Modelling unpredictability in College Basketball

Hussien Hussien
hhussien@uwo.ca
Western University
London, Ontario, Canada

ACM Reference Format:
Hussien Hussien. 2020. Modelling unpredictability in College Basketball. In ,. ACM, New York, NY, USA, 3 pages.

1 INTRODUCTION

Every year, the NCAA (National Collegiate Athletic Association) hosts a popular college basketball tournament dubbed 'March Madness'. Mostly in March, 68 Division-1 college basketball teams compete in a single-elimination style tournament with the final two teams competing for the national championship. Match-ups are determined by committee using divisions (regions) and seeds (ranking). Teams with seeds 1-4 would all be randomly assigned to separate divisions, teams with seeds 5-9 would all be randomly assigned to separate divisions, and so on until each region has 16 teams.

It has become a wildly popular for fans to predict the outcomes of each game, with an estimated 60 million Americans filling out a bracket each year. In fact, in 2014 Investor Warren Buffet's Berkshire Hathaway and Quicken Loans teamed up to offer $1 Billion to any fan who can perfectly predict the 2014 Men's Bracket. With 67 games, the odds of randomly guessing a bracket is 0.5^{67}.

In this paper, we attempt to predict tournament outcomes by analyzing various performance metrics of each team in a match up. The input to our algorithms are the difference between each matched up teams regular season stats. For instance, we may have an observation that represents the difference between 2 teams' total points scored that season. We then use Logistic Regression, SVM, K-Nearest Neighbors and XGBoost algorithms to output a predicted a match outcome.

2 DATASET AND FEATURES

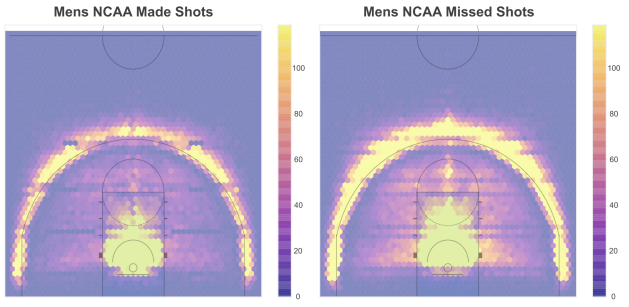

Figure 1: Heat map of court location of made vs. missed shots from the 2019 Season

The datasets used in this project were provided by Google Cloud and NCAA. 25 datasets were provided containing game-level, player-level and team-level data for both the Female and Male leagues. The earliest game we have data on in 1985 Season but, presumably due to technology advances, we have access to much more detailed game statistics in the later seasons. Some of the data sets include:

- Tournament Results: Win/Loss results of each tournament game
- Regular Season Detailed Results: Detailed aggregate results of each team's regular season game performance. of Assists, blocks, 3 pointers, etc made by each team.
- Play-By-Play [2016-2019]: A granular event-level log of each play in that season. Events tracked include timestamped turnovers, 3/2-point shots made/missed, assists, etc by player and location on the court. Figure 1 and 2 include visualizations from this dataset.
- Massey Ordinals: Team and Season level log of each team's ranking according to many well known ranking systems such as TrueSkill, ESPN SOR, Haslametrics and PowerRank. Put together by Kenneth Massey.

These data-sets give interesting insights into the history of March Madness. For instance, a Seed 1 team has never lost to a Seed 16 team. Seed difference is a common statistics analyzed by casual sports fans when filling out brackets.

2.1 Preprocessing and Selection

In order to format the data for modelling, the data is organized by the difference between each team's season performance and seed in a given tournament match up. Where y is a binary outcome variable of each tournament and each $X_1, ..., X_n$ is the difference between each Team1 and Team2's various season stats. For example, a given predictor X_k, may be Team1's Average Blocks per Game - Team2's. This is based on the intuition and insight from data exploration that a historically stronger team should beat a historically weaker team. Thus the better each teams' strength can be gauged the greater predictive power statistical models can capture.

For this project only 2 data-sets are used to build model inputs; Regular Season Detailed Results and Tournament Results. Detailed regular season results are only tracked from 2003 on-wards so only data from the 2003-2019 period are analyzed. In addition to the game statistics provided in the data-set; advanced basketball statistics are also engineered to be predictors. Created using helper functions from publicly available notebooks on Kaggle. All in all, each predictor is a function of each team's difference in statistics like:

- Total Season Points
- Seed
- Offensive efficiency
- Defensive efficiency
- 3-Point Percentage
- Average Possession

To prepare for modelling, each variable is normalized and PCA is applied to reduce dimensions from 39 to 15 columns while capturing 90% of the total variance of the data-set. The data is then split into training set of tournament games from 2003-2015 and testing set of tournament games from 2016-2019.

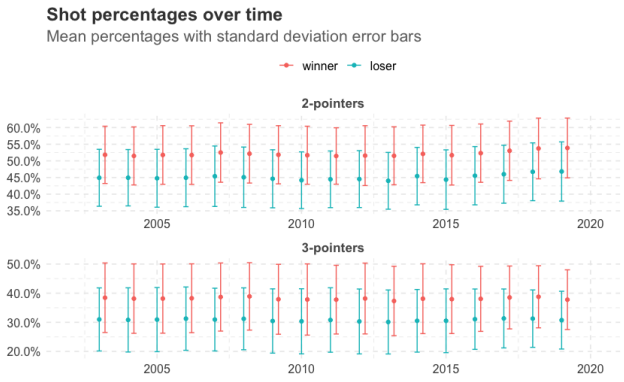

Figure 2: Shot's Made percentages from 2003
Teams have become gradually become more efficient at 2 pointers over time. Generated using code from Martin Henze's *"Jump Shot to Conclusions"* Notebook.

3 METHODS

LOGISTIC REGRESSION
A binary classification algorithm that extends the linear regression model. It outputs probability of a given class via the logit function. Where B_i represents coefficient of the ith predictor.
Logit Function

$$P(X) = \frac{e^{\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}{1+e^{\beta_0+\beta_1 X_1+\ldots+\beta_n X_n}}$$

We choose the l2 penalty parameter, denoted by λ and set it to 0.1. After choosing the regularization method, logistic regression aims to fit it's coefficients to maximize the penalized likelihood function.

SUPPORT VECTOR MACHINES
The support vector machine algorithm classifies data by fitting an n-dimensional hyperplane to the data set, where n is the number of features. Data points are classified based on which side of the hyper plane they fall on. SVM fits a hyperplane to the data that distinctly maximized the distance between data points of each class. In other words, it aims to maximized the margin, distance, between the plane and the two classes. Hyper Parameters chosen:

- **Gamma:** 1e-05
- **Kernel:** rbf
- **Squared l2 penalty,** λ: 1000.0

The cost function for the Support Vector Classifier is the Regularized Hinge Loss function.

$$HingeLoss = [\frac{1}{n}\sum_{i=1}^{n} max(0, 1 - y_i(w * x_i - b))] + \lambda||w||^2$$

K-NEAREST NEIGHBORS
K-Nearest neighbors is a non-parametric algorithm used in this case for classification. It works by simply classifying the a data point by a majority vote of it's k-neighbors. K-neighbors are classified by the k-data points that have the smallest Eucidian distance to the data point we are trying to output a prediction for. If the majority of a datapoint's K-neighbors fall into a certain class than that will be our output for $\hat{y}$. After testing parameters different k's from 1 - 30, our grid search algorithms showed that $K = 9$ gives us the greatest AUC.

XGBOOST
XGBoost, short for Extreme Gradient Boosting, is a derivation of the Gradient Boosted Tree Algorithm with regularization. The ensemble technique, Boosting, works by creating new models that correct the errors made by existing models. All models are then compounded until there are no more improvements to be made. Gradient Boosting uses the approach of creating new models that predict the residuals of the prior models then compound them to make a prediction. It uses a gradient descent algorithm to minimze it's loss function, hence the name Gradient Boosting. XGBoost is framework that implements a the gradient boosted algorithms with additional system, model and algorithm improvements. Hyper parameters are chosen to optimize with learning rate, regularization, tree-depth and tree-pruning for best model performance. Hyper Parameters tuned to:

- **Gamma:** 0
- **Learning Rate:** 0.03
- **Max Depth:** 6
- **Lambda:** 6

XGBoost minimizes the Log Loss function which is described in the following section on Evaluation.

4 RESULTS

4.1 Evaluation

We the primary metric we use to evaluate model performance on both train and test sets is Logarithmic Loss:

$$\text{LogLoss} = -\frac{1}{n}\sum_{i=1}^{n} [y_i \log(\hat{y}_i) + (1 - y_i) \log(1 - \hat{y}_i)]$$
$$where$$

- n is the number of games played
- $\hat{y}$ is the predicted probability of team 1 beating team 2
- y_i is 1 if team 1 wins, 0 if team 2 wins

The use of the logarithm provides extreme punishments for being both confident and wrong. In the worst possible case, a prediction that something is true when it is actually false will add an infinite amount to your error score. In order to prevent this, predictions are bounded away from the extremes by a small value.

We also analyze a variety of secondary metrics like accuracy (percentage of correct predictions out of all predictions) and F-Measure. The F-Measure can be described as the harmonic mean of both precision and recall. It is at it's highest at 1, which is perfect precision and recall.

$$F - Measure = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4.2 Training Results

Table 1: CV Train Results on 2003-2015 Seasons

Model	Log Loss
Logistic Regression	0.570
Support Vector Machine	0.5681
K-Nearest Neighbors	0.3225
XGBoost	0.2380

4.3 Out-Of-Sample Results

Table 2: Test Results on 2016-2019 Seasons

Model	Log Loss	Accuracy	F-Measure
Logistic Regression	0.5700	0.6902	0.6903
Support Vector Machine	0.5684	0.6903	0.6903
K-Nearest Neighbors	0.3225	0.8731	0.8731
XGBoost	0.2315	0.9011	0.9064

4.4 Discussion

Clearly, XGBoost produces the best performance on all of our test/train evaluation metrics. It mis-classed only 24 games out of the 268 games played over the test period as seen in figure 3. It would be interesting to see which specific match-ups it got right and wrong but time constrains. The model doesn't appear to be over-fit since there is no large discrepancy between test and training results. Proactive measures were taken to avoid over fitting such as dimensional reduction, regularization and Cross-validation.

5 CONCLUSION/FUTURE WORK

The predictive power of these models was not expected. Based on previous winners of Kaggle competition around this dataset, it appears these results are significant. The winners of the 2019 challenge achieved a logloss of 0.44.

Future work is extensive; in the future I want to see whether we predicted any upsets. There is so much more data to take into account that I had time to look at but I would love to get features out of them in the future.

6 REFERENCES

Mulla, Rob. "NCAA Basketball Court Plot Helper Functions." Kaggle, Kaggle, 27 Feb. 2020, www.kaggle.com/robikscube/ncaa-basketball-court-plot-helper-functions.

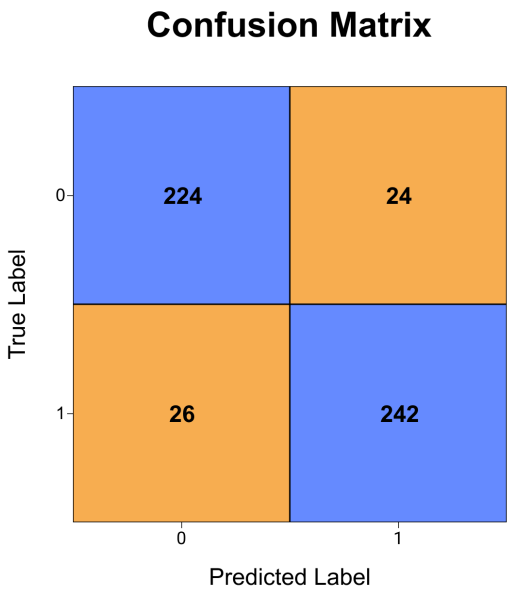

Figure 3: Confusion Matrix of XGBoost predictions on 2016-2019 Tournament

Henze, Martin. "Jump Shot to Conclusions - March Madness EDA." Kaggle, Kaggle, 8 Mar. 2020, www.kaggle.com/headsortails/jump-shot-to-conclusions-march-madness-eda.
Nathan, Laksan. "Feature Engineering with Advanced Stats." Kaggle, Kaggle, 7 Mar. 2018, www.kaggle.com/lnatml/feature-engineering-with-advanced-stats. NCAA, Kaggle.com. "Google Cloud amp; NCAA® ML Competition 2020-NCAAM." Google Cloud NCAA March Madness 2020 Division-1 Mens Tournament, Kaggle.com, Feb. 2020, www.kaggle.com/c/google-cloud-ncaa-march-madness-2020-division-1-mens-tournament/data.
Takahashi, Hiromu. "2020 Basic Starter Kernel." Kaggle, Kaggle, 15 Feb. 2020, www.kaggle.com/hiromoon166/2020-basic-starter-kernel.
Geiling, Natasha. "When Did Filling Out A March Madness Bracket Become Popular?" Smithsonian.com, Smithsonian Institution, 20 Mar. 2014, www.smithsonianmag.com/history/when-did-filling-out-march-madness-bracket-become-popular-180950162/.