Modeling the Madness: A Machine Learning Approach to Predicting the NCAA Basketball Tournament

Andrew DiLernia and Evan Olawsky

May 2, 2019

1 Introduction

The National Collegiate Athletic Association (NCAA) hosts an annual basketball tournament for Division I teams. This tournament is a single elimination competition which takes place in March after months of regular season play. Prior to the commencement of the tournament, a selection committee determines which teams performed well enough during the regular season to earn entry into the tournament, grouping 68 total teams into 4 separate regions: East, West, South, and Midwest. This results in 63 total games being played each tournament along with 4 additional play-in games. Predicting the outcomes of these tournament games is naturally of great interest to both gamblers and sports fans. The NCAA and Google Cloud jointly host an annual Kaggle competition for the tournament with $\$25,000$ in prize money (Kaggle, 2019). The main criteria for judging model performance was the average log loss across the 63 tournament games. More discussion of the loss function can be found in section 3.

For our project, we used machine learning methods to predict win probabilities for each NCAA Tournament game for the 2019 Kaggle competition. Kaggle provided a wealth of data for modeling tournament games, and we considered a variety of models ranging from a full tree to the more complex neural network. Using a training set and corresponding test set we conducted model selection, and a Linear Discriminant Analysis (LDA) model displayed the best performance in terms of log loss. Using this LDA model for the competition, we finished 485^{th} out of 866 total teams.

2 Data

We used data provided by Kaggle (2019) for use in the competition. There were two main datasets: box score data and rankings data. The box score data includes information on every NCAA basketball game played since the 2002-2003 season. The rankings data gives team rankings from a number of different sources (Kenpom.com, AP poll, Massey rankings, etc.) on either a weekly or a daily basis dating back to the same 2002-2003 season. All in all, we had data on over 82,000 games and around 3.5 million individual team rankings.

The box score data was aggregated over a team's games within a particular season to create season-to-date predictors. For example, the predictors relating to Minnesota's game on January 3rd, 2019 are the averages of all the predictors across all games the Golden Gophers had played in the 2018-2019 season prior to January 3rd. Each team was assigned a ranking equal to their median ranking given in the week prior to a given game. A full list of the predictors is given in the Appendix.

We considered two different response variables for each of our models. We first assume that we have randomly labeled one of the two teams in a given game as the reference team. The first response variable is a binary indicator for whether the reference team won. This response is assumed to have a Bernoulli distribution with parameter π_{ij}, where $\pi_{ij} = P(\text{team } i \text{ beats team } j)$. The second response is a continuous response corresponding to the reference team's margin-of-victory (MOV) in the game. We assume that this follows a continuous distribution with expected value μ_{ij} for a game between team i and team j. Since the Kaggle competition requires win probability estimates, we convert estimated MOV, $\hat{\mu}_{ij}$, into estimated win probabilities, $\hat{\pi}_{ij}$, using a simple logistic regression modeling the actual win/loss outcomes in the training set as the responses and the fitted MOV's as the predictors.

3 Methods

For our predictions, we used two main approaches: modeling the margin of victory of the reference team as a continuous response (positive for a reference team win and negative for a loss), and modeling a binary win indicator for the reference team. Within each of these two approaches,

we considered 10 candidate models: a full tree, pruned tree, random forest, elastic net regression, principal components regression (PCR), linear discriminant analysis (LDA), stochastic gradient boosting, a neural network (NNet), a neural network with feature extraction (NNetF), and support vector machine with a radial basis. We covered all of these methods in class except for the neural network with feature extraction, which was just a neural network fit using principal components as predictors. We note that models which typically are only fit for a continuous response were still fit using the win/loss outcome by just modeling 0's and 1's instead. All models were fit using various packages in R (R Core Team, 2018) . For specific references of the R packages see the References section.

To evaluate competitors, Kaggle used the average log loss given by

$$\text{Log Loss} = -\frac{1}{N}\sum_{i=1}^{N}\left(y_i \log(\hat{y}_i) + (1-y_i)\log(1-\hat{y}_i)\right),$$

where $\hat{y}_i$ is the predicted probability of the reference team winning game i, y_i is an indicator of the reference team winning game i, and N is the total number of games observed where $N = 63$. This choice of penalty penalizes harshly predictions which are both overly confident and incorrect.

For selecting our model for the competition, we used a random subset of 80% of the data for training (roughly 65,000 games) and the remaining 20% of the data as a test set. A summary of each win/loss model's performance on the test set is included in table 1, and for the MOV models in table 2 in the Appendix. Overall the LDA model for the win/loss outcome performed the best on the test set, achieving a log loss of 0.5331. Many of the other win/loss models performed quite similarly. However, given that LDA is a fairly simple model in terms of complexity we used this for our competition predictions. It is notable that SVM did not converge when fitting to the training data for the win/loss response despite running for several days, so it is not included in the summary tables.

4 Results

The LDA model for the win/loss response earned us 485^{th} place out of 866 teams in the Kaggle competition, yielding an average log loss of 0.5103 with some predictions manually fixed by us to

0 or 1, and a log loss of 0.5185 for the true model loss. For reference, a naive model predicting a 50/50 chance for the outcome of each game would have placed 730^{th} with a log loss of 0.6931, so our model performed significantly better than chance alone. The true performance of our models to other Kaggle competitors is displayed in figures 1 and 2 in the appendix. We were fairly satisfied with the performance of the LDA model and our placement in the standings, especially since it appeared that many other competitors fixed a large number of games to improve their performances, manually setting win probabilities equal to one for high seeds. Although this is somewhat risky since a 100% game prediction being incorrect makes it nearly impossible to win, this year there were very few upsets in the tournament which benefited more confident submissions.

Overall, the neural network for the win/loss response performed the best for the NCAA Tournament games, achieving a log loss of 0.4915. This was considerably lower than the 0.5185 log loss achieved by LDA, which was the fourth-best performing model overall. Given that the tournament only evaluated our models based on a sample of 63 games, we interpret the performance results with caution. Relative to the number of games in our test set (roughly 16,000), this is extremely small, so it is not necessarily the case that neural networks are truly better for predicting college basketball games overall.

5 Discussion

Our chosen LDA model finished in the middle of the pack in the Kaggle competition, coming in 485^{th} out of 866 entries. The 2019 tournament was notable for its lack of upsets, as highly-seeded teams generally breezed through the early rounds before meeting other highly-seeded teams. This is why the log losses reported for the 2019 tournament data are substantially lower than log losses that come from the test set. This relative paucity of upsets benefited models which were more aggressive in picking favorites. Our model seems to have been somewhat conservative in assigning win probabilities to top-ranked teams, which perhaps explains our roughly-average finish. Interestingly, our model performed much better in the more standard bracket-scoring paradigm, riding the success of Virginia and Auburn to a 98^{th} percentile finish in ESPN.com's bracket competition.

While the LDA model with binary response performed the best on the test set data, several other models showed very similar performance. Several of those models, most notably the neural

networks for both responses, outperformed the LDA model on the actual tournament data.

We will likely make changes to our model for use in future tournaments. One area in which we might improve is incorporating knowledge about the ordering of games. We may consider giving greater weight to games played more recently, as teams often enter the tournament playing either very well or very poorly. Another change, simpler to implement, would be to exclude early season games, when season-to-date stats are based on much smaller samples and often against poor competition.

We may also consider a mixed effects model that includes a random term for each team in each year. This could help the model deal with teams that consistently over- or under-perform the model's expectations.

References

Greenwell, B., Boehmke, B., Cunningham, J. and Developers, G. (2019), *gbm: Generalized Boosted Regression Models*. R package version 2.1.5.

URL: *https://CRAN.R-project.org/package=gbm*

Kaggle (2019), 'Google cloud ncaa ml competition'.

URL: *https://www.kaggle.com/c/mens-machine-learning-competition-2019*

Karatzoglou, A., Smola, A., Hornik, K. and Zeileis, A. (2004), 'kernlab – an S4 package for kernel methods in R', *Journal of Statistical Software* **11**(9), 1–20.

URL: *http://www.jstatsoft.org/v11/i09/*

Liaw, A. and Wiener, M. (2002), 'Classification and regression by randomforest', *R News* **2**(3), 18–22.

URL: *https://CRAN.R-project.org/doc/Rnews/*

Mevik, B.-H., Wehrens, R. and Liland, K. H. (2018), *pls: Partial Least Squares and Principal Component Regression*. R package version 2.7-0.

URL: *https://CRAN.R-project.org/package=pls*

Meyer, D., Dimitriadou, E., Hornik, K., Weingessel, A. and Leisch, F. (2017), *e1071: Misc Functions of the Department of Statistics, Probability Theory Group (Formerly: E1071), TU Wien*. R package version 1.6-8.

URL: *https://CRAN.R-project.org/package=e1071*

R Core Team (2018), *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria.

URL: *https://www.R-project.org/*

Ripley, B. (2018), *tree: Classification and Regression Trees*. R package version 1.0-39.

URL: *https://CRAN.R-project.org/package=tree*

Simon, N., Friedman, J., Hastie, T. and Tibshirani, R. (2011), 'Regularization paths for cox's proportional hazards model via coordinate descent', *Journal of Statistical Software* **39**(5), 1–13.

URL: *http://www.jstatsoft.org/v39/i05/*

Venables, W. N. and Ripley, B. D. (2002), *Modern Applied Statistics with S*, fourth edn, Springer, New York. ISBN 0-387-95457-0.

URL: *http://www.stats.ox.ac.uk/pub/MASS4*

Appendix

List of Predictors

For each team, on offense and on defense:

- Assist Ratio (AR)
- Effective Field Goal percentage (eFG%)
- Free throw rate (FTR)
- Points per possession (PPP)
- Points scored (Pts)
- Possessions per game (Tempo)
- Rebounding rate (REB)
- True Shooting percentage (TS)
- Turnover ratio (TR)

Other predictors:

- Location of the game (home/neutral/away) for the reference team
- Median ranking in the past week for both teams
- Overall Net Ranking (NET) for both teams

Tables and Figures

	Model	Misclassification Rate	Log Loss
1	LDA	0.2737	0.5331
2	Grad Boost	0.2775	0.5333
3	NNet	0.2743	0.5343
4	PCR	0.2735	0.5364
5	NNetF	0.2838	0.5445
6	Random Forest	0.2794	0.5450
7	Elastic Net	0.2861	0.5515
8	Full Tree	0.3178	0.5979
9	Pruned Tree	0.3178	0.6052

Table 1: Summary of performance of win/loss models for test set.

	Model	Misclassification Rate	Log Loss
1	Grad Boost	0.2755	0.5355
2	PCR	0.2745	0.5363
3	SVM Radial	0.2760	0.5365
4	NNet	0.2764	0.5369
5	NNetF	0.2809	0.5468
6	Elastic Net	0.2828	0.5483
7	Random Forest	0.2752	0.5624
8	Full Tree	0.3177	0.6008
9	Pruned Tree	0.3390	0.6080

Table 2: Summary of performance of MOV models for test set.

	Model	Misclassification Rate	Log Loss
1	NNet	0.3175	0.4915
2	LDA	0.3175	0.5185
3	NNetF	0.3016	0.5207
4	Grad Boost	0.2857	0.5242
5	PCR	0.3175	0.5254
6	Elastic Net	0.2857	0.5343
7	Random Forest	0.3175	0.5898
8	Full Tree	0.3492	0.7305
9	Pruned Tree	0.3492	0.7305

Table 3: Summary of performance of win/loss models for 2019 NCAA Tournament.

	Model	Misclassification Rate	Log Loss
1	NNet	0.2540	0.5017
2	PCR	0.3016	0.5126
3	Grad Boost	0.2698	0.5172
4	Random Forest	0.3333	0.5402
5	SVM Radial	0.3175	0.5668
6	Elastic Net	0.3492	0.5785
7	NNetF	0.3333	0.5819
8	Full Tree	0.3492	0.6485
9	Pruned Tree	0.3492	0.6486

Table 4: Summary of performance of MOV models for 2019 NCAA Tournament.

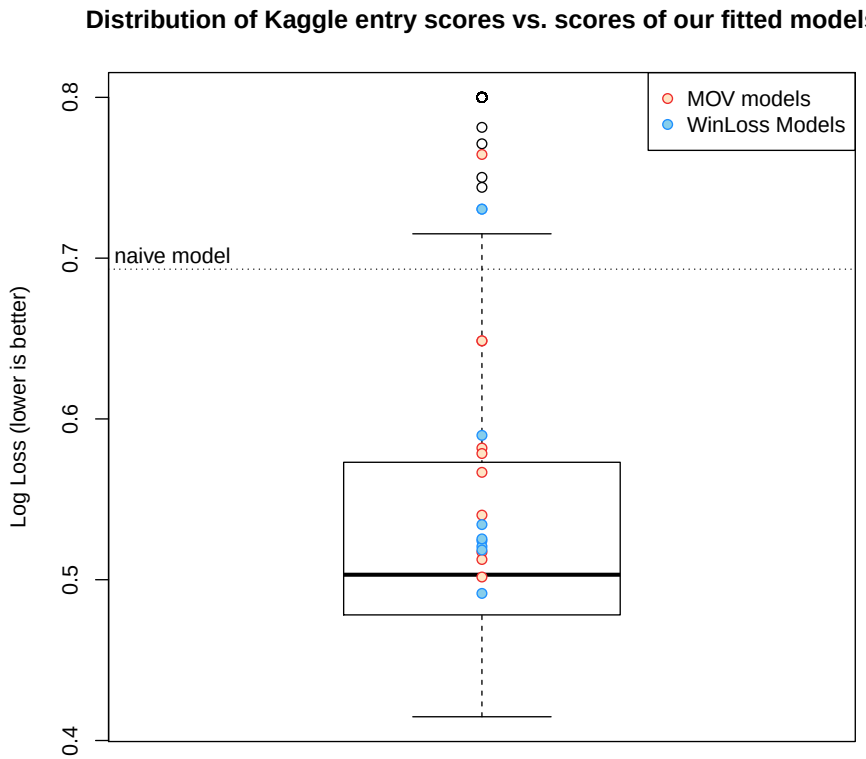

Figure 1: Comparison of our models to all Kaggle submissions

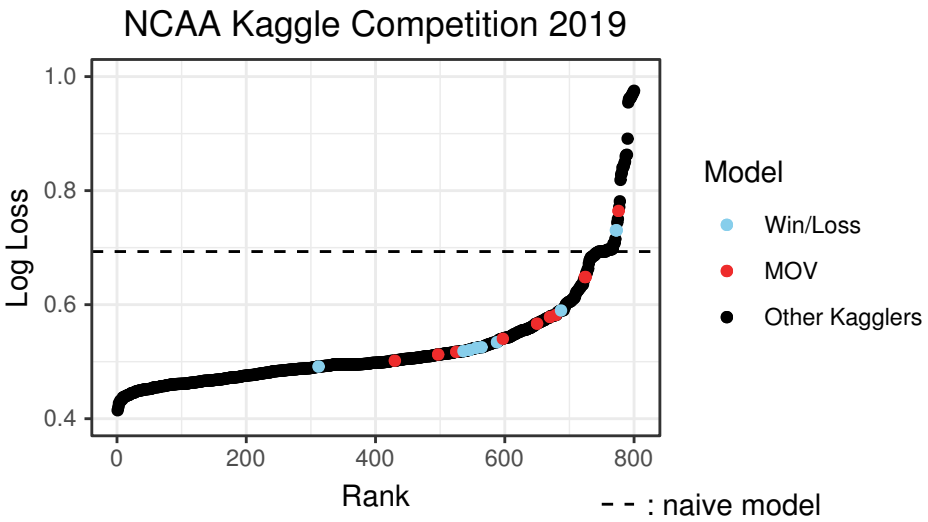

Figure 2: Comparison of our models to all Kaggle submissions