Forecasting the 2023 NCAA Basketball Tournament

Aleesa Mann and Cyruss Tsurgeon
School of Applied Computational Sciences, MSDS 565, Spring 2023
3401 West End Avenue, Suite 260, Nashville, TN 37203
amann22@email.mmc.edu; ctsurgeon07@email.mmc.edu

***Abstract* — Predicting the winner of individual tournament games and the ultimate champion of the NCAA Basketball Tournament is a fun, yet challenging endeavor. The Kaggle 2023 March Machine Learning Mania competition is considerably more complex with an objective to predict the probabilistic outcome of the hypothetical matchups of each team against all other teams in their league. We sought to tackle this challenge using machine learning predictive analytics. In this work we show that several machine learning algorithms are appropriate to accomplish this task. However, trying to accurately predict outcomes that show so much unpredictability proved to be very difficult.**

Keywords— College Basketball; NCAA Tournament; Predictive Analytics; Machine learning; Kaggle

I. INTRODUCTION

Background

Predicting the winner of individual tournament games and the ultimate champion of the NCAA Basketball Tournament is a fun, yet challenging endeavor many individuals around the world attempt each March-April when the tournament is held. According to the American Gaming Association, in 2017 as many as 70 million brackets were completed[1] and an estimated 45 million people last year[2] competed through some verifiable online sports websites (such as ESPN, CBSSports, NCAA, Yahoo!, etc.).

To date, no one has ever completed a 'perfect bracket' - correctly predicting the winner of each of the 63 total games in the NCAA Tournament. In fact, the odds of completing a 'perfect bracket' are truly absurd. If you were to just guess or flip a coin the odds of all of your picks being correct are 1 in (9.2 quintillion) 9,223,372,036,854,775,808 - that's 1 in 9.2 quintillion. Or if you know a little something about basketball, the odds are estimated to be around 1 in 120.2 billion[3]. Although many people complete their bracket in private (and never share them), to date, the longest (verifiable) streak of correct picks in an NCAA tournament bracket is at 49 games - which is at the start of the third round (by Greg Nigl of Columbus, Ohio in 2019)[4]. Each year, there is a team with 'no chance' who manages to beat a higher seeded team; or a team with a great regular season winning percentage who manages to be upset, unexpectedly, by another team. Hence, the popular reference to the tournament as 'March Madness'. Out of the popularity for the tournament itself and the rising popularity of predictive machine learning algorithms, many data enthusiasts have entered into competitions to try to use predictive analytics to forecast the NCAA tournament.

For our project, we decided to use the provided data and competitive structure of the Kaggle 2023 March Machine Learning Mania competition, – now into its ninth annual edition[5]. For the millions of people who fill out brackets predicting the ultimate Champion of the tournament, the objective is to consider each game individually, predict who will continue and who will not; then repeat this for all 63 games - declaring one team as the Champion. For the Kaggle competition (and our project), the task was considerably more complex with an objective to predict the outcome of a hypothetical matchup of each team matched against all other teams in their league. There are 363 men's Division-I teams this season, thus a submission for the Kaggle competition would need to include predictions for all possible pairs of those 363 teams (leading to over 65,000 total possible combinations), and there are 363 * 362/2 = 65,703 possible combinations. We also needed to repeat this exercise for the women's league, as well. There are 361 women's Division-I teams this season - which corresponds to 64,980 possible combinations. So our final submission file must have 65,703 + 64,980 = 130,683 predictions. An additional layer of complexity for the Kaggle competition requires that we not just submit a winner for each matchup, rather we were tasked to submit a probability for each outcome.

Goal: For the purposes of this class project, we simplified our problem definition to focus only on predicting the outcomes of the 2022 Men's NCAA College Basketball Tournament.

Related Work

In 2012, Chris Wright[6] provided a statistical analysis of the predictors of success in March Madness. He found that win percentage and defensive efficiency both had a large positive impact on the outcome. Overall, he concluded that it is very difficult to predict the winners of March Madness matchups. Noting that even with an unbiased model, the error term is large enough to create a lot of outcomes that were not predicted.

In 2014, Alex Tran and Adam Ginzberg[7] reported in their Stanford final project paper, that nearly all of the game

statistics were useless except for FG%, FT%, and 3PT%, which were marginally helpful. Also in 2014, Levi Franklin[8] found in his project paper that margin of victory, difference between seeding, and performance from the previous tournament were useful features for training a machine learning model.

And finally, in 2018, Cody Kocher and Tim Hoblin[9], used generalized linear, random forest, and decision tree models to perform predictive analytics on the NCAA Tournament. They found that the correlation between the statistical variables used for analysis tended to be high among several variables. "A team's win-loss record has a strong positive correlation to its average margin of victory." "Seed and strength of schedule are also highly correlated because teams who play more difficult schedules are usually rewarded with better seeds." They used principal component analysis to reduce related statistics with limited success.

We took these considerations into account when we performed our analysis.

II. DATASET AND FEATURES

The dataset we used included 31 well structured csv files containing various items of information and statistics about Men's NCAA basketball teams. These files were completely organized without missing values. We are not going to run through all the data files in great detail, but will just mention each one to give you a complete understanding of the data and features we were working with.

The first set of files (see Table 1) contains the main files we used for our analysis. The data includes a file that lists all of the seasons beginning from 1985 through 2023 and the date the season started, 'DayZero'. It should also be noted that there were no tournament games in 2020 due to the COVID-19 pandemic.

Table 1: Dataset and features – main files

Data Section 1 Main Files	
Seasons	Season, DayZero, RegionW, RegionX, Region Y, Region Z
Teams	TeamID, TeamName, FirstD1Season, LastD1Season
Regular Season Detailed Results	Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, WFGM, WFGA, WFGM3, WFGA3, WFTM, WFTA, WOR, WDR, WAst, WTO, WStl, WBlk, WPF
Tourney Detailed Results	Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, WFGM, WFGA, WFGM3, WFGA3, WFTM, WFTA, WOR, WDR, WAst, WTO, WStl, WBlk, WPF
Tourney Seeds	Season, Seed, TeamID
Sample Submission	ID, Pred (e.g. 2018_3181_3314,0.516)

The data includes a file containing all of the teams listed by season. The number of division schools varies from season to season.

The data includes files containing detailed game-by-game matchups for entire seasons. Each game included the number of days from 'DayZero' the game took place, team id, scores, as well as other statistics for both the winning and losing teams. The detailed game-by-game data was also provided for the NCAA tournament games for previous years. These detailed game results only went back as far as 2003. But, we found these files to be the most beneficial.

The data also includes files providing the seeding for each team in the tournament; and a sample of how the submission file should look.

The next set of files (see Table 2) includes some extra files such as the cities where the games are played - this could be beneficial as teams tend to perform better in a friendlier home crowd.

Table 2: dataset and features – extra files

Data Section 2 Extra Files	
Cities	CityID, City, State
Game Cities	Season, DayNum, WTeamID, LTeamID, CRType, CityID
Compact Season Results	Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT
Compact Tourney Results	Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT
Compact Secondary Tourney Results	Season, DayNum, WTeamID, WScore, LTeamID, LScore, WLoc, NumOT, SecondaryTourney
Tourney Slots	Season, Slot, StrongSeed, WeakSeed

It includes game-by-game statistics (similar to the detailed files) but with only a smaller set of features. The files include

a file containing game-by-game statistics for secondary tournaments.

And the last set of files (see Table 3) includes some supplementary files such as coaches (which could be useful as some more experienced coaches show better team success in tournament games where the stakes are all-in. Other files included in this set are the conferences the teams play in (some conferences perform better than others during the tournament). Other files include team rankings for a number of different ranking systems, and team spelling (which provides alternate forms of the spelling of the team name).

III. METHODS

One of the feature sets we developed was by taking the regular season data, and splitting it so that one dataframe represented the winning teams' metrics, and another dataframe represented the losing teams' metrics. These data frames were then used to develop average, median and count metrics grouped by season and team. In this way, each team had a datapoint representing their performance in each season (2023 - 2022). These two data frames were then merged together as they were originally and the columns representing the winning and losing teams were randomized so that not all the winning teams were listed in the same column. In this final version of the dataset, each datapoint represented a match (Team A vs. Team B) and the relevant metrics of both teams. And each of these matchups were represented twice (A vs. B, and B vs. A) to give the models more information about the matchups. The final dataset representing tournaments since 2003 had 1,181 rows and 70 columns. This dataset was then used to determine the outcome of the team in the first team column (team_1) winning the game. (In this dataset team_0 was listed after team_1 in the order of the columns).

Table 3: Dataset and features – supplementary files

Data Section 3 Supplements	
Team Coaches	Season, TeamID, FirstDayNum, LastDayNum, CoachName
Conferences	ConfAbbrev, Description
Team Conferences	Season, TeamID, ConfAbbrev
Conference Tourney Games	ConfAbbrev, Season, DayNum, WTeamID, LTeamID
Secondary Tourney Teams	Season, SecondaryTourney, TeamID
Massey Ordinals	Season, RankingDayNum, SystemName, TeamID, OrdinalRank
Team Spellings	TeamNameSpelling, TeamID
Seed Round Slots	Seed, GameRound, GameSlot, EarlyDayNum, LateDayNum

For feature selection, we used the scikit learn Sequential Feature selector with forward selection and time series cross validation to reduce the feature set. We used this process to reduce our 70 features to two feature sets of 15 and 10 to see how feature size influenced the models.

RR 15	RR 10	LR 15	LR 10	SVM 15	SVM 10
team_1	team_1	**Seed_1**	**Seed_1**	team_1	**Seed_1**
Seed_1	**Seed_1**	**Seed_0**	**Seed_0**	**Seed_1**	**Seed_0**
team_0	team_0	***num_ot_mean_1***	***num_ot_mean_***	**Seed_0**	***num_ot_mean_1***
Seed_0	**Seed_0**	fga_mean_1	fga_mean_1	***num_ot_mean_***	fga_mean_1
team_score_mean_1	team_score_mean_1	fgm3_mean_1	ast_mean_1	fga_mean_1	num_ot_median_1
num_ot_mean_1	stl_mean_1	ast_mean_1	stl_mean_1	num_ot_median_	ftm_median_1
dr_mean_1	blk_mean_1	stl_mean_1	num_ot_median_	ftm_median_1	blk_median_1
stl_mean_1	fga_median_1	num_ot_median_1	dr_mean_0	blk_median_1	num_ot_mean_0
blk_mean_1	pf_median_1	fga3_median_1	num_ot_median_	pf_median_1	ast_mean_0
num_ot_median_1	dr_median_0	ftm_median_1	fgm3_median_0'	num_ot_mean_0	dr_median_0
fga_median_1		blk_median_1		ast_mean_0	
pf_median_1		dr_mean_0		num_ot_median_0	
dr_mean_0		pf_mean_0		fga_median_0	
num_ot_median_0		num_ot_median_0		dr_median_0	
dr_median_0		fgm3_median_0		blk_median_0	

Table 4: Selected feature sets and models

The features bolded and highlighted in green were included in all models. The features bolded and in italics were included in at least five of the models, signifying their importance in the prediction calculation.

We then ran these feature sets through Ridge Regression, Logistic Regression and SVM models. Then we used scikit learn's GridSearchCV() to tune the parameters. For logistic regression, the data was scaled before fitting to the model. Then we evaluated the different models using the accuracy score metric. Again, we are testing to see if the team listed as "team_1" is the winner (signified by a '1' in the data frame's 'winner' column which was our target variable).

IV. EXPERIMENTS AND RESULTS

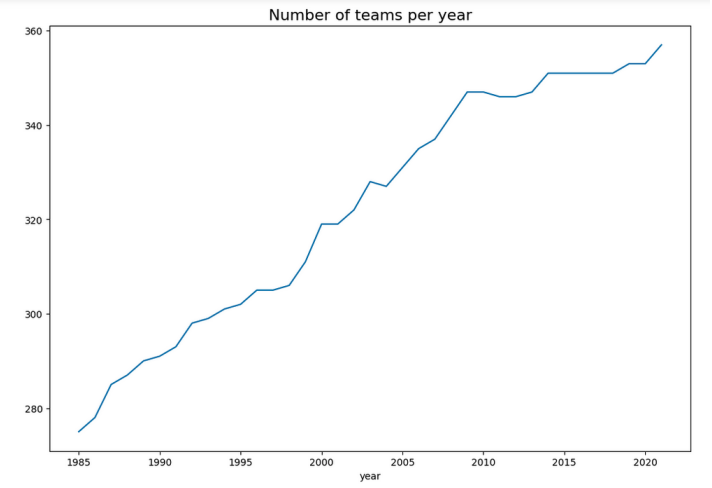
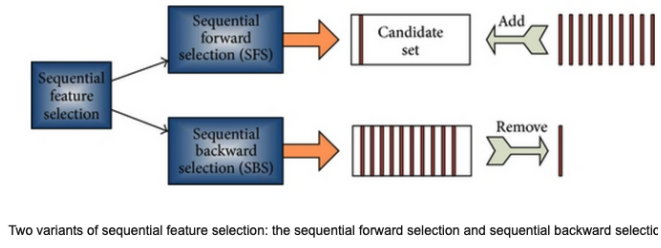


Figure 1: Number of teams per year

	team_1	Seed_1	team_0	Seed_0	winner
0	1411	16	1421	16	0
1	1400	1	1421	16	1
2	1112	1	1436	16	1
3	1112	1	1211	9	1
4	1153	8	1211	9	0
...	...	...	...	...	...
1176	1325	13	1438	4	1
1177	1222	2	1333	12	1
1178	1329	4	1333	12	0
1179	1260	8	1333	12	0
1180	1234	2	1332	7	0

Figure 2: Comparison of seed vs win in team_1 and team_0

We also wanted to test the models against the accuracy prediction of always choosing the team with the seed with the lower value (which is the higher ranking, Seed 1 is the top seed, Seed 16 is the lowest) as the winning team. If you always chose the seed with the lower numeric value as the winning seed, you would have 33% accuracy. This is fairly low, but it is one of the metrics we could test our models against, and all of the models tested higher than 33% accuracy.



Two variants of sequential feature selection: the sequential forward selection and sequential backward selectic

Figure 3: Sequential feature selector and forward selection[10]

In using the sequential features selector (Figure 3) across the RidgeRegression, Logistic Regression, and SVM models, we developed two sets of features, one set of 15 features and another set of 10 features, we then fed those two feature sets into each model to compare performance.

Table 5: Example model tuning

Model	# Features	Tuning
SVM	19	kernel = 'linear', probability = True
KNN	19	k = 5
GNB	19	
DT	19	max_depth = 5
RF	19	n_estimators = 10
MLP	19	alpha = 1, max_iter = 1000

We used a number of different algorithms to build a good model for our predictions. One of the approaches we took is stacking. We used six different algorithms to model our game predictions and later used a stacking classifier to combine all of these models into a single ensemble model. We used scikit learn's StackingClassifier method for assembling various models as estimators and then using these estimators as input for a LogisticRegression model. For stacking, we excluded the decision tree model because the model did not perform well during validation and all of the predicted outcomes appeared to be too similar.

Table 6: Example model tuning and accuracy

Model	# Features	Tuning	Avg Accuracy
SVM	15	Linear Kernel	0.691
	10	Linear Kernel	0.692
	10	RBF Kernel	0.693
Ridge Regression	15	Alpha = 1	0.698
	15	Alpha = 1	0.700
	10	Alpha = 0.01	0.700
Logistic Regression	15	Data Scaling	0.699
	10	Data Scaling	0.698

What we see from these models in Table 4 is that they all perform similarly on the data set, and that using a smaller number of features is a better option for developing our model. On the training data (season tournaments between 2003 -2021), the Ridge Regression model performed slightly better, and on the testing set (2022 tournament games) the Ridge Regression and SVM models performed best, probably because they are more complex that the logistic regression models. The small size of the dataset may be one contributor to the performance of the models on this dataset, with a larger dataset, the models may be able to pick up more nuance and make better predictions.

This feature set and model selection would have to be further improved, because in the real-world setting we would not have the actual Team vs Team matchups. But it provides a basis for a model. We actually implemented our model development in the 2023 Kaggle March Madness competition. In this case, instead of predicting a limited number of

matchups, we made predictions for all possible combinations of teams in the tournament.

Table 7: Example model testing accuracy

Model	# Features	Tuning	Accuracy
SVM	10	RBF Kernel	0.686
Ridge Regression	10	Alpha = 0.01	0.686
Logistic Regression	10	Scaling Data	0.64

For the Kaggle competition, the metric used to score predictions was the Brier Score. This metric was unfamiliar to us prior to this competition; however, we made an effort to incorporate the metric into our model evaluation. Scikit learn includes a brier_score_loss metric in its metrics class. We attempted to define a custom function to create a brier score metric, but ultimately, we went with the scikit learn brier_score_loss metric because it incorporated well with all models.

In terms of results, for the models that we trained using the stacking classifier, we saw very high accuracy scores. However, using the brier metric to evaluate our model, we saw the scores right around what would be expected. For the Brier Score the lower number is better.

The performance metrics for the various models used in the sacking classifier:

Figure 4: Support Vector Machine (SVM) Model

```
Model performance for Training set
- Brier: 2.7481818774448784e-05
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Validation set
- Brier: 2.7529527645039904e-05
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 2.7481818774448784e-05
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

Figure 5: K-Nearest Neighbor (KNN) Model

```
Model performance for Training set
- Brier: 0.028234475906607055
- Accuracy: 0.9690511674118232
- MCC: 0.9380381167989247
- F1 score: 0.9690493556340358
----------------------------------
Model performance for Validation set
- Brier: 0.05172932330827068
- Accuracy: 0.9473684210526315
- MCC: 0.8946297104615851
- F1 score: 0.9473505323383233
----------------------------------
Model performance for Test set
- Brier: 0.028234475906607055
- Accuracy: 0.9675116029989289
- MCC: 0.9349235013292083
- F1 score: 0.9675058978031689
```

Figure 6: Gaussian Naive Bayes (GNB) Model

```
Model performance for Training set
- Brier: 0.02758298982492225
- Accuracy: 0.962046696472926
- MCC: 0.9240196564392616
- F1 score: 0.9620474052496035
----------------------------------
Model performance for Validation set
- Brier: 0.0296741742301681
- Accuracy: 0.9624060150375939
- MCC: 0.9248529693696184
- F1 score: 0.9624145238329231
----------------------------------
Model performance for Test set
- Brier: 0.02758298982492225
- Accuracy: 0.9593002499107461
- MCC: 0.9184746737351663
- F1 score: 0.959301980094881
```

Figure 7: Decision Tree (DT) Model

```
Model performance for Training set
- Brier: 0.0
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Validation set
- Brier: 0.0
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 0.0
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

Figure 8: Random Forest (RF) Model

```
Model performance for Training set
- Brier: 0.0004619970193740685
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Validation set
- Brier: 0.0010526315789473686
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 0.0004619970193740685
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

```
Model performance for Training set
- Brier: 0.00023696987546745112
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Validation set
- Brier: 0.0004611734904114934
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 0.00023696987546745112
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

Figure 10: Stacked Model

```
Model performance for Training set
- Brier: 4.087152605165435e-07
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
----------------------------------
Model performance for Test set
- Brier: 6.749724411646529e-07
- Accuracy: 1.0
- MCC: 1.0
- F1 score: 1.0
```

Figure 11: Summary Table

	Brier	Accuracy	MCC	F1
svm	2.748182e-05	1.000000	1.000000	1.000000
knn	2.823448e-02	0.969051	0.938038	0.969049
gnb	2.758299e-02	0.962047	0.924020	0.962047
rf	4.619970e-04	1.000000	1.000000	1.000000
mlp	2.369699e-04	1.000000	1.000000	1.000000
model	4.087153e-07	1.000000	1.000000	1.000000

V. CONCLUSION AND FUTURE WORK

This project was fun, but very challenging. Trying to accurately predict outcomes that show so much unpredictability proved to be challenging, but there were a few insights that could be carried over into future work.

Using the sequential features selector across the RidgeRegression, Logistic Regression, and SVM models, there were some variables that appeared more often in the selection processes. The seed number for team 1 (the winning team we were predicting the outcome for) and the seed for team 0 were used in all eight models. The repetition of these variables means they have significant weight in determining whether or not 'team_1' wins in our tournament predictions when using the outlined models. There could be more exploration into why these variables are significant.

There needs to be a way to account for the randomness of the matchups, what kind of metric can be used for this? (i.e. a player is hurt, team's lack of sleep due to travel schedule, etc.). Especially considering this is a single-elimination tournament, which means any aberration in performance could mean a top team losing or a bottom team winning. This type of unpredictability is the 'madness' of March Madness.

One learning point from this project was that we should have provisioned our time at the onset (to allow sufficient time for model and hyperparameter tuning). Another learning point, feature engineering is very important. It would have been better to establish 'hot team' and 'tired team' features - for streaking teams; bring in time-series.

We believe using a convoluted neural network model with transfer learning from regular season data might be a better approach. The Brier score metric was new for us, fully understanding this metric and using this metric to optimize the model may also prove to be a smarter tactic.

REFERENCES

[1] David Purdum (2017). ESPN – 70 million brackets, $10.4 billion in bets expected for March Madness. American Gaming Association. https://www.americangaming.org/new/espn-70-million-brackets-10-4-billion-in-bets-expected-for-march-madness/ accessed: 18 March 2023.

[2] American Gaming Association (2022). 2022 March Madness Wagering Estimates. American Gaming Association. https://www.americangaming.org/resources/march-madness-2022/ accessed: 18 March 2023.

[3] Daniel Wilco (2023). The absurd odds of a perfect NCAA bracket. https://www.ncaa.com/news/basketball-men/bracketiq/2023-03-16/perfect-ncaa-bracket-absurd-odds-march-madness-dream accessed: 18 March 2023.

[4] Brian Budzynski (2022). Who picked the best March Madness bracket of all time? https://www.wavy.com/sports/ncaa-bastketball/who-picked-the-best-march-madness-bracket-of-all-time/ accessed: 18 March 2023.

[5] Jeff Sonas, Last-Place Larry, Maggie, and Will Cukierski (2023). March Machine Learning Mania 2023. Kaggle. https://kaggle.com/competitions/march-machine-learning-mania-2023 accessed: 18 March 2023.

[6] Chris Wright (2012). Statistical Predictors of March Madness: An Examination of the NCAA Men's' Basketball Championship. Thesis for Pomona College Economics Department.

[7] Alex Tran and Adam Ginzberg (2014). Making Sense of the Mayhem: Machine Learning and March Madness. Stanford CS229 Final Project paper December 2014.

[8] Levi Franklin (2014). Predicting March Madness: Winning the Office Pool. Stanford CS229 Final Project paper December 2014.

[9] Cody Kocher and Tim Hoblin (2018). Predictive Model for the NCAA Men's Basketball Tournament. Ball State Undergraduate Mathematics Exchange vol 12(1) 15-23.

[10] Zhang, Yudong & Wang, Shuihua & ji, Genlin. (2013). A Rule-Based Model for Bankruptcy Prediction Based on an Improved Genetic Ant Colony Algorithm. Mathematical Problems in Engineering. 2013. https://www.hindawi.com/journals/mpe/2013/753251/