Applying Machine Learning to NCAA March Madness

Herambeshwar Pendyala, Srinath Botsa

cs795/895 - Practical Machine Learning | Course project | Old Dominion University

01130541, hpend001@odu.edu

01129719, sbots001@odu.edu

***Abstract*— This document serves as a project report for course cs795/895 Practical Machine Learning. The problem is from kaggle, the solution is given using a machine learning approach using various algorithms like Logistic regression and random forest and see which performs better in a given scenario along with the scores submitted in the kaggle to check our positions in the leaderboard.**

***Keywords*— March Madness, NCAA Basketball, bracket, Kaggle, Machine Learning, Logistic Regression, Random forest, ELO score, feature selection.**

I. Introduction

March Madness refers to the annual collegiate men's basketball tournament. The tournament is made up of 64 college teams competing in a single elimination format. In order to win the championship, a team has to win 6 consecutive games. As a result of the continued collaboration between Google Cloud and the NCAA®, Kaggle organizes the March Madness® competition to predict the winners of the tournament. As part of the course Practical Machine Learning, we were given this Kaggle problem to work on as a project. This proposed solution uses Machine Learning methods taught in the class to find the optimum path, These methods are discussed in detail in the following sections.

II. Problem Statement

The tournament is broken up into 4 regions. Each region has 16 teams, ranked from 1 to 16. This ranking is determined by a NCAA committee and is based on each team's regular season performance. The NCAA structures the games so that the highest seed in a region plays the lowest seed, the 2nd highest plays the 2^{nd} lowest, and so on.

Those are the odds that you will correctly pick the winners of all 63 games played over the course of the tournament. Mathematically speaking, there are 2^{63} (~ 9.2 quintillion) number of ways that you can fill the bracket. In 2014, Warren Buffet famously offered 1 billion dollars to anyone who could fill out a perfect bracket (Needless to say, nobody even really came close).

We want to see if we can build a Machine Learning model that is able to look at training data (past NCAA basketball games), find the relationship between a team's success and their attributes (stats), and output predictions for future games.

So, why can machine learning be a possible use case for this prediction problem? First of all, we have the data in place. Over 100,000 NCAA regular season games were played over the last 25 years, and we generally have plenty of statistics about the teams for each season. Because we have all of this data, we can try to use machine learning to find out what particular statistics most correlate with a team winning a particular matchup.

The given problem looks more like a classification problem where we are given with two team vectors and we need to predict which team has higher probability to win. Our approach to solve the problem is to use the machine learning algorithms taught to us in the class to predict the winning probability of the team.

a. Logistic Regression
b. Random Forest

III. Dataset

We are provided Season data, In each season there are thousands of NCAA basketball games played between Division I men's teams, culminating in March Madness, the 68-team national championship that starts in the middle of March. We have provided a large amount of historical data about college basketball games and teams, going back many years. Below is the image showing how the regular season data looks like.

	Season	Daynum	Wteam	Wscore	Lteam	Lscore	Wloc	Numot
0	1985	20	1228	81	1328	64	N	0
1	1985	25	1106	77	1354	70	H	0
2	1985	25	1112	63	1223	56	H	0
3	1985	25	1165	70	1432	54	H	0
4	1985	25	1192	86	1447	74	H	0

Fig. 1 First 5 rows in the Regular Season Data

Apart from the season data, we have Seasons, Teams, Tournament Seeds, Tournament Slots Regular Season and Tournament data with team related details like team scores of both winning and losing teams. Other files include detailed results of box scores like field goals, three pointers, free throws, rebounds, assists, turnovers, steals, blocks, personal faults. Using these details we can analyze the patterns and use those attributes in our predictions.

IV. Visualizations

As we are approaching a problem, it is always good we start with a proper visualization of the data. as this gives us more insights about the data. At the very beginning we plot the given data to see what it looks like, below is the image of teams with most season wins.

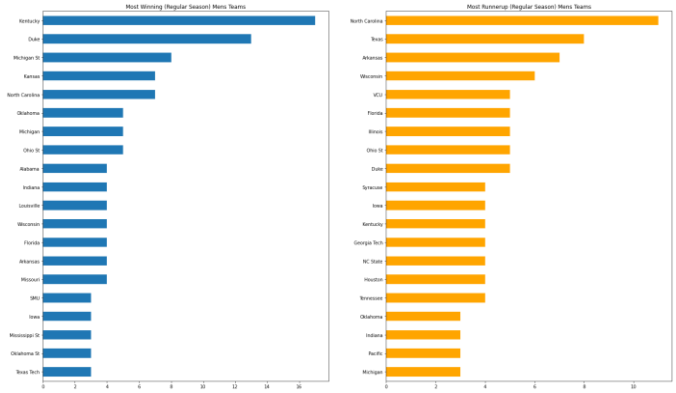


Fig. 2 visualization of the most season wins and runner ups

So it looks like Kentucky, Duke, Michigan St, Kansas, North Carolina have won most seasons from 1985 to 2019. Also on the right side we have North Carolina, Texas, Arkansas with most runner ups. This could possibly be a feature as we have seen teams with most season wins have a good chance of winning the tournament as well.

As we have seen how visualization helps us to see what is there in data, We will be doing more visualizations in the next sections of the paper as well.

V. Methods

To start with solving the problem we would like to build a machine learning model that takes information about two teams (Team 1 and Team 2) as input, and then outputs a probability of Team 1 winning that matchup.

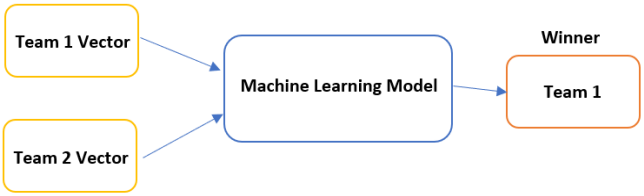


Fig. 3 Proposed machine learning model

The next step in the flow would be selecting the feature vectors that represent information on each team for two teams. To select the features we would be doing exploratory data analysis, followed by feature engineering to decide which features are to be selected and used for modelling, Then we use those features and Train our machine learning model. We use the trained model to predict the test set and see the accuracy of the model, if we find the accuracy to be good we will use the model to make the predictions for all the possible match ups of seasons 2017 and 2019.

A. Exploratory Data Analysis

We first start with exploring the season and tournament data to get insights about the data, below figure shows team wins(left blue) and runner ups(Right orange) in tournaments.

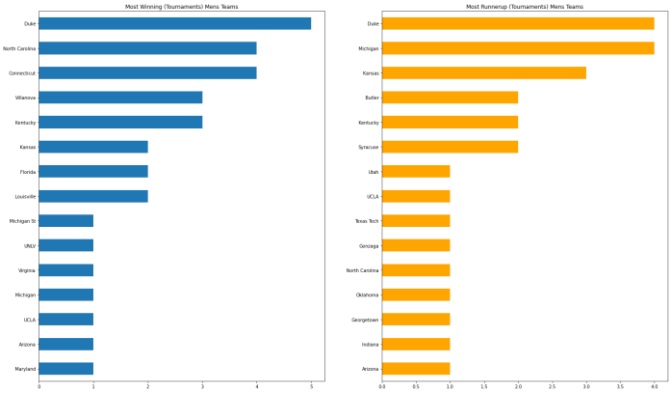


Fig. 4 Most tournament wins and runner ups

By seeing the tournament and season winner and runner up stats we thought this might be a good feature to include Previous season/tournament Wins, loss, and also number of tournament appearances as not every team in 362 makes into a tournament of 64 teams. The next feature we looked into was season scores.

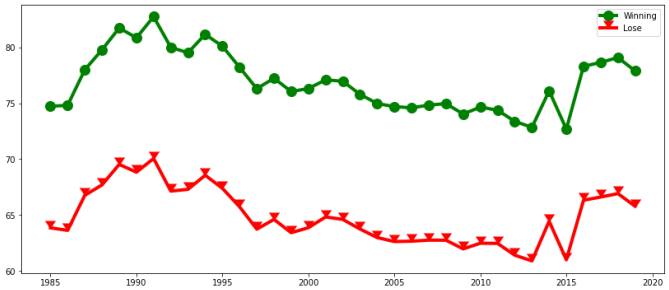


Fig. 5 Season wise mean scores of winning and losing teams

From the above figure we can see that for every season the gap between the winning team score and losing team score seems to be steady, in other words nearly the same or uniformly distributed, so we decided to take the mean of scores of both winning and losing teams for past 10 seasons. As we have seen that previous seasons and tournaments play an important role we have decided to take the box scores like field goals, three pointers, free throws, rebounds, assists, turnovers, steals, blocks, personal faults to be considered as features for teams. We will be looking more into them in the coming sections.

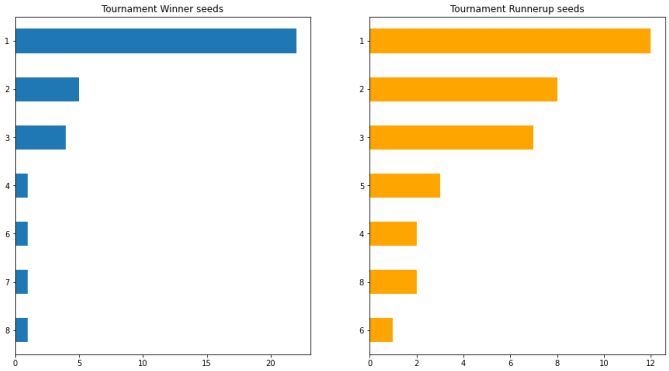


Fig. 6 Seeds of top winning and runner up teams

One more feature we thought would be important was the seed difference because we have seen that teams with less seed numbers tend to be the winner of most of the matches.

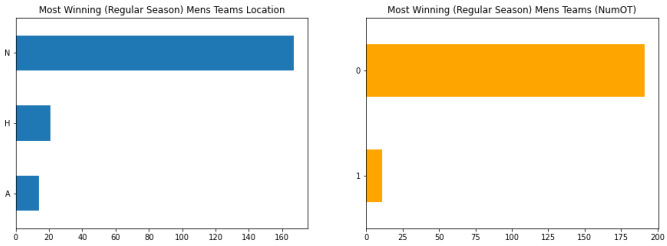


Fig. 7 Team's location and NumOT plots of season and tournament games

Though we were doing a lot of exploratory data analysis we did not consider every feature from the data. Above image shows two plots, one on the left is the team's location for the match, we have seen that the features are not distributed evenly as most of the team's location for a match is N(Neutral), we can see that the Location does not play a crucial role in winning or losing. The right plot in orange shows the number of overtime in a game. from the plot itself it is clear that as most of the teams have 0 overtime we thought even this feature would not be useful in a team vector. So we discarded these features from our feature vector.

B. Elo Score

As we have seen from our analysis that the box score features alone are not so good in getting a better prediction, so we have used an algorithm called the Elo Ranking Algorithm, This Algorithm is a widely used rating algorithm to rank players in many competitive games.

Players with higher ELO rating have a higher probability of winning a game than a player with lower ELO rating. After each game, the ELO rating of players is updated. If a player with higher ELO rating wins, only a few points are transferred from the lower rated player. However if a lower rated player wins, then transferred points from a higher rated player are far greater.

Elo system consists of two parts:

When Player A competes in a match against Player B, Player A has an expected outcome (probability or score) for team A (EA). Where $R_A$is Rating for team A and R_B is Rating for team B, The expected outcome for team A (EA) can be calculated by the formula below:

$$EA = \frac{1}{1 + 10^{(R_B - R_A)/400}}$$

The same calculation (EB) has to be done for Player B, but with RA (current rating A) and RB (current rating B) swapped so that EA + EB = 1. Once the match is played and SA (actual outcome or score for team A), and SB (actual outcome or score for team B) are determined, R' A (New rating for A) and R' B (New rating for A) are calculated with the formula below:

$$R'A = R_A + K(SA - EA)$$

Where K is an optimization constant that usually takes different values according to the sport and the amount of games available. For the values of S we will stay with the classic definition of win-loss-tie and we will ignore the points made so that: Win = 1, Loss = 0, Tie = ½.

Below is the pseudocode for a function to calculate the elo score

```
def calculateELO(winRank, loseRank, Season):
    rankDiff = winRank - loseRank
    exp = (rankDiff * -1) / 400
    odds = 1 / (1 + math.pow(10, exp))
    if winRank < 2100:
        k = 32
    elif winRank >= 2100 and winRank < 2400:
        k = 24
    else:
        k = 16
  winRankNew = round(winRank  + (k * (1 - odds)))
  newRankDiff = winRankNew - winRank
  loseRankNew = loseRank- newRankDiff
```

For this problem we have created a dictionary in python that stores the elo score of the teams for every game. We calculate the elo score for each team by setting a base score and this would be the hyper parameter that we tweak to get better results.

C. Feature Engineering

During exploratory data analysis, we assumed box scores could be a good feature, so we plotted correlation between the box score like field goals, free throws, 3 pointers made and attempted. From the above image of correlation heatmap, we can see that all these box scores are correlated, so we thought of calculating the percentage of these box scores, this gives us three more features, field goal percentage, 3 pointer percentage, free throw percentage. This brings our team features to 20 including elo score.

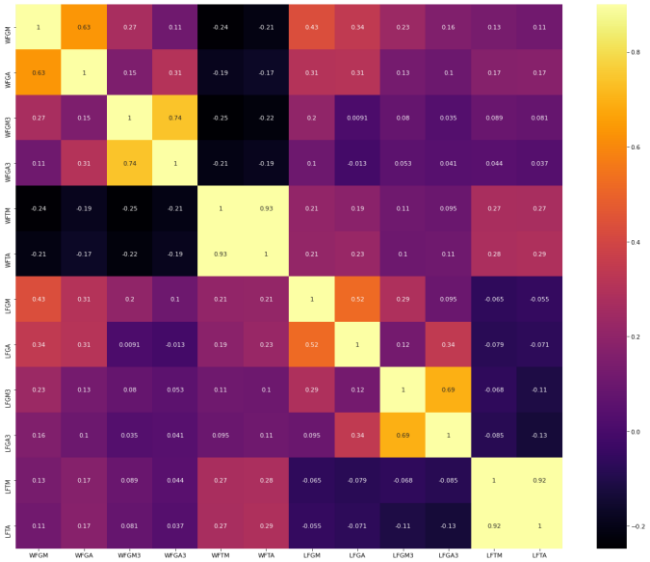


Fig. 8 Correlation of box scores of winning and losing teams

Similarly, we thought of calculating correlation of every feature with the result to see how much they are correlated. We started with seed difference and plotted their correlation, below image shows the same. We were surprised to see the correlation came out to be -0.5 which is less than 0. We know anything less than 0 is not considered a good feature as they are not really well correlated to the final result. so we thought of discarding this feature of seed difference. This brings our features to 19 for each team.

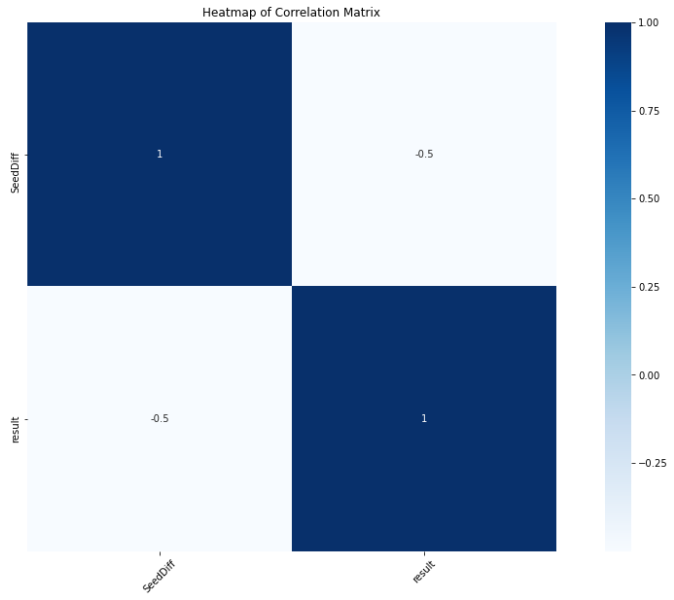


Fig. 9 Correlation of seed difference and match result.

As we have seen how correlation can help us find better features, we plotted correlation of every feature and the correlation above 0, So we have decided to use all the features.

D. Creating the Final Data Set

Now that we have decided on features, we are left with creating the final dataset. For creating the final dataset we consider all the previous season and tournament data and feed it to a custom function that outputs our final training data. Below is the pseudocode for that function.

```
def buildSeasonData(allData):
   for every row in the dataset:
       winElo = calculate_elo for winning team
       lossElo = calculate_elo for losing team
       winfeatures = getallFeatures()
       lossfeatures = getallFeatures()

finalData.append(winfeatures+lossfeatures)
   return finalData
```

In the above function we iterate over each row and calculate and get all the features append it to an array. Here we append both the team's features into one array and create the new dataset. Estimated time to create the dataset using this function is 1 hour.

E. Machine Learning Models

Now that we have all the data in place, It is time for us to build some machine learning models. As we are dealing with classification problem, we settled on using the below models for this problem.

1. Logistic Regression

For logistic regression model, we used basic configuration that is offered by the scikit learn package, This gives us LBFGS solver with l2 regularizer, maximum iterations of 100 are set by default. We will be using 10 fold cross validation while training to get the best model.

2. Random Forest

A random forest is a meta estimator that fits a number of decision tree classifiers on various sub-samples of the dataset and uses averaging to improve the predictive accuracy and control over-fitting. The sub-sample size is always the same as the original input sample size but the samples are drawn with replacement using bootstrapping.

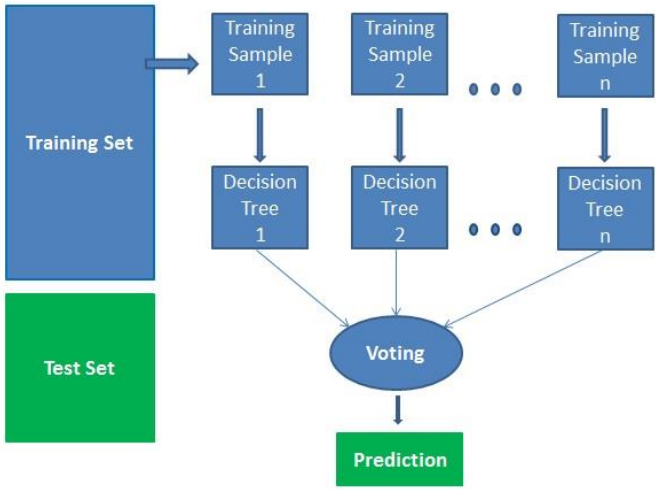


Fig. 10 Working of Random Forest

For Random forest the hyper parameters we use include 200 trees, gini as info gain criteria.

F. Pseudo Code of Implementation

For training and testing the model we created a function that trains and returns a model.

```
def trainData(algorithm, Data):
   if algorithm == 'Logistic':
      model = logisticregression.fit(Data)
      return model
   elif algorithm == 'Random Forest':
      model = randomforest.fit(Data)
      return model
```

G. Final Features for Predicting the winner

As we are done with feature engineering we have left with 19 features for each team that include previous Season Wins, previous Season loss, previous Tournament Appearances, previous Tournament Wins, previous Tournament Loss, EloScore, mean of all the box scores that include field goals, three pointers, free throws, rebounds, assists, turnovers, steals, meanblocks, personal faults. The below image depicts all the features of both the teams.

```
array(['WprevSeasonWins', 'WprevSeasonLoss', 'WtourAppearances',
       'WprevtourWins', 'WprevtourLoss', 'WELO', 'Wscore', 'Wfga', 'Wfgp',
       'Wfga3', 'W3pp', 'Wftp', 'Wor', 'Wdr', 'Wast', 'Wto', 'Wstl',
       'Wblk', 'Wpf', 'LprevSeasonWins', 'LprevSeasonLoss',
       'LtourAppearances', 'LprevtourWins', 'LprevtourLoss', 'LELO',
       'Lscore', 'Lfga', 'Lfgp', 'Lfga3', 'L3pp', 'Lftp', 'Lor', 'Ldr',
       'Last', 'Lto', 'Lstl', 'Lblk', 'Lpf'], dtype='<U16')
```

Fig. 11 All Features.

VI. Languages and Packages

We used Python as the programming language to solve this problem as it is easy to implement. We used Numpy, and Random for optimizing array operations. We used Pandas to read the dataset and for other operations like data manipulations, creating the datasets, Feature Engineering, Used Matplotlib, Seaborn in Exploratory Data Analysis, Plotting the Results. We have used Scikit Learn for MAchine learning related utilities, Bracketeer library to create the brackets for the seasons.

VII. Experiments and Results

As we have done Exploratory Data Analysis, Feature engineering to bring out useful features, we can now apply those features we have and predict the winning probability of all the matchups for 2017, 2019 tournaments. We first build the training data according to our requirements and save those data in the csv files for later usage, we have built two models for the seasons 2017 and 2019 using the datasets, As in 2019, we got more math data compared to 2017.

For predicting the 2017 season, we have used data upto 2016 to train our model and for 2019, we have used Training data upto 2018. We are using two algorithms, logistic regression and random forest, so we trained our data on 4 models to see which one gives us with better accuracy.

We split our data into training and test sets with 80% as training data and 20% as test data.

First we started with all the box score averages and Previous Season and tournament statistics. Using 10-fold cross validation we got a training accuracy of 68% on logistic regression. Below tables tell us about the performance of both the models for two seasons on the test set.

TABLE 1
Test Performance of Logistic regression on Box Scores and Previous Statistics Data

Season	Accuracy	Precision	Recall
2017	0.678	0.682	0.671
2019	0.680	0.691	0.672

TABLE 2
Test Performance of Random Forest on Box Scores and Previous Statistics Data

Season	Accuracy	Precision	Recall
2017	0.671	0.677	0.660
2019	0.680	0.678	0.679

From the tables It can be inferred that the logistic regression performs better than random forest, but accuracy of 68% is pretty low, and cannot be used as the final model.

As using boxscores was not enough to predict the winner with a decent accuracy, we then added the Elo score as a feature and trained our model once again to see the performance. In 2017, We got a training accuracy of 73% using 10 fold cross validation. From tables we can see that adding an

Elo score has improved our performance and accuracy of the model.

TABLE 3
TEST PERFORMANCE OF LOGISTIC REGRESSION ON BOX SCORES, PREVIOUS STATISTICS DATA AND ELO SCORE

Season	Accuracy	Precision	Recall
2017	0.725	0.728	0.722
2019	0.730	0.739	0.723

TABLE 4
TEST PERFORMANCE OF RANDOM FOREST ON BOX SCORES, PREVIOUS STATISTICS DATA AND ELO SCORE

Season	Accuracy	Precision	Recall
2017	0.718	0.722	0.713
2019	0.723	0.735	0.710

Below confusion Matrix for logistic regression model shows us about the misclassified samples in the test set of 15947 samples.

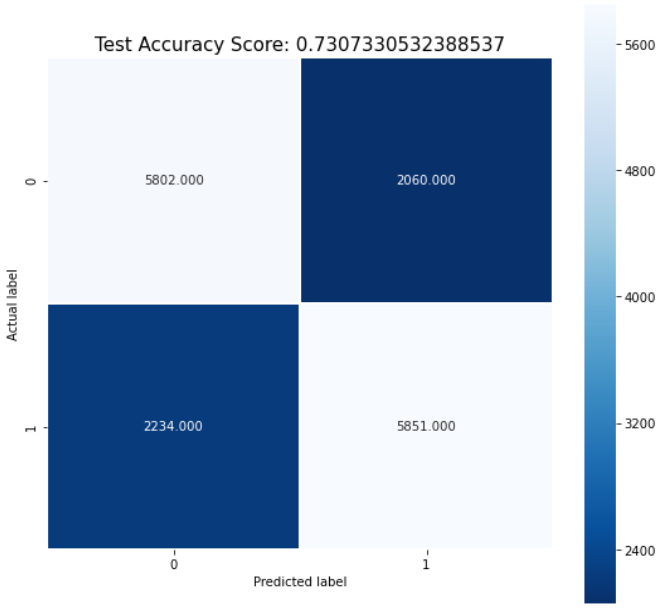


Fig. 12 Confusion Matrix of testSet for season 2017

From our analysis above, both the models perform equally better, but logistic regression performs slightly better than Random forest. Below image displays the feature importance that shows us which features were the deciding factor during the predictions.

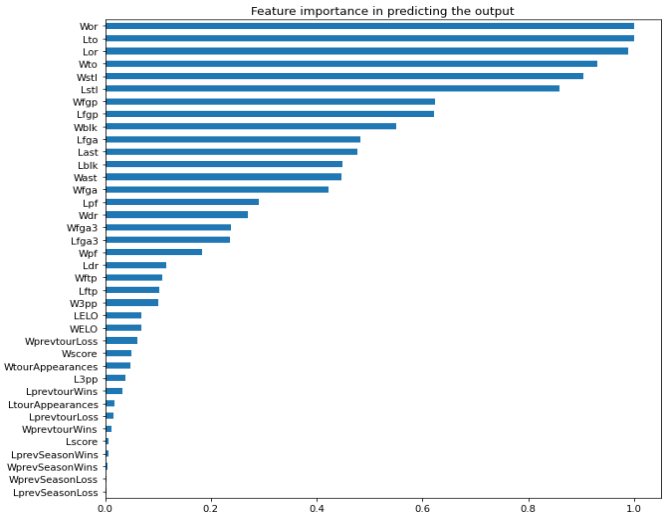


Fig. 13 feature importance for logistic regression, season 2017

For the final submission we have to calculate the probability of winning teams and submit our predictions to kaggle to see which model performs better. In the kaggle logloss is the evaluation metric, so low logloss determines the performance of the model. Analysis of logloss and kaggle submissions are discussed in the next section.

VIII. ANALYSIS

As we have trained and tested the models in the previous section we got a decent accuracy score of 73%. This may not be a good score, But we have used the models to generate the submission files for 2017 and 2019. After generating the submission file, we submitted it and below tables show the results and predicted position if we participated in the competition.

TABLE 5
LOGLOSS & SCORES FOR ALL THE LOGISTIC REGRESSION MODELS

Year	Future Data	ELO Score	Logloss Score	Predicted Position
2017	No	No	0.50753	163/441
2017	No	Yes	0.43510	1/441
2017	Yes	No	0.50729	160/441
2019	No	No	0.67801	712/866
2019	No	Yes	0.46028	88/866
2019	Yes	No	0.63143	702/866

TABLE 6
LOGLOSS & SCORES FOR ALL THE RANDOM FOREST MODELS

Year	Future Data	ELO Score	Logloss Score	Predicted Position
2017	No	No	0.54011	280/441
2017	No	Yes	0.48540	621/441
2017	Yes	No	0.44093	3/441
2019	No	No	0.67915	714/866
2019	No	Yes	0.54153	591/866
2019	Yes	No	0.44815	31/866

From the above tables we can see that in the two algorithms, random forest and logistic regression, the logistic regression model has performed very well by giving us less logloss and also a good predicted position in the leaderboard.

IX. CONCLUSIONS

We have learned about Basketball whose strategies are unknown to us before, We have used the strategies taught to us in class to select features, using feature engineering techniques such as finding correlation between the features and dependent variables. We also got to see how adding features can improve the accuracy of the prediction. In our case we found that ELO score to be a holy grail for use as it has improved the performance of the algorithm in a very good scale. As we have described the problem as a classification problem, we determined the probability of classifying a given team as a loser or winner using Logistic Regression and Random forest algorithms. Out of both the algorithms, Logistic Regression outperformed Random Forest by giving us a less logloss and the predicted position on kaggle to be 1/441 for 2017 and 88/866 for 2019. The positions on kaggle for 2017 and 2019 got a logloss score of 0.43857 and 0.41477. In the bottom of the paper we have included the predicted and true brackets, along with our kaggle scores.

This implementation can be further improved by various methods available. The first spot on kaggle was able to achieve a less log loss score. We can use other state of art methods like Gradient Boosting Trees and Neural networks to get a better log loss and improve our position in the kaggle leaderboard.

ACKNOWLEDGMENT AND FUTURE SCOPE

As we have implemented only a basic logistic regression model using 10 fold cross validation, along with ELO score and other statistical features it gave us a pretty decent log loss and predicted the right tournament winner in the 2017 and right sweet 16 in 2019 brackets.

The other set of approached we wanted to implement were to use Feature Selection and scaling methods like PCA, Ridge regression to scale the features and For prediction we can use modern state of art neural network architectures like GAN as we have a lot of previous data which can help us in training and predicting the right winner for the tournament, but this approach requires a lot more computing power, so we have kept this as a future scope.

REFERENCES

[1] https://www.kaggle.com/c/google-cloud-ncaa-march-madness-2020-division-1-mens-tournament
[2] https://www.kaggle.com/c/mens-machine-learning-competition-2019
[3] https://www.kaggle.com/c/march-machine-learning-mania-2017
[4] https://adeshpande3.github.io/Applying-Machine-Learning-to-March-Madness
[5] https://blog.coast.ai/villanova-will-win-the-2016-march-madness-tournament-according-to-my-machine-learning-model-2f1e4e18037a#.a9xphyenr
[6] https://blog.coast.ai/this-is-how-i-used-machine-learning-to-accurately-predict-villanova-to-win-the-2016-march-madness-ba5c074f1583#.e6xllp64p
[7] https://www.kaggle.com/aashita/feature-engineering-for-march-madness
[8] https://www.zdnet.com/article/students-create-ncaa-march-madness-predictive-analysis-via-google-cloud/
[9] https://www.kaggle.com/parulpandey/decoding-march-madness
[10] https://www.kaggle.com/humburgc/history-eda-machine-learning-march-madness
[11] https://www.kaggle.com/raddar/paris-madness
[12] https://minerkasch.com/machine-learning-the-ncaa-and-you-part-1/
[13] https://www.geeksforgeeks.org/elo-rating-algorithm/

Kaggle Scores

2017

Overview Data Notebooks Discussion Leaderboard Rules Team | My Submissions | Late Submission

Submission and Description	Private Score	Public Score	Use for Final Score
submission-2017_rf02.csv 2 days ago by Heramb Pendyala Random Forest Without Future Data, Including Elo Score	0.48540	0.48540	☐
submission-2017_lr02.csv 2 days ago by Heramb Pendyala Logistic with no future data, including elo score	0.43510	0.43510	☐
submission-2017_rf01.csv 2 days ago by Heramb Pendyala Random forest without Future Data	0.54011	0.54011	☐
submission-2017_lr01.csv 2 days ago by Heramb Pendyala Logistic without future data 01	0.50753	0.50753	☐

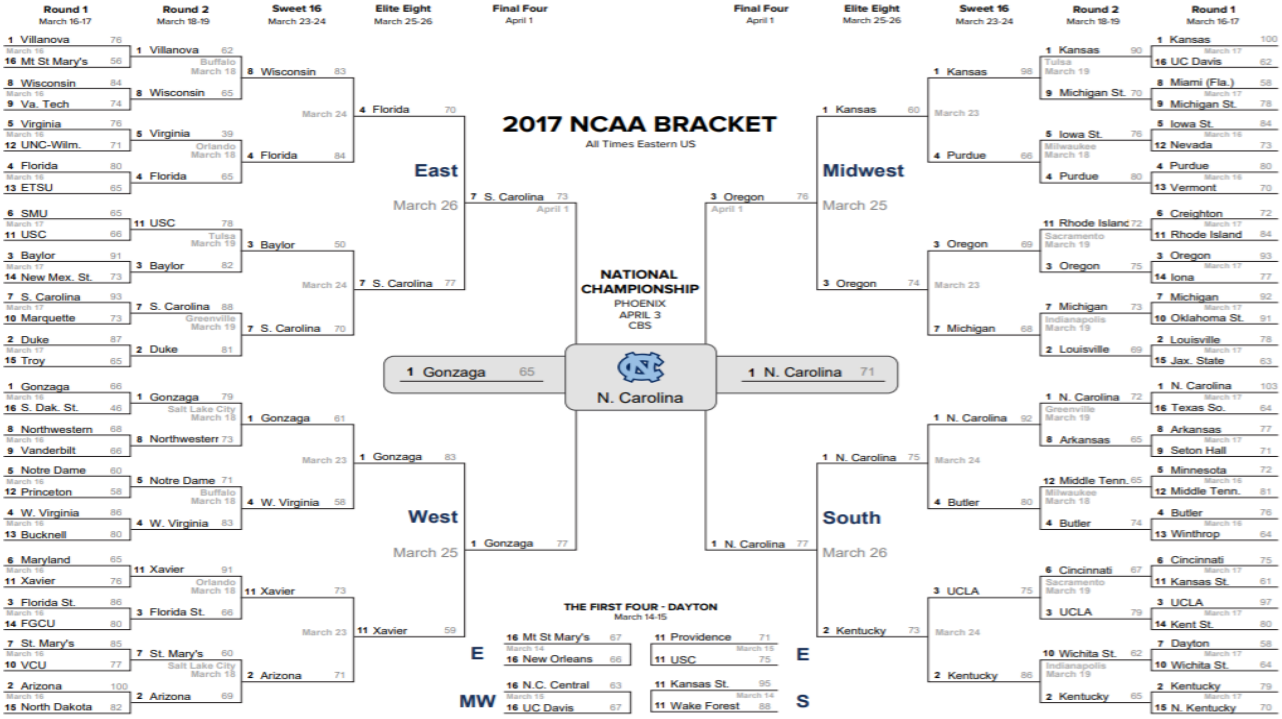
2019

Overview Data Notebooks Discussion Leaderboard Rules Team | My Submissions | Late Submission

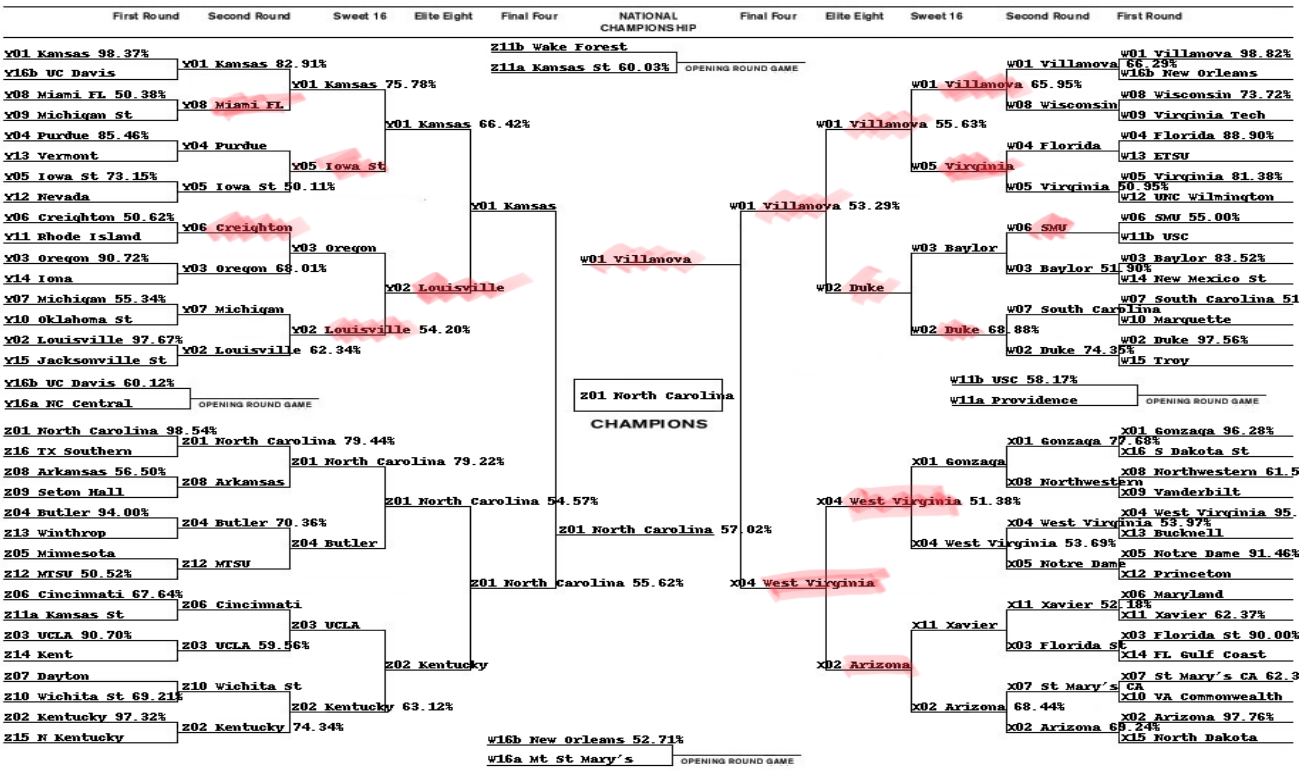
Submission and Description	Private Score	Public Score	Use for Final Score
submission-2019_rf02.csv 2 days ago by Heramb Pendyala Random Forest, without Future Data, Including ELO	0.54153	0.54153	☐
submission-2019_lr02.csv 2 days ago by Heramb Pendyala Logistic without future, including elo	0.46028	0.46028	☐
submission-2019_rf02.csv 2 days ago by Heramb Pendyala random forest without future Data	0.67915	0.67915	☐
submission-2019_lr02.csv 2 days ago by Heramb Pendyala	0.67801	0.67801	☐

Brackets

2017 True

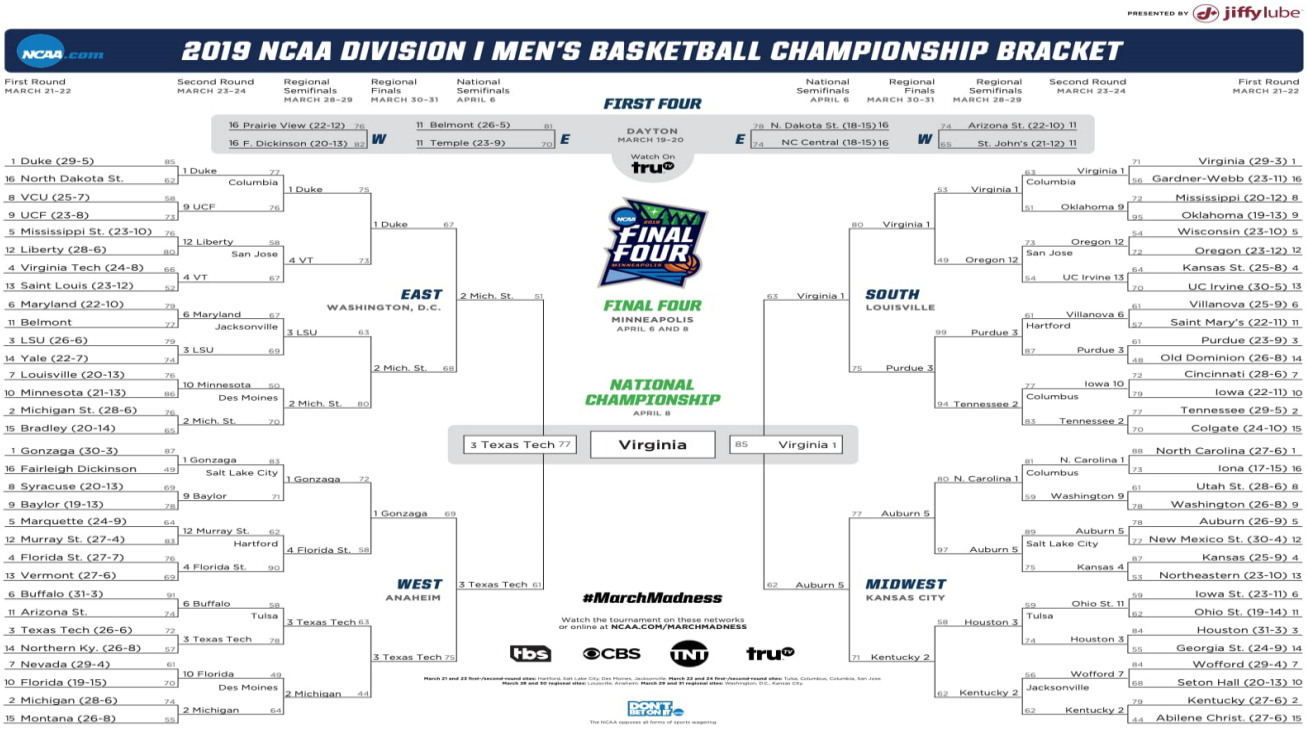


2017 Predicted with upsets

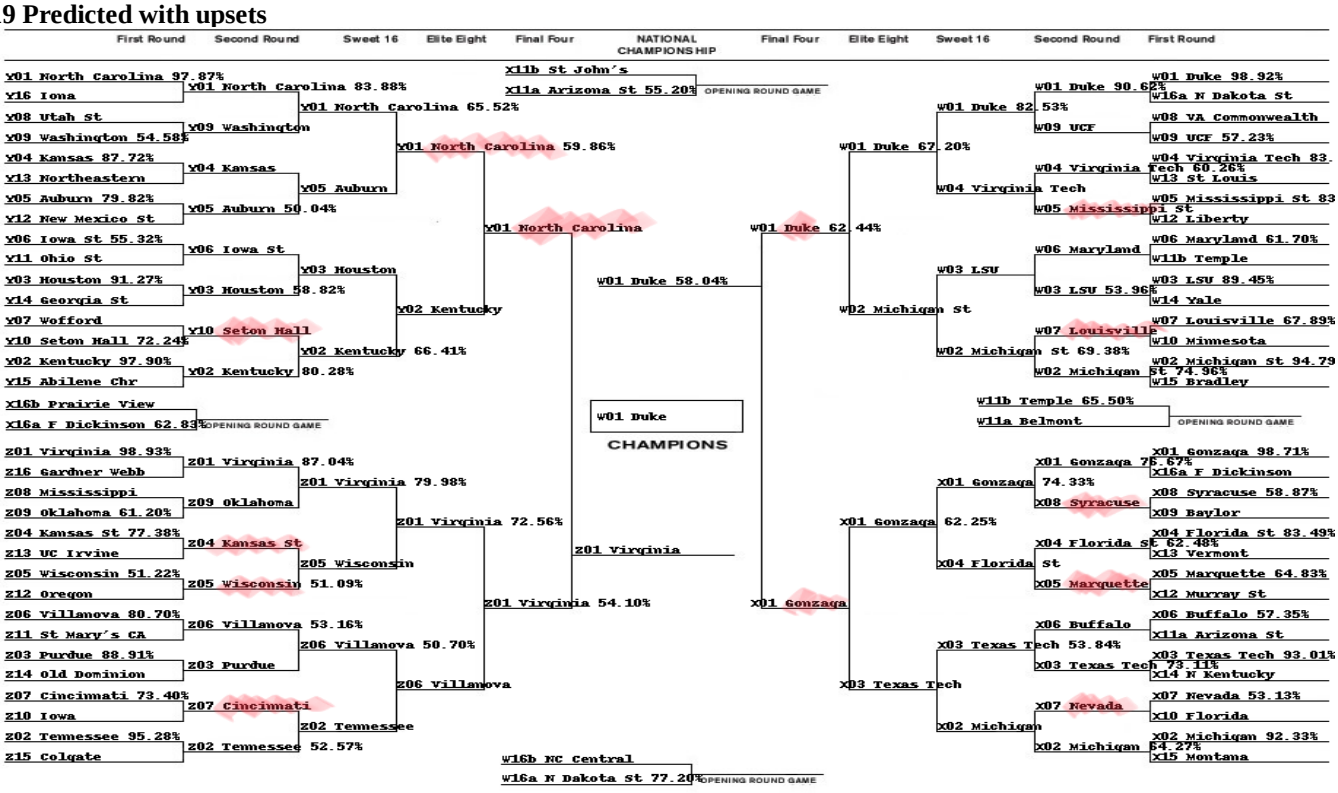


We have predicted the right winner for the 2017 season that is north carolina but we have mispredicted many matches, As the south region has zero mispredictions, we are able to predict north carolina till the final 4.

2019 True



2019 Predicted with upsets



For 2019, We have predicted the right set for sweet 16, but we have mispredicted all other sets. all the mis predictions are marked in red.