# Project Report: Google Cloud & NCAA® ML Competition 2020-NCAAM

CS 795/895 Practical Machine Learning (Spring, 2020)

Apurva Modi
Department of Computer Science
Old Dominion University
amodi002@odu.edu

Kritika Garg
Department of Computer Science
Old Dominion University
kgarg001@odu.edu

***Abstract*— Kaggle March Madness Competition presents a challenge to forecast the outcomes of all possible matchups in the Men's Basketball Championships. In this Report, we describe our approach to predicting the outcomes of the 2017 and 2019 march madness tournament games.**

## I. Problem Definition

The Division I NCAA Men's Basketball Tournament is a popular sporting event held annually to determine the league's National Champion. Over the past several years the betting scene surrounding the tournament has become arguably more popular than the tournament itself, drawing in fans who bet billions overall on its outcome. The Kaggle March madness competition is based on this popular trend. Every year kaggle presents a challenge to correctly predict winners in the march madness tournament using machine learning strategies. Participants use different machine learning based techniques to predict their bracket. The competition is divided into two stages: stage1 and stage2. In stage one, the participants build their model based on the historic data. The stage2 starts after selection sunday in which final 68 teams are announced, the participants then make the predictions to the detailed outcome of that year's final bracket using their constructed models. The predictions submitted to kaggle are scored using a metric called log loss. The log loss provides extreme punishments for being both confident and wrong which means that the more certain your algorithm is (i.e. winning probability close to 0 or 1) the more severe is the penalty if it's wrong.

The March madness competition is a semi-supervised machine learning task. In this work, we present different machine learning algorithms to predict the brackets using historical data for march madness 2017 and 2019. We have used classic machine learning classification algorithms like logistic regression (LR), k-nearest neighbour (KNN) and support vector machines (SVM) to construct our classifier. The classifier classifies the teams into winning teams and losing teams. It provides the winning probability of each match which is then used to predict the brackets.

## II. Dataset Description

The Google Cloud & NCAA® ML Competition 2020-NCAAM contains a large dataset that comprises 5 sections. The basic data section includes the details about the Team, Seasons, Seeds Information, Game Results. The team box scores section provides game-by-game stats at a team level (free throws attempted, defensive rebounds, turnovers, etc.) for all regular season, conference tournament, and NCAA® tournament games since the 2002-03 season.

In this work, we have mainly worked with these two sections. The basic data was used to perform data analysis and also to construct the model using seeds and game results. The team box scores helped in adding detailed features to improve the model. The data used in this work includes:

1. Team Data: Team name and Team ID, first and last D1 Season. Sorting by the FirstD1Season column we can see some of the newest teams in D1 basketball.
2. Seasons Data: These files identify the different seasons included in the historical data, along with certain season-level properties
3. Tourney Seed Data: This file identifies the seeds for all teams for all seasons of historical data.
4. Regular Season Results: These files identify the game-by-game NCAA® tournament results for all seasons of historical data, where:
    a. WScore - the number of points scored by the winning team.
    b. WTeamID - the id number of the team that won the game
    c. LTeamID - the id number of the team that lost the game.
    d. LScore - the number of points scored by the losing team.
5. Tourney Compact Results: This file identifies the game-by-game tournament results for all seasons of historical data.
6. Tourney Detailed Results: This file provides team-level box scores for many NCAA® tournaments, starting with the 2003 season

## III. METHODOLOGY

The Details of our approach is presented in this section. After understanding the datasets provided by the kaggle, We performed exploratory data analysis (EDA) on our selected datasets to acquire in-depth understanding of the dataset. We processed the input data to extract the useful features and then used this data to train our classifiers.

### *A. DATA ANALYSIS*

Exploratory Data Analysis (EDA) is a technique that allows a greater understanding of data through visualization. In our work, we used EDA to understand the importance of seeds in Basketball games. In basketball, seed is given to a team as a preliminary ranking. The seed data provided by kaggle consists of both seed of the team and the region in which the team played. We separated the region from seed, to extract the seed position of each team. We used a python library called pandas to handle our datasets and matplotlib library was used to create the visualizations. Fig1. shows the top 10 teams that have been in the top seeded positions most number of times.

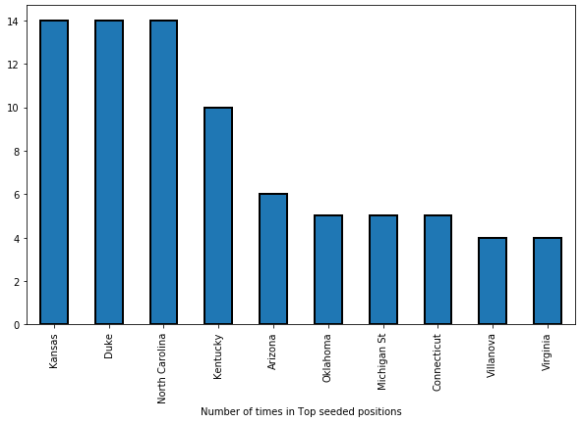


Fig. 1. Top 10 teams that mostly acquire top seeded positions

Similarly, Fig2. shows the top 10 teams that have been in the bottom seeded positions most number of times.

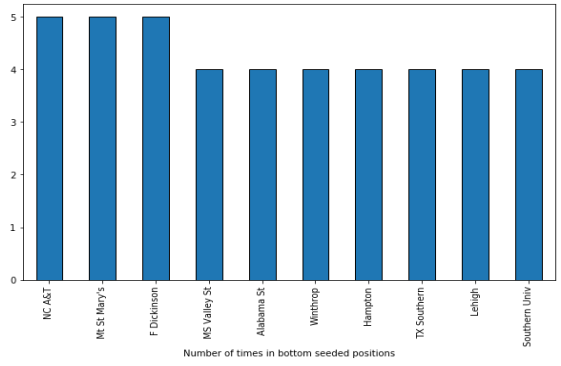


Fig. 2. Top 10 teams that mostly acquire bottom seeded positions

Apart from seeds, score also seems to be an important feature in the march madness. We explored the winning team scores and losing team scores of the regular seasons. Fig. 3. shows the distribution of winning score and losing score from 1985 to 2020. Both scores can be seen following a downward trend till 2013 after which there is a sudden surge in score values.

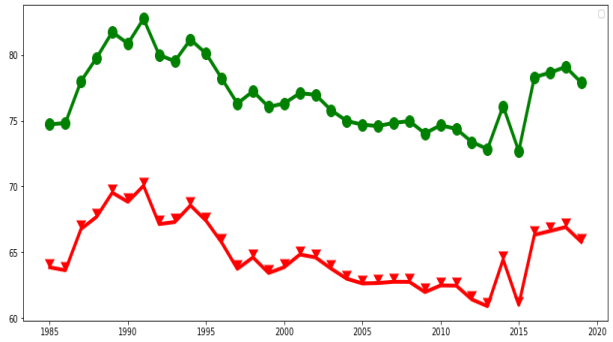


Fig. 3. The analysis of winning scores (green) and losing scores (red) over the years..

### *B. BASELINE MODEL*

We constructed a baseline model based on seed difference, since we analysed through EDA that seeds do impact the winning and losing teams. we hypothesized If that team has a lower seed than team2, there is a high probability of team1 winning. We gathered the seeds for the teams on both sides and then used them to extract seed differences between both teams. This new feature was then trained using logistic regression to perform the predictions. We examined the baseline classifier predictions in Fig.4.

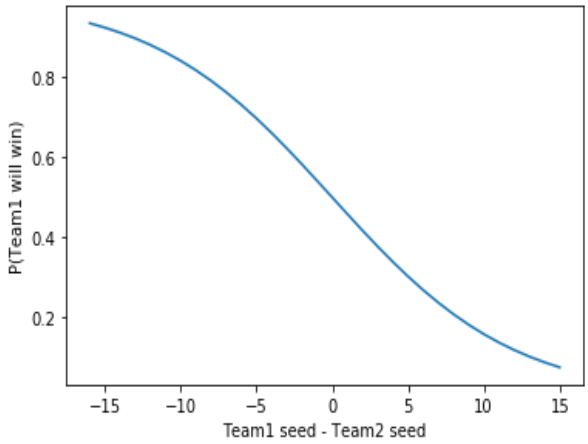


Fig. 4. Relation between seed difference and probability of team1 winning. (Predictions on 2017 dataset)

The baseline model performed well with log loss of 0.55. This encouraged us to explore and extract more features from the dataset.

## C. FEATURE ENGINEERING

We performed feature engineering on the detailed datasets to gather better representative features and construct an improved classifier. The feature engineering was performed using the 2019 detail season data.

We started by constructing two sets of the same dataset. Each dataset was then transformed with respect to winning features and losing features. The datasets were transformed by swapping the winning and losing data. For example for the winning teams, 'WTeamID' was renamed to 'TeamID' and 'LScore' to "OppScore". This allowed us to double the values in our dataset. It also introduced the opposite team features in the data and provided us the label. The binary label describes whether the team won (1) or lost (0) the match. Using these labels we analysed the teams that have won the most number of times (Fig. 5.).

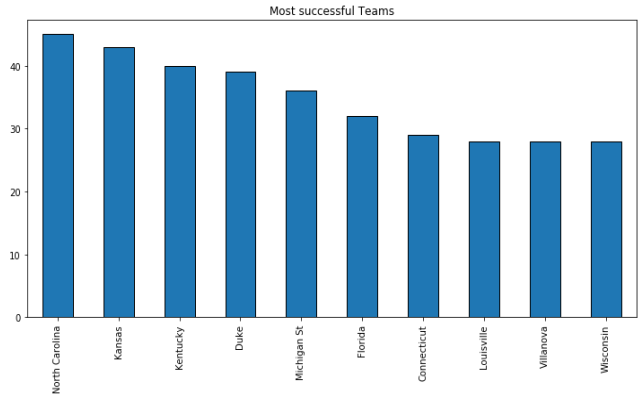


Fig. 5. Wining Frequency of top most successful teams since 2003.

Further, We used **Feature Scaling** to normalize the range of independent variables or features of data. We performed aggregation functions on the team box-scores to extract new features for the training. The box-scores contain game-by-game stats like free throws attempted, defensive rebounds, turnovers, etc. at the team-level. We performed mathematical operations on these stats to extract their mean or percentage values. We constructed the following features:

1. Wins: The Wins feature represents the number of times a team has won a match.
2. Field Goal Percentage (FGP): The FGP is calculated dividing the number of field goals made by the field goals attempted by each team
3. 3 pointer Percentage (FG3P): The FG3P contains the information about the percentage of successful 3 pointers out of all the attempted 3 pointers by each team.
4. Free Throw Percentage (FTP): The FTP shows the percentage of successful free throws made out of the attempted free throws by each team.
5. Average Offensive Rebound (OR_Avg): The average offensive rebound was calculated by dividing each team's offensive rebound by the number of offensive rebound values present in the dataset.
6. Average Defensive Rebound: This represents the mean values of defensive rebound of each team.
7. Average Assist: This represents the mean values of assist of each team.
8. Turnover Average: This represents the mean values of turnover of each team.
9. Average Steal: This represents the mean values of steals of each team.
10. Average Blocks: This represents the mean block values of each team.
11. Average Personal Foul: The field represents the mean personal foul of each team.
12. Average Team Score
13. Average Opponent Score

These features contained the play-by-play information of the regular seasons. We used regular season features as regular season results provide larger samples as compared to tournament results. To construct a classifier to predict the ncaa march madness tournament games results, we used the tournament compact results which contains the results of tournament games since 1985. We combined our extracted features with this dataset to create the tournament training data. Since the season feature contains data from 2003 to 2020, this left us with a lot of empty data for 1985 to 2003. This empty data was removed by the pandas library while merging the datasets.

Apart from these extracted features, we also added the seed information used in our baseline model. We merged the seed of each team and seeds of the opponent team with these extracted features using teamID and OppID as keys. We also combined the extracted seed difference feature.

After extracting these features, we transformed each feature into a winning team and losing team feature. We separated these features into winning and losing statistics. We created these positive and negative versions of the data so the supervised learning algorithm has sample data of each class to classify. These statistics were used to get the margin or difference between the winning feature values and the losing team feature values. This allowed us to reduce the number of values in the dataset and gave us a new set of features. The following features are extracted from above operations:

1. Seed Difference
2. Win Difference
3. Field Goal Percent Difference
4. FG3 Percent Difference (3 Pointers PercentDiff)
5. Free Throw Percent Difference
6. Offensive Rebound Avg Difference
7. Defensive Rebound Avg Difference

8. Assist Avg Difference
9. Turnover Avg Difference
10. Steal Avg Difference
11. Block Avg Difference
12. Personal Foul Avg Difference
13. Average Team Score Difference (PPGDiff)
14. Average Opponent Score Difference (OppPPGDiff)
15. Win Margin Difference: The Win margin was calculated by subtracting the opponent team's points per game from the winning team's points.

We examined these features by constructing a correlation matrix between these newly extracted features.

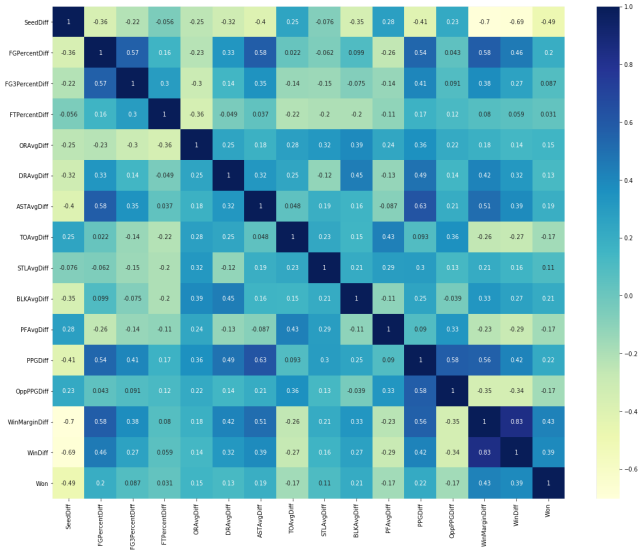


Fig. 6. Correlation matrix of extracted features.

From Fig.6. correlation matrix, we can confirm that none of these features are highly correlated (>0.9). However, as expected the correlation between the Win margin and Win diff is the highest at 0.83. We decided to select all these 15 features for the training. We dropped all the other features like teamID, season, DayNum, location from our training data because from the knowledge we acquired while working on this project, we learned that these features do not contribute towards the predictions.

We selected the 15 features for training our classifiers. The win feature is used as the target variable that contains the binary value of 0/1. We used the extracted features as an input to our binary classification algorithms.

## *D. MODEL SELECTION*

The constructed models are implemented to perform a binary classification. Binary or binomial classification is the task of classifying the elements of a given set into two groups (predicting which group each one belongs to) on the basis of a classification rule. In this case, we are classifying the teams into a winning and a losing group.

We have implemented the classic machine learning algorithms such as logistic regression (LR), k-nearest neighbour (KNN) and support vector machines (SVM) to perform binary classification. The python library sklearn is used to implement these models.

### *1.* K Nearest Neighbors Classifier (KNN)

The k-nearest neighbors (KNN) algorithm is a simple, supervised machine learning algorithm that stores all available cases and classifies new cases based on a similarity measure (e.g., distance functions).
For the implementation of the knn, we used sklear KNeighborsClassifier(). We also used the GridSearchCV function present in the sklearn model selection module to find the best k in the range of 80 for our classifier.

### *2.* Support Vector Classification (SVC)

The Support Vector Classification(SVC) is a simple, Scalable Linear Support Vector Machine for classification.A Support Vector Machine (SVM) is a discriminative classifier formally defined by a separating hyperplane. In two dimensional space this hyperplane is a line dividing a plane in two parts where in each class lay in either side.
For the implementation of the svm, we used sklearn SVC(). We specified the rfb kernel and other parameters such as gamma. We also used the GridSearchCV function to tune the selected parameters.

### *3.* Logistic Regression (LR)

The Logistic Regression (LR) algorithm is widely used for classification problems. It is a statistical model that in its basic form uses a logistic function to model a binary dependent variable.
For the implementation of the lr, we used sklearn LogisticRegression() function. We used the GridSearchCV function to find the best regularization parameter for our classifier.

## IV. RESULTS

After training our models on our featured dataset, We calculated the log loss of each model across both the dataset.

TABLE I. LOG LOG TABLE

| Model | | Log Loss | |
|---|---|---|---|
| | | **2017** | **2019** |
| 1. | Baseline model | 0.523 | 0.5144 |
| 2. | K Nearest Neighbors Classifier (KNN) | 0.5539 | 0.557 |
| 3. | Support Vector Classification (SVC) | 0.5428 | 0.5435 |
| 4. | Logistic Regression (LR) | **0.5161** | **0.4835** |

As shown in Table1. Logistic Regression is typically the top-performer. We use this classifier to make future predictions. Baseline model also performed better than svm and KNN across both dataset. SVC typically came close third. The provided values of log loss are a single representation of our classifier's success. Depending on how the data is shuffled, each run of the program may yield a slightly different classifier and thus different predictions /success rate.

Since logistic regression performed the best among the other trained models, we used it to make the predictions for the final brackets. To perform the final predictions, we first processed the sample submission file and transformed it into the input that can be fed into our classifier. We constructed the extracted season features for the teams in the sample submission files. Then we used this data to perform the classification using our trained logistic regression classifier and obtained the winning probability of each match. The obtained outcomes were clipped into the range of 0.05 to 0.95 to reduce the log loss. We similarly processed the submission file for both 2017 and 2019 datasets.

We submitted our predictions in the kaggle for both March madness 2017 and 2019. For 2017 predictions, we got the **log loss of 0.51613** , which is among the top 46% of the submission. Our predicted rank for 2017 competition would have been 200.

For 2019 predictions, we got the **log loss of 0.48538** , which is among the top 33% of the submission. Our predicted rank for 2019 competition would have been 264.

Using the submitted predictions, we build the brackets using bracketeer python library. Fig 7 shows the generated bracket for 2017 . Similarly, Fig 8. shows the generated bracket for 2019 tournament games.

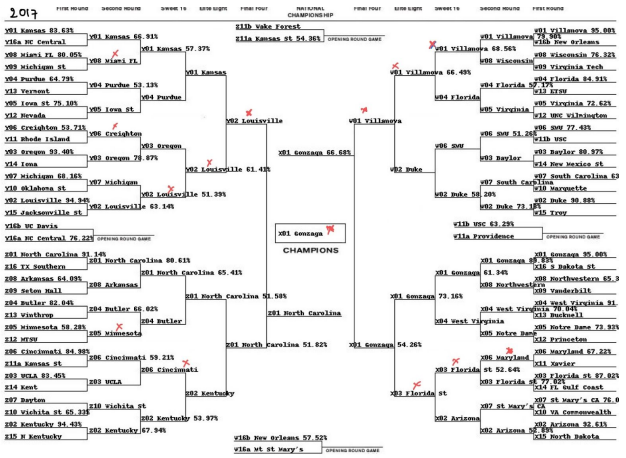

Fig. 7. Predicted Bracket for 2017

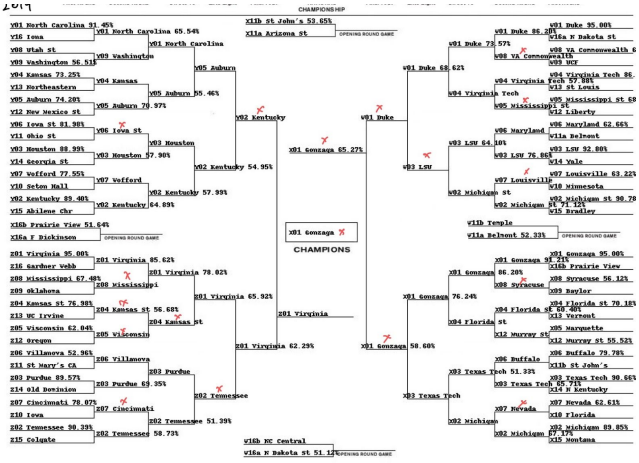

Fig. 8. Predicted Bracket for 2019

## V. CONCLUSION

This project presents a classical machine learning approach to predicting the NCAA Men's Basketball Tournament. The project provides a good exposure to feature extraction and feature selection. We found that the most challenging part of the project is extracting and selecting the best features that will help in predicting the correct outcomes of the matches.

In this work, we performed extensive feature extraction, feature selection and model selection to predict the outcome of march madness games . We confirmed that the simple logistic regression predicts the best outcomes with log loss of 0.48538.

For the 2017 bracket, we correctly predicted **49/63 matches**. For the 2019 bracket, we correctly predicted **46/63 matches.**