension who won the game, with high accuracy on the testing dataset. The graph in Figure 8 below is a scatterplot showing how there is little difference in the predicted point differentials and the true point differentials for the games in the testing data.

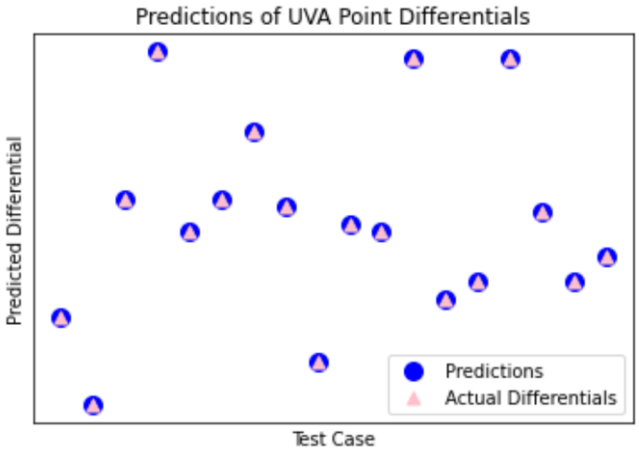


Figure 8: Point Differentials. This figure shows the difference in point differentials. (Kim & Mente, 2022).

The value for the error we obtained for the decision tree regressor was 0.0, which is suggestive of overfitting of the data, so we decided to use cross-fold validation to get a more accurate representation of the true error of the model. Using this method we received an array of the following values 14.76482306, 17.20465053, 14.86606875, 9.21954446, 9.5131488, 2.91547595, 12.74754878, 1.58113883, 3, and 17. From this, it is evident that the model is not as powerful as was initially thought, and suggests that decision tree regression may not be the best model for the defined problem.

We then evaluated the error for the random forest regression model, which we got as 4.178825592595764. Using cross-fold validation, we got 9.37599716, 10.93255948, 2.34927929, 1.76126142, 7.00490485, 6.13798938, 2.22110384, 7.10704711, 3.381, and 1.8764. From this, it seems as though the random forest RMSE is a good indicator of the true error of the model.

To determine the error of the logistic regression model, we used the log-loss function and got an error of 9.992007221626415e-16, which is a very low value and indicative that the model is powerful.

RESULTS OF EXPANSION FROM ACC DATASET

Figure 9 below shows how each college's RMSE from the linear model before adding the extra features to after adding them decreased compared to the previous run.

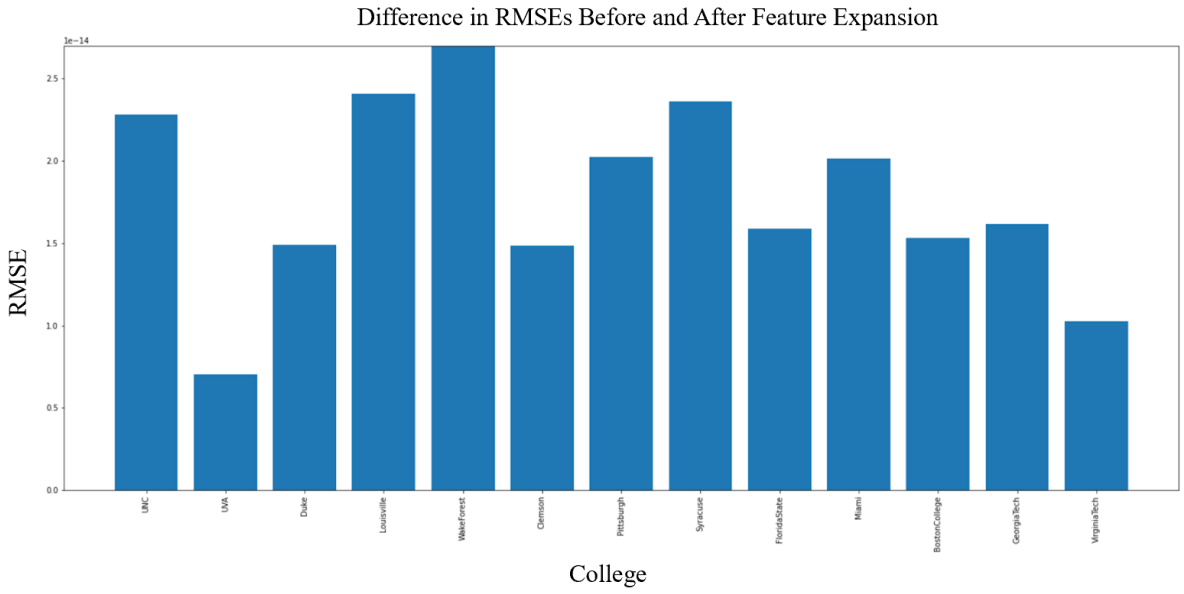


Figure 9: RMSE Differences. This figure shows the range of basketball features used to build the models. (Kim & Mente, 2022).

Additionally, the graph in Figure 10 below shows the difference in the predicted point differentials and the true point differentials for the games in the extended testing data for the different schools and games.

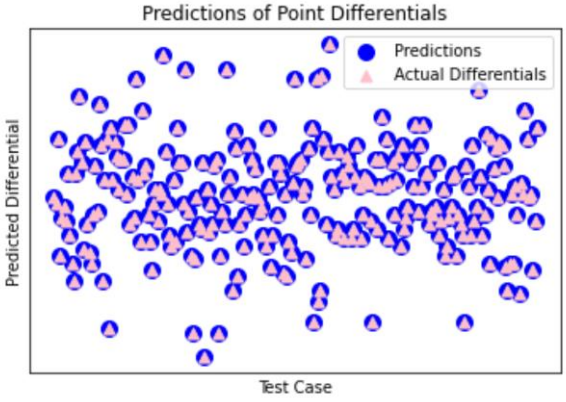


Figure 10: Expanded Point Differentials. This figure shows the expanded point differentials for more schools. (Kim & Mente, 2022).

Similar to the model building process for the UVA dataset, we then trained a decision tree regressor, which yielded an average tree RMSE of 5.824087911424415.

Then we trained a random forest regression model which resulted in the following random tree RMSE with the average being 4.179457850105979.

The last regression model we trained and fit with the expanded dataset was a logistic regression model, and used the log-loss function to obtain the error, which produced the following errors: 9.992007221626413e-16, 9.992007221626415e-16, 9.992007221626415e-16, 9.992007221626413e-16, 9.992007221626415e-16, 9.992007221626415e-16, 9.992007221626415e-16, 9.992007221626413e-16, 9.992007221626415e-16, 9.992007221626415e-16, 9.992007221626413e-16, 9.992007221626413e-16, 9.992007221626415e-16, 9.992007221626413e-16. This suggests that the model was able to provide highly accurate predictions for the data varying from school to school.

RESULTS AFTER BIAS REMOVAL

Figure 11 below presents a graph containing the difference in the performance and behavior solely for the linear model prior to removing the features causing overfitting and afterwards for each of the colleges participating in March Madness. Before we expanded the dataset and removed the features causing overfitting there was no visible difference between the predictions of the point differential and the true value of the point differential for a given game with respect to a particular college. After making the modifications there was great variety and difference in the predictions versus the true outcome of the games. This was true for each of the types of models trained.

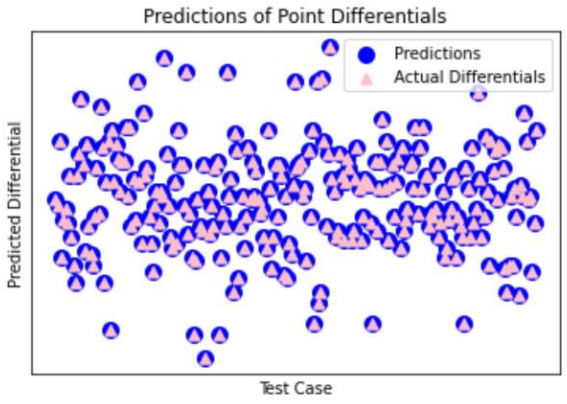
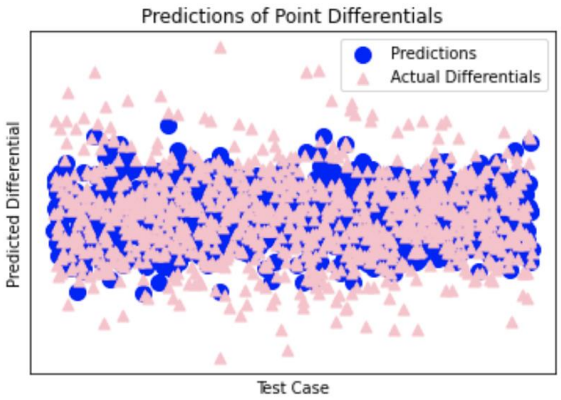
Pre-Expansion + Overfitting features	Expansion and Overfitting features removed
	

Figure 11: Graph of removed overfitting. This figure shows the difference in the point differential predictions once the overfitting had been reduced. (Kim & Mente, 2022).

After the "Tm" and "Opp" features were removed and new teams were added to the overall data frames list, we obtained the following average errors (shown in Figure 12) for each of the different models trained on data from all 63 colleges.

Model	Avg. Error
Linear Regression	1176659071735.2527
Decision Tree	19.101394692872628
Random Forest	14.32164923730064
Logistic Regression	0.596255892458005

Figure 12: Final Errors of the model. This figure shows the average error of the models tested. (Kim & Mente, 2022).

FINAL RESULTS

After creating the final models and integrating them with python's bracketology package, the March Madness bracket for the 2022 season was filled out, and the predicted results were compared to the actual results. Our March Madness bracket solver correctly predicted the outcome of 20 out of the 32 games of the first round, resulting in an accuracy of 62.5%. Below is Figure 13 depicting the model running in this year's tournament.

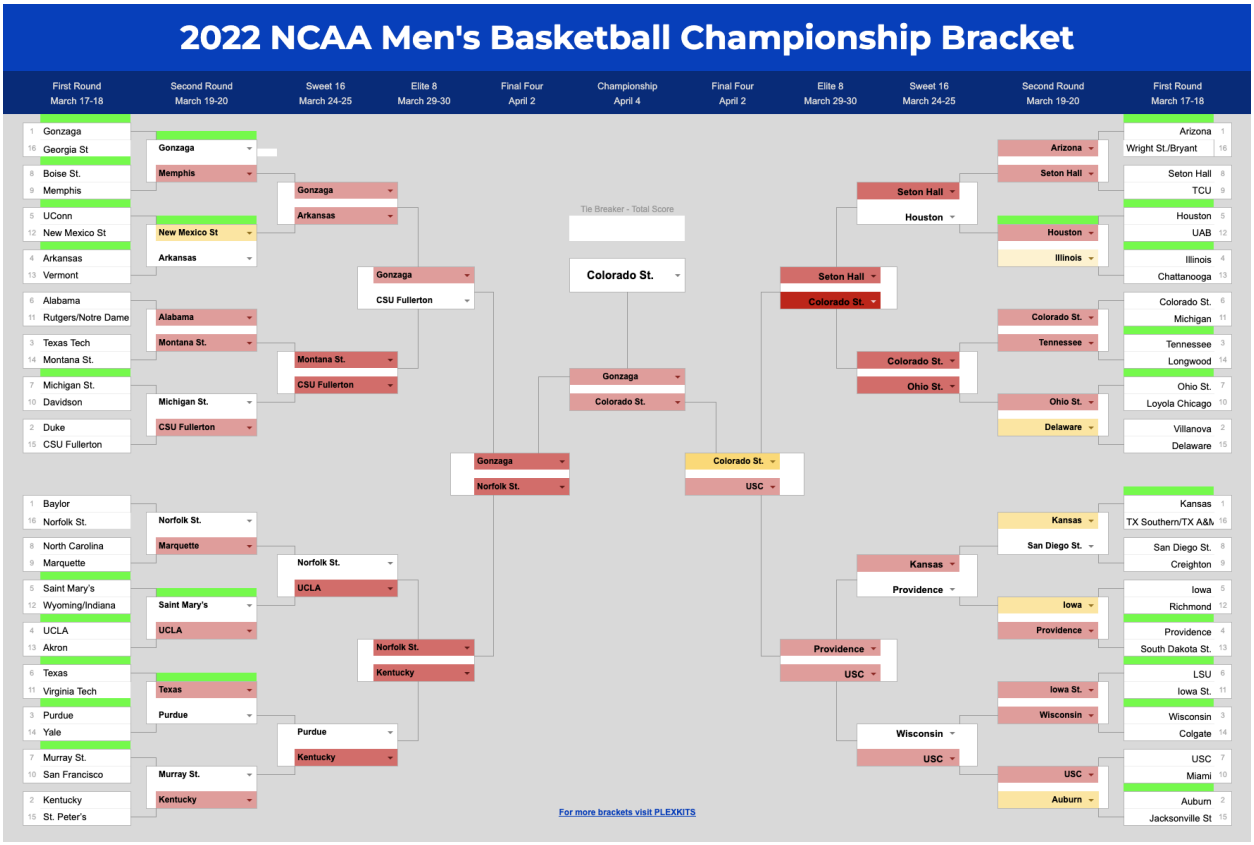


Figure 13: 2022 March Madness Bracket. This figure shows the bracket that the logistic regression model would have outputted. (Kim & Mente, 2022).

This is a lower threshold than expected and means that the models have to be tuned further to yield better results. Regardless, the implications of the process used to build the

models and the March Madness predictor are significant and can be used to aid in predictive modeling for other sports and in the sports betting industry.

SYNTHESIS AND NEXT STEPS

The model building process was an iterative process in which we had to establish a proof of concept that the models were viable. We started by creating models for one school, before continuing to create models for every college participating in March Madness. We then determined that the data were biased and that overfitting was happening, so we removed the problematic features and created final models for each of the schools. Then we verified the accuracy of the models by creating a March Madness predictor, which yielded an accuracy of __. To create better models in the future for this task, we could expand the datasets further to include data regarding each player on the team, since this varies from season to season and greatly influences the performance of each team. We could also explore more models including neural networks and other classification algorithms to determine if there are models better suited for this problem than logistic regression.

REFERENCES

Korpar, L. (2022, March). March Madness betting expected to exceed $3 billion, set all-time high. *Newsweek*. https://www.newsweek.com/march-madness-betting-expected-exceed-3-billion-set-all-time-high-1687917

Kim, J. & Mente, S. (2022). *Bracketology Example*. [4]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Choose Team Logic*. [5]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Expanded Point Differentials.* [10]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Final Errors of the model*. [12]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Four Factors*. [1]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Formulas for Features*. [2]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Graph of removed overfitting*. [11]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Importance List*. [6]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Louisville Dataframe*. [3]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *2022 March Madness Bracket*. [13]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *Point Differentials*. [8]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022). *RMSE Differences*. [9]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Kim, J. & Mente, S. (2022).*Toss-up Code*. [7]. *Thesis* (Unpublished undergraduate thesis). School of Engineering and Applied Science, University of Virginia. Charlottesville, VA

Levandoski, A., & Lobo, J. (2017). *Predicting the NCAA men's basketball tournament with machine learning*. Jonathan Lobo. http://jonathanlobo.com/docs/predicting_mm.pdf

Moorthy, S., Shi, Z., & Zimmermann, A. (2013). Predicting NCAAB match outcomes using ML techniques – some results and lessons learned. *arXiv Cornell University*. *1(1310)*. https://arxiv.org/pdf/1310.3607.pdf

PyPI. (2020, March). *bracketology*. https://pypi.org/project/bracketology/