Machine Learning Engineer Nanodegree
Capstone Project
Matthew McFalls
May 5th, 2019

I. Definition

Project Overview

Every year, since the inception of the NCAA March Madness tournament, people have tried to predict who will win at each stage of the tournament. The single elimination tournament starts with 68 teams from the NCAA Division I. At each stage of the tournament people try to predict the winner with many different methods ranging from fan favorites to in-depth analysis of each team. The ultimate goal of many is perfectly predict every game victor, while other our happy to predict the victors from the final four teams on-ward.
For this project, I took the regular season team data, using various weighting through a logarithmic regression to attempt to predict the victors at each stage of the tournament, and the ultimate national champion. This project was inspired by the “Google Cloud & NCAA ML Competition 2019-Men’s on Kaggle” ("Google Cloud & NCAA® ML Competition 2019-Men's | Kaggle," n.d.).

Problem Statement

We are trying to solve this problem through the use of Machine Learning, specifically by applying a logarithmic regression. The outline of the process we followed is:

1. Download the data from the Kaggle competition.
2. Split the data into the prior years’ data and the current year’s data.
3. Create the common basketball features from the provided data.
4. Merge the data as needed into a single logical dataframe.
5. Weight the regular season data based on the last percent of games played.
6. Weight the regular season data based on the location of the last games played.
7. Build a classifier and loop over permutations of weights to find the best weights.
8. Predict the outcome of the season by building a March Madness bracket.

Metrics

The Kaggle competition is using log loss as their metric for determining the best model, which is normal for many Kaggle competitions. Log loss is used, because is determines accuracy through penalizing false classifications the model may make. Log loss penalizes highly confident incorrect classifications much more than slightly wrong classifications.

II. Analysis

Data Exploration

A dataset of previous NCAA tournaments and regular season data from provided by Google Cloud and Kaggle for this competition. The data included play by play data going back to 2010, Massey Ordinals, Cities involved in the NCAA, Regular Season results, Previous NCAA Tournament results, NCAA Tournament seed information under DataFiles, with the information for the 2019 Tournament under Stage2DataFiles. All the data uncompressed to around 1.5 GBs of comma separated files.
For this analysis, we will use Teams.csv, “RegularSeasonDetailedResults.csv, NCAATourneySeeds, TeamConferences.csv, Conferences.csv, NCAATourneyCompactResults.csv.

The Regular Season Detailed Results uses the following fields:

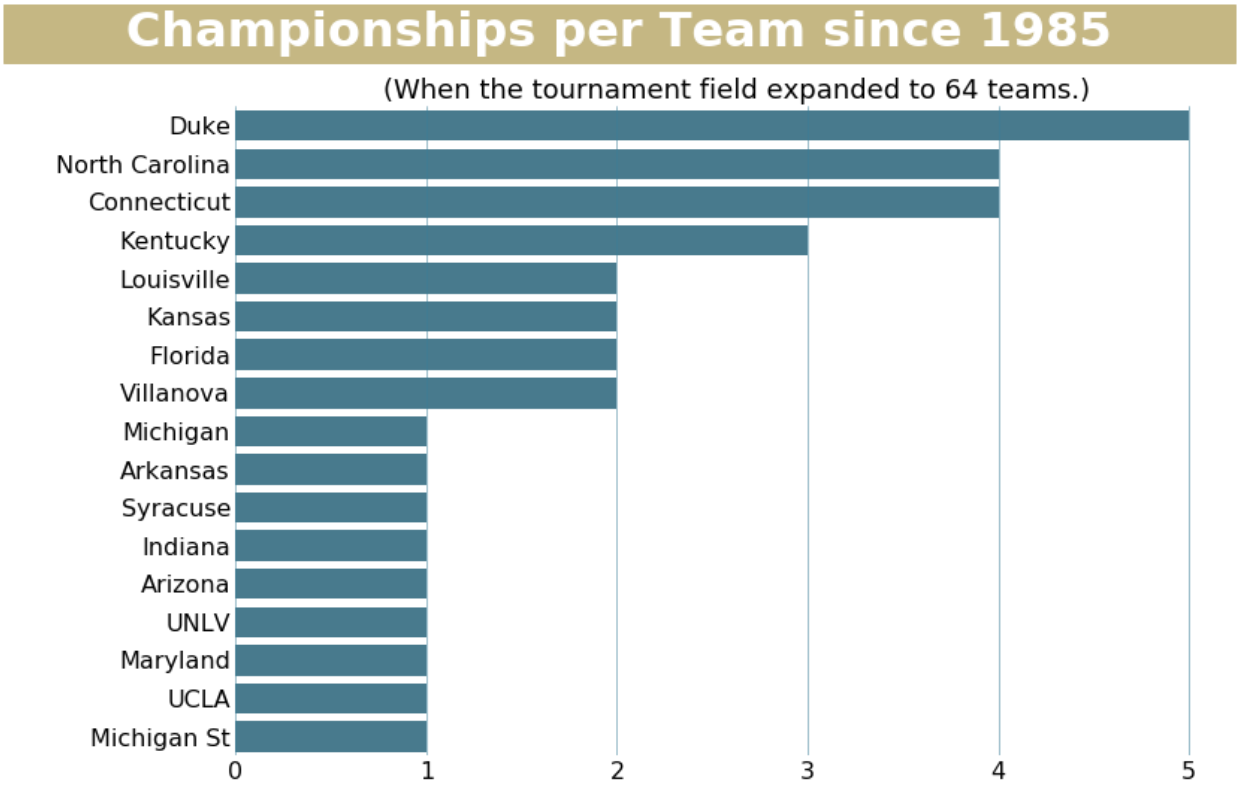
- Season - Year in which the tournament occurs
- DayNum - Integer representing the day a game was played from the start of the season; ranges from 0 to 132
- WTeamID - ID of the winning team
- WScore - Score of the winning team
- LTeamID - ID of the losing team
- LScore - Score of the losing team
- WLoc - Location of the winning team; “H” is a game where the winning team is the home team, “N” is a neutral location, “A” means the winning team is the visiting team.
- NumOT - Number of Overtimes
- WFGM - Winning team’s number of field goals made
- WFGA - Winning team’s number of field goals attempted
- WFGM3 - Winning team’s number of three point field goals made
- WFGA3 - Winning team’s number of three point field goals attempted
- WFTM - Winning team’s free throws made
- WFTA - Winning team’s free throws attempted
- WOR - Winning team’s number of offensive rebounds
- WDR - Winning team’s number of defensive rebounds
- WAst - Winning team’s number of assists
- WTO - Winning team’s number of turnovers
- WStl - Winning team’s number of steals
- WBlk - Winning team’s number of blocks
- WPF - Winning team’s number of personal fouls

The same information is created for the losing team, but instead of preceded with a “W”, the field is preceded with a “L”.
Many of the other fields in the data are self-explanatory, with the exception of the ‘Seed’ field in NCAATourneySeeds.csv. The “Seed” field is a concatenation of a region identifier and a numerical seed identifier. The region identifier is a W, X, Y, or Z used identify the region the team was from, and the numerical identifier represents the seed (ex: W02). If the team, is a play-in team, there is an additional a or b added to the end of the ‘seed’ composite field. The a or b is assigned by on which TeamID is lower during the play-in game. Play-in teams are the lowest seeded qualifying teams for the tournament which play against each other before the tournament; this allows a potentially greater variability to the first tournament round, and allows for a number of teams which is not a power of two.

Exploratory Visualization

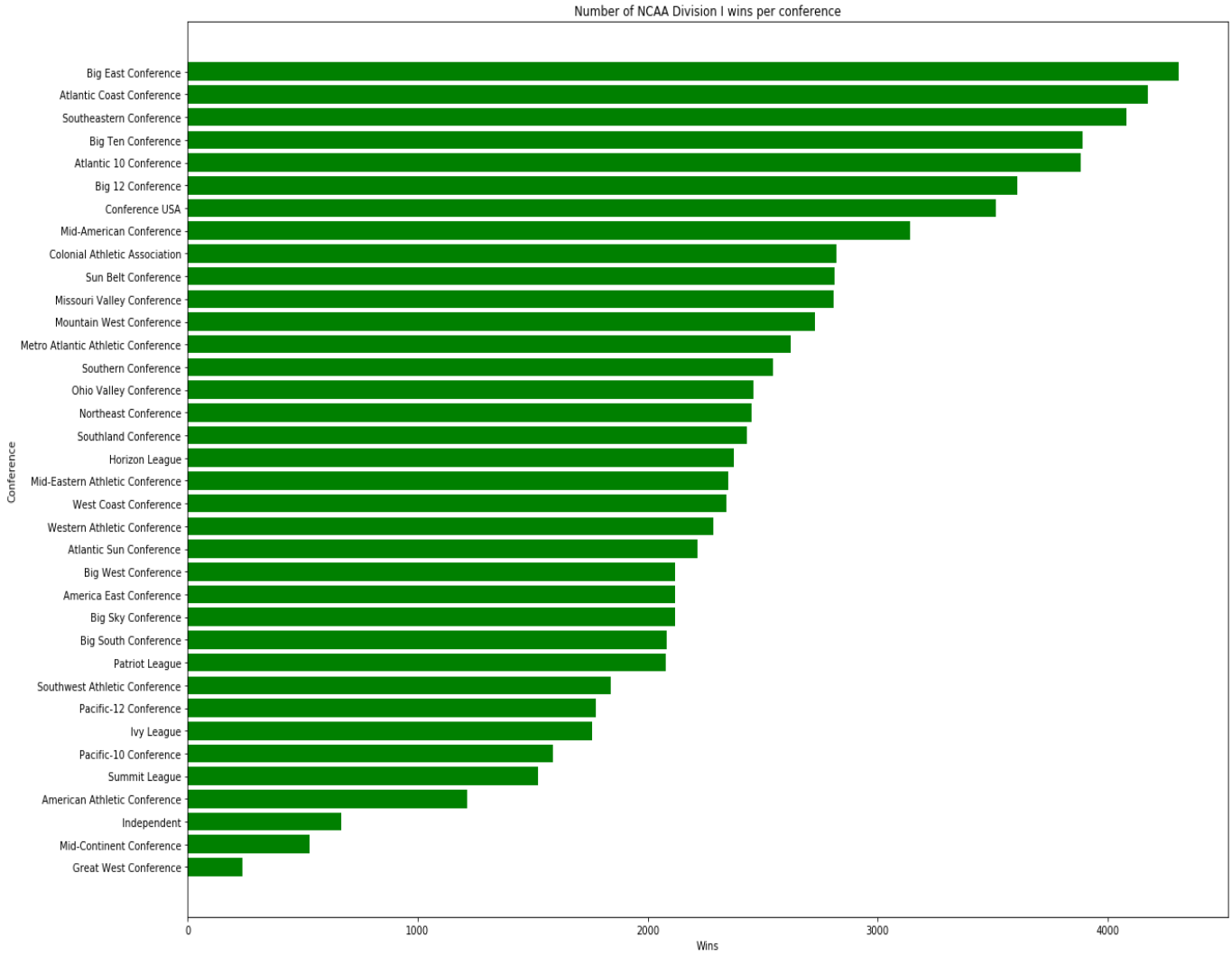
On an initial impression, it could be expected to see an even distribution of championship wins across many teams. However, since 1985, when the tournament expanded to 64 teams in the first round, we see many of the same teams repeatedly winning the championship. Many reasons could be asserted about why the same teams win repeatedly, but this would require a more in-depth analysis of each school and the history of the school.

Fig.1 *A horizontal bar plot showing the number of championship wins per team since 1985. The graph shows the top 17 teams with wins.*



The plot shows that the top two schools with the most wins come from North Carolina, with the third most championship wins going to Connecticut, and the fourth and fifth most wins going to schools in Kentucky.

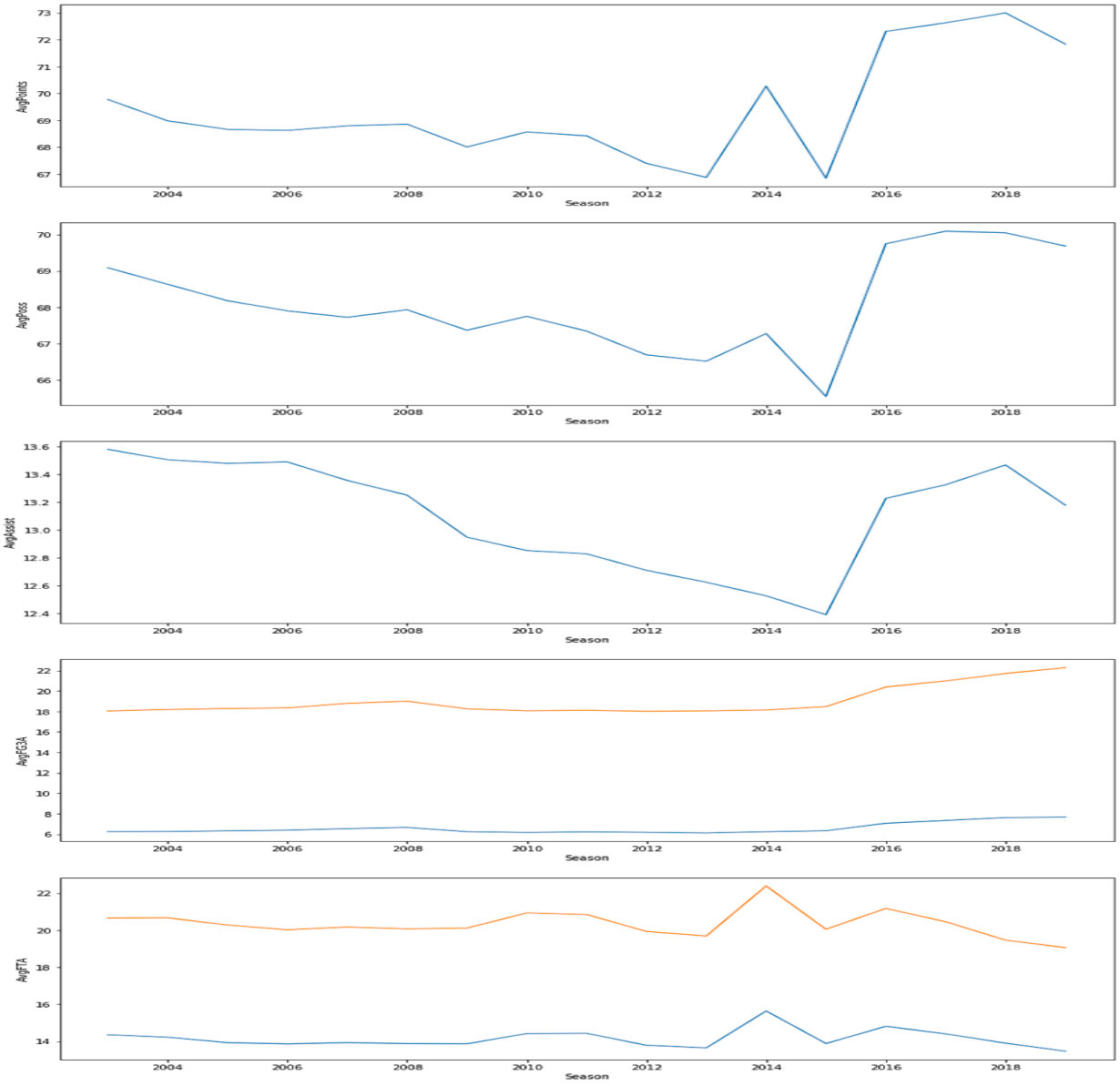
Fig. 2 The following horizontal bar plot shows the distribution of all NCAA Division I wins per any conference.



The above plot shows the distribution of all wins per conference. The conference with the most wins has been the Big East Conference, which currently only contains the Connecticut -- a team with the third most championship wins, while the Atlantic Coast Conference has fewer wins overall, but contains Duke University -- a school with the most all time championship wins.

The figure below shows the change in some common basketball metrics since 2004. The graphs show a distinct downward trend in average points average possessions, and average assists, then in 2014 there is a distinct increase in the previous metrics followed by a decrease in 2015. In 2016, the data shows a significant increase in average possessions, assists, and points, which indicates a change in the basketball metagame. This could be caused by many different factors which would require their own individual analysis.

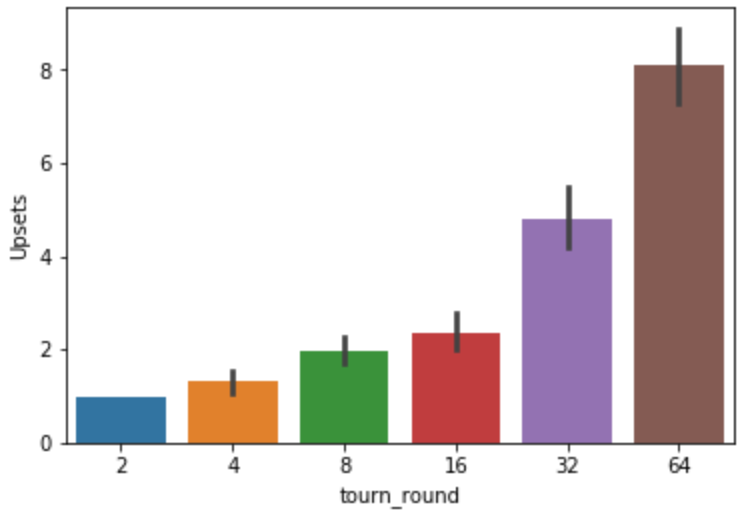
Fig 3. The line charts show the change in performance of specific basketball statistics over time. From top to bottom, average points per game, average number of possessions per game, average assists per game, average three point field goals attempted, average free throws attempted per season.For average three point field goals, the orange line indicates attempts, while the blue line indicates average successes. Average attempted free throws are indicated by the orange line, and successful free throws are indicated by the blue line.



Over time, the average three point field goals attempted relatively stagnant, without a large change, until 2015 when we see a definitive increase in the number of attempted three point field goals, along with a smaller increase in the successful three point field goals.
The average free throws attempted and completed have seen similar ups and downs over the years, with a notable increase in 2014, followed by a small decrease in 2015. In 2016 we see another small speak in free throws attempts and success, followed by a gradually decrease in

free throws in the years after 2016. This seems to indicate the metagame change has placed less importance on free throws, and more importance on successful three point field goals.

Fig. 4 shows the number of times a lower seeded team has won against a higher seeded team in reach round of the tournament.



During the tournament fans, expect quite a few lower seeded teams to be victorious against higher seeded teams, especially in the first several rounds of the tournament. In the first round, all 64 teams, there are typically quite a few upsets. Over the course of the tournament, the data shows fewer and fewer upsets per found, with the final four and final round having the least number of lower seeded teams winning against higher seeded teams. This shows the seeding calculation by the NCAA is a reasonably reliable indicator of a team's performance, especially when a team is highly seeded -- higher seeded teams have lower numbers.

Algorithms and Techniques

We are using a logarithmic regression for our prediction of tournament game outcomes based on regular season basketball data. Our goal is to minimize the log loss regression as close to zero as possible. We have the following parameters we can tune to affect the model:

- DayWeight - The weight of the last 30 days of the tournament
- LocWeight - Weight of the away games
- clf_C - Range of values to search for the best value.
- Clf_penalty - l1 or l2 penalty

Test and training data is split from the master data on each iteration of the search loop for weights; while is is not efficient it showed to be reliability for our use.

Benchmark

To gauge the performance of the model being built, we will create a benchmark model for comparison. The benchmark model we will be using the default Kaggle starter kernel for the competition. The starter kernel uses the difference between the seed of two different teams in a logarithmic regression to create a model for predicting which team would win.

The starter kernel's logarithmic regression finds a best log loss of 0.5546 and the best C of 0.01.

III. Methodology

Data Preprocessing

From the data provided through the Kaggle competition, we will need to generate some additional features to aid our analysis and merge some of the data into a single point for manipulation.

The regular season conference information, conference names, and team conference information, will be merged into a single dataframe. Once this information is merged into a single dataframe, then it is split into winning teams, winning conferences, losing teams, and losing conference for later use

The regular season data will need to have several features generated for use in the logarithmic regression. First the winning team, winning conference, losing team, losing conference information is merged with the regular season data as a basis for generating many of the statistics -- many of the statistics need to be generated from the winning and losing team perspectives and merging this information makes the process simpler.

The following statistics will be generated from the merged regular season data; the same data is calculated for the losing team, but we will only show the winning team:

- WFGM2 = WFGM - WFGM3
- WFGA2 = WFGA - WFGA3
- Wposs = WFGA + 0.475 * WFTA + WTO - WOT
- Wshoot_eff = WScore / (WFGA + 0.475 * WFTA)
- Woff_rtg = WScore / Wposs * 100
- Wdef_rtg = Loff_rtg
- Wsos = Woff_rtg - Loff_rtg
- Wts_pct = WScore / (2 * (WFGA + 0.475 * WFTA)) * 100
- Wefg_pct = (WFGM2 + 1.5 * WFGM3) / WFGA
- Worb_pct = WOR / (WOR + LDR)

- Wdrb_pct = WDR / (WDR + LOR)
- Wreb_pct = (Worb_pct + Wdrb_pct) / 2
- Wto_poss = WTO / Wposs
- Wft_rate = WFTM / WFG
- Wast_rtio = WAst / (WFGA + 0.475*WFTA + WTO + WAst) * 100
- Wblk_pct = WBlk / LFGA2 * 100
- Wstl_pct =WStl / Lposs * 100
-

Once all the features have been engineered, we average all the metrics for each winning team over the course of the entire regular season.

Implementation

We implemented a logarithmic regression using log loss and C as our metrics to analyze which combination of features provided the fitting regression.
We decided to weight the last 30 days of the regular season, as this would be more reflective of performance headed into the tournament, and we decided to weight games away from the team's home court, because this would be more reflective of the performance of the team while traveling during the tournament.
After preprocessing the data into a standard dataframe, we needed to find the best weights, penalties, log loss and C. To find the best weights for the last 30 days of the regular season and the location of the game, we created a list of permutations of the weights between 0 and 1 in increments of 0.1, then we looped over each permutation and build a logarithmic regression to determine the log loss of the combination. Next, we dropped the Seed data and built another logarithmic regression using the same weights to determine if the Seed data was needed. Each iteration of logarithmic regression used a grid search to find the best penalty between l1 and l2 penalties, and to find the best C value.
To build the logarithmic regression, we take the difference between the two teams, and feed the data into the classifier to build the regression.

Refinement

After the initial search for the best weights, we returned to the permutations, and reduced the search range based on the initial best weights. The initial best weights were 0.2 for e last 30 days of the season and 0.3 for the location. We created a new list of permutations limited to between 0.1 and 0.3 to search for a more accurate weights. We found 0.28 was an optimal weight for the last 30 days of the regular season, and 0.22 was an optimal weight for the location of the regular season games.
The log loss with the original was 0.534233, while after reducing the weight search range based on the initial weights yielded a log loss of 0.54093. The log loss was slightly worse after finding more precise weights, as we are trying to get the log loss as close to zero as possible.

IV. Results

Model Evaluation and Validation

For evaluation and testing, we applied the model to the prior year's NCAA tournaments and generated a log loss. We chose to use the model generated during the parameter search which yielded the lowest log loss of all the searched models.
During the search for the best weights, the model showed to be quite sensitive to small changes in a teams statistics, which would result in a change in the likelihood of a team winning.
The final validation was to run the model on the current year's tournament contenders and attempt to predict which team would win at each stage of the tournament, along with the final championship winner. The final log loss for the 2019 NCAA tournament was 0.5577.

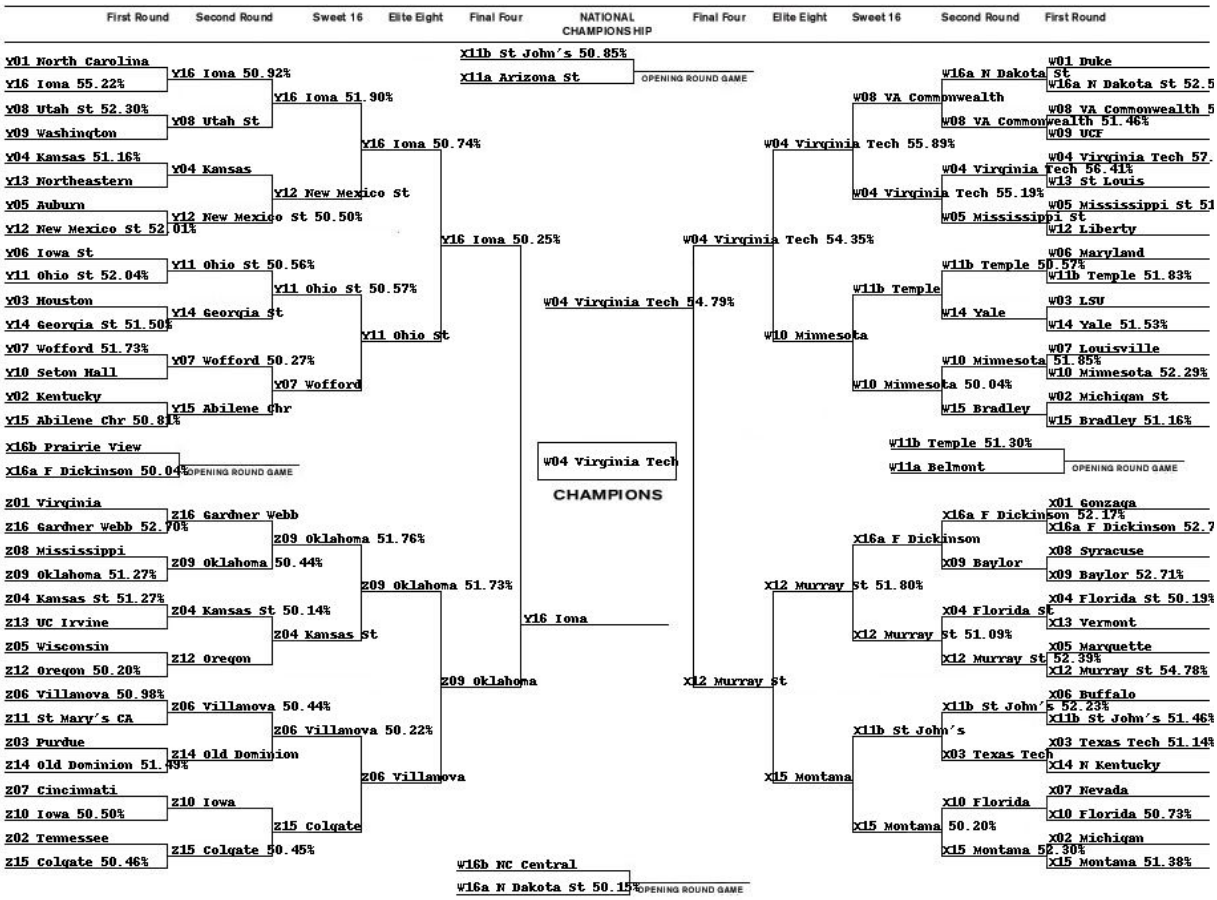
Justification

Overall, the model we built performed worse than the baseline model. The baseline model had a log loss of 0.5546 and our model had a log loss of 0.5577 -- a difference of -0.0031 from the baseline. While the difference is small, such a difference can result in different teams being predicted to win or lose from the baseline model.
The ultimately check of the results was to compare the predicted tournament results to the actual tournament results. Several of the games were predicted with accurate winners; however, there were many upsets later in the tournament by lower seeds which were difficult to predict based on the way the model was crafted using only regular season average statistics.
In the end, the model was not effective to predicting all the games nor the final winner of the championship for 2019.

V. Conclusion

Free-Form Visualization

Fig. X The final bracket generated by the model to predict the NCAA 2019 basketball championships.



Virginia representing the University of Virginia won the 2019 NCAA tournament, while our model predicted them being eliminated in the first round of the tournament.

Reflection

We used the following overall process for this problem:

1. Define problem based on Kaggle competition requirements.
2. Download provided dataset.
3. Generate new features from basic provided data.
4. Search for the best weights for the regular season, by building a regression represented by each permutation of weights.
5. Test model against prior years tournaments.
6. Build model on current year's tournaments.
7. Build bracket based on model.

This problem has many difficult aspects related to predicting human performance over a period of time based on their prior performance; it is interesting to try to predict how sports team will

perform based on historical data, there is still much left to be learned about how to model human competitive behavior. A more in-depth knowledge of the sport, along with a better analysis of choosing models may generate better results for future predictions. With more tuning, this model may provide a better basic predictive capability.

Improvement

From a programming standpoint, in the loop for finding the best weights, it would be better to not save each classifier, but to save the critical values of each result for later comparison. This would free up resources, and may give a small performance increase due to more available resources.

For better predictive abilities, modeling the performance of individual players, groups of players, and individual plays could provide better predictions by providing more granular data. Currently, the model does not take in the effect of an outstanding player on a team, or the general teamwork of a specific group of players. The model does not consider how teams change their plays based on their opponents, nor how plays affect the outcome of each game.

References

Google Cloud & NCAA® ML Competition 2019-Men's | Kaggle. (2019). Retrieved from https://www.kaggle.com/c/mens-machine-learning-competition-2019