NBA Winning Rate and Season Record Prediction

Penghao Weng
New York University Shanghai
pw1298@nyu.edu

Dennis Hu
New York University Shanghai
sh5701@nyu.edu

ABSTRACT

Statistics driven performance analysis has been gaining ground in all major sports leagues. Particularly, the National Basketball Association (NBA) is at the frontlines of transitioning to data-driven approaches. While various statistical models for predicting the MVP candidate and the championship possibilities have been developed, the field of single-game winning rate prediction has gone rather under-explored. Here we report our efforts in achieving accuracies of predicting single-game outcomes comparable to preexisting machine learning models. After formatting the raw dataset with aggregate functions that yield recent performance, we evaluated across five different models to generate our best guess at the W/L result. Since our model takes into account the lineups of each team, it also offers quantitative insights into NBA team's roster management.

I. INTRODUCTION

The Golden State Warriors have dominated the past decade of the NBA, mainly by virtue of their innovation combining statistical analysis with basketball operations. This gave birth to a new generation of superstars such as Stephen Curry and Luka Doncic. With more and more teams applying statistical models and Machine Learning algorithms to devising their game strategies, there has been a major shift in the way that the game of basketball is played and perceived. Instead of relying on instincts to make game decisions, it has been proven that utilizing the power of Machine Learning models gives more of a competitive edge to a team over those who do not. Each of the 30 teams in the NBA are now in an arms race to build the best model possible in predicting winning rates in order to help evaluate players and make in-game adjustments.

However, NBA games are notoriously difficult to predict. The average score differential is about 10 points with average scores varying from 100-112 points, depending on the season. Still, the best machine learning models are only able to achieve 65-70% accuracy in predicting the game outcome. This can be attributed to a whole range of reasons, from intra-player chemistry to play-type specific restrictions, all of which statistical models are unable to capture. Our model aims to fill that gap by considering team metrics that truly indicate their current power, instead of regressing only on historic data. Additionally, we will train and fine-tune different possible models to see which attains the highest score and stability.

II. DATA PREPROCESSING

2.1 Source and overview

Our datasets of 'NBA games data' was obtained from Kaggle provided by Nathan Lauga, Data Scientist at Caisse d'épargne Bordeaux, Nouvelle-Aquitaine, France. For these datasets, he used the nba stats website, which is the official website for NBA stats. There are 5 datasets:

- 'games.csv': contains all games from the 2004 season to last update with the date, teams and some details including final score, team box scores, etc.
- 'games_details.csv': details of games dataset, all individual statistics of players for a given game.
- 'players.csv': players details (names, height, weight, year pro, etc.)
- 'ranking.csv': ranking of NBA given a day (split into west and east on CONFERENCE column)
- 'teams.csv': all teams of NBA, each with unique "TEAM_ID" keys.

For the datasets above, two of them are at the center of our focus, which are 'games.csv', 'ranking.csv'. The 'games' dataset contains 21 feature columns and 24196 rows for the data of all games from the 2004 season to the last updated date teams and some details like number of points, etc. While the 'ranking.csv' dataset contains the number of games played in the season, along with the rank of playing as home team and playing as road team during the season.

2.2 Explanatory analysis

i. Season distribution

We first take a look at the season distribution of the 'games' dataset. We get the value counts for the column "SEASON" of 'games.csv' and draw a bar chart for the season distribution. As we could capture from Fig. 1, The

2011–12 NBA season was the 66th season of the National Basketball Association (NBA), which began with the signing of a new collective bargaining agreement (CBA) between the owners of the 30 NBA teams and the NBA's players. The previous CBA, which was ratified in 2005, resulting in a lockout. With the new deal in place, the regular season was shortened from the normal 82 games per team to 66, because of nearly two months of inactivity.

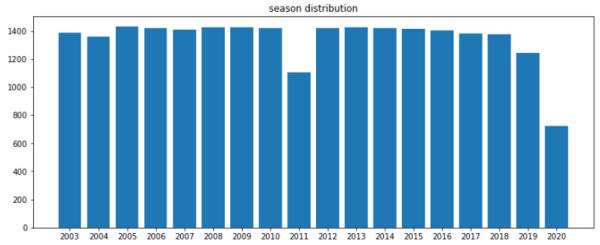

fig. 1. Season Distribution Bar Chart

ii. Home team win probability

For 'games.csv' dataset already gives a column "HOME_TEAM_WINS" with values 0 (home team lost) and 1 (home team won) which contributes to our research of forecasting winner of one NBA game, we first choose to see the winning percentage of playing as a home team, so we draw a bar chart (see fig. 2 below), which shows that choosing the winner as the home team will have a 59% to identify the winner.

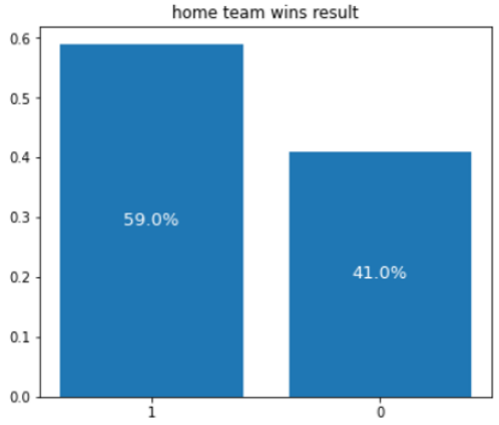

Fig. 2. Result of home team wins

2.3 Formatting and Cleansing

To predict the winner of a game based on a whole team's metrics, we only need two datasets: "games.csv" and "ranking.csv". For each game, we need rows of past games that contain the correlated data of both the home team and the road team that we want to predict.

i. Format 'rankings.csv'

For 'ranking.csv', the record of ranking of a team plays as home team and road team are stored in columns "HOME_RECORD" and "ROAD_RECORD". However, the ranking stores in the format "numbers of winning – numbers of losing" in both columns, so we use the formula $\frac{numbers\ of\ wining}{numbers\ of\ wining+numbers\ of\ losing}$ to transform the ranking columns into floats, which helps with the following work of calculating past game data. Then we use two functions "get_team_ranking_befor_date" and "get_team_ranking_before_game" to extract context form "TEAM_ID" and "STANDINGDATE" columns to return a data frame with the team id, number of games played, win percentage, home and road record for current and previous season based on the date. We can use the functions to calculate the team ranking before n days from current date. Additionally, we drop the column "RETURNTOPLAY" which contains 99% "null" data and only refer to the video records of the games that are uncorrelated to results of prediction.

ii. Get game statistics

Same for getting team rank functions, we also wrote two functions "get_team_stats_before_date" and "get_team_stats_before_game" to extract important game statistics such as "PTS" (number of points scored), "FG_PCT" (field goal percentage), "FT_PCT" (free throw percentage), "FG3_PCT" (three-point percentage), "AST" (assists) and "REB" (rebounds) from the past games base on date. These feature columns are the most significant components of our data analysis. We can use the functions to calculate the team game statistics before n days from current date.

iii. Combine team ranking stats and game stats

We prepared two datasets 'games.csv' and 'rankings.csv' to combine team ranking statistics and game statistics for the past games before our target forecast game. We finally choose to combine the statistics from 3 games before and 10 games before to improve our accuracy of prediction. Then we store the formatted game data set after season 2010 into the csv file 'games_formated.csv'. We check the data frame information with no null data. The suffixws "_3g" and "_10g" represents the data 3 games before and 10 games before.

iv. Define feature / target

We choose some features manually from 'games_formated.csv', drop the columns such as "GAMES_ID", "GAME_DATE_EST" and "SEASON" which are definitely uncorrelated to our predictions as first cleanup preprocess, and choose the column "HOME_TEAM_WINS" as our classification target. Then we used the Seaborn library to draw the correlation between different features (See Fig. 3 below). We finally choose the features whose correlation coefficient with our target "HOME_TEAM_WINS" is larger than 0.1. Numbers larger than 0.1 will lead to an underfit problem and numbers smaller than 0.1 will lead to an overfit problem of the model after our testing.

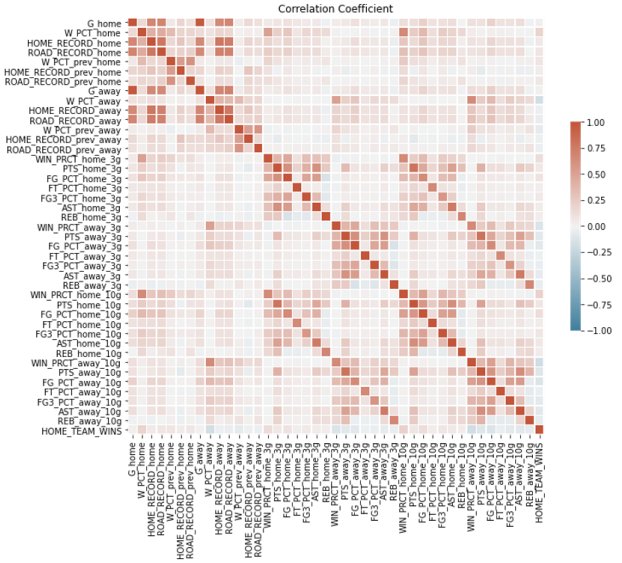

Fig 3. Correlation coefficient of features

III. MODEL TRAINING

3.1 Training and Testing Dataset

We chose various cutoff points between the given 2003 - 2020 seasons to be our train / test split. At first, we only took into consideration seasons that were played after the year 2010, due to the fact that around the 2010s the league went through a seismic shift and play type changed dramatically. With a training set starting from the 2010 season, we came to yield a max test accuracy of 64.8% with the random forest model (which we will go into in its own section). After that, we tried to expand the training set by including the seasons from 2003 to 2009. The result was surprising: we actually got a worse accuracy score. A possible contributing factor to this is that: the NBA did not complete its advanced stats compiling before 2010, and the seasons before that will see many missing values from each box score column. Eventually, we decided on the 2010 - 2018 season to be our training set, and 2019 season to be our testing set.

3.2 Models and Evaluation

a) Logistic regression

Since we still have a fair amount of collinearity in our dataset, we experimented with both Ridge Regression and Lasso Regression in an attempt to do some indirect feature selection. The key difference between Ridge Regression (which uses L2 loss) and Lasso Regression (which uses L1 loss) is that Lasso shrinks the less important features to zero, while ridge shrinks the features proportionally. In our case, since the dataset contained common-sensically irrelevant features like minutes played, DNPs, etc. Lasso would work better for our prediction.

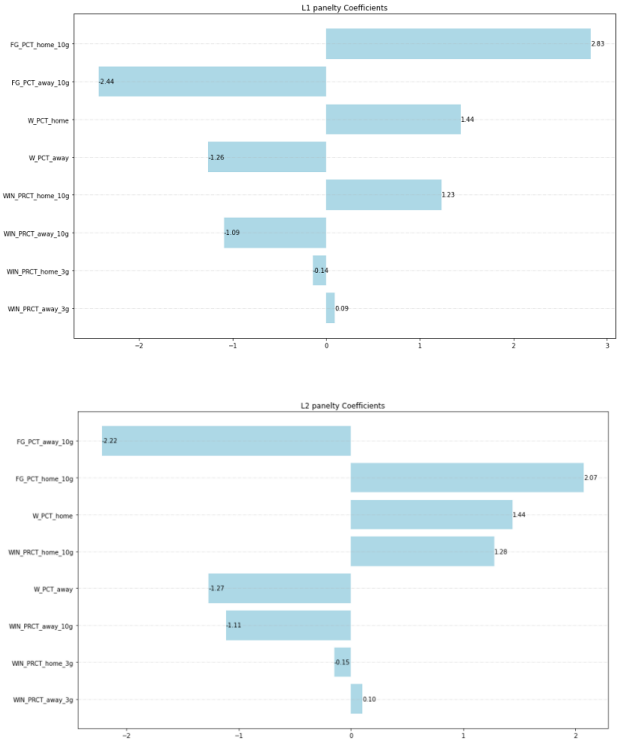

Fig 4. L1 and L2 Regularization Coefficients

As can be seen from Fig. 5, the coefficient distribution of Lasso is rather evenly spread out, which contributes to it yielding a higher accuracy score, in line with our expectations before the training. The test accuracy for L1 penalty is 0.635, and the test accuracy for L2 is 0.633. To achieve those best testing results, our tuned regularization coefficient for L1 is 0.938 and that for L2 is 1.441.

b) Random Forest

As an ensemble method, the random forest constructs numerous trees and bags the predictions yielded by each weak classifying tree. Our random forest tunes the max depth parameter, which limits the complexity of each tree. As can be seen in Fig 6, when this parameter goes beyond 7, testing accuracy does not improve and we have an overfitting problem.

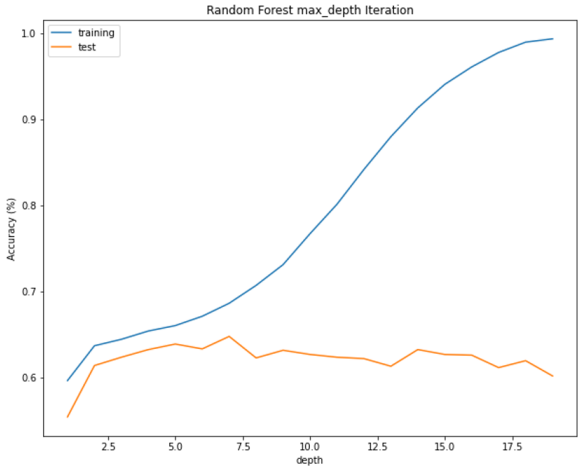

Fig 5. Max_depth iteration of Random Forest

Therefore, we choose 7 as the optimal max depth to train our forest. Another hyper parameter is the total number of trees to construct, so we also iterated through 1 to 200 estimators.

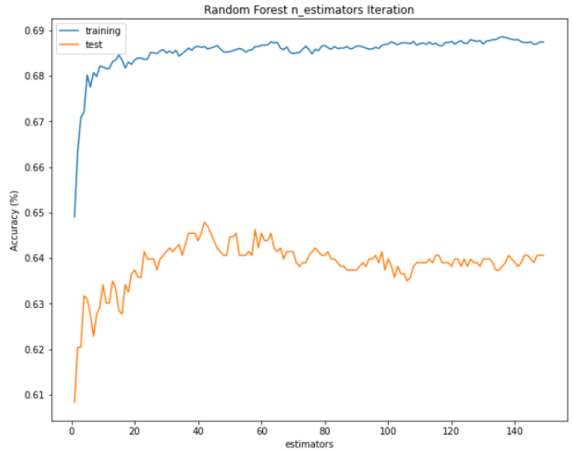

Fig 6. N trees iteration of Random Forest

From the above graph, we can conclude that gowing 42 trees gives the best testing accuracy.

Given the two tuned hyper parameters, our random forest model turned out to perform the best among our four models, tapping out a testing accuracy at 64.8%.

c) Support Vector Machine

Here we gave another attempt to train our dataset on a Support Vector Machine. It has been placed high hope on since it is known to produce significant accuracy with much less computation power. We tried both a soft margin and a hard margin SVM. With C as our penalty coefficient, we found that a soft margin SVM takes much less time to train than a dard margin one on our dataset, while not compromising accuracy.

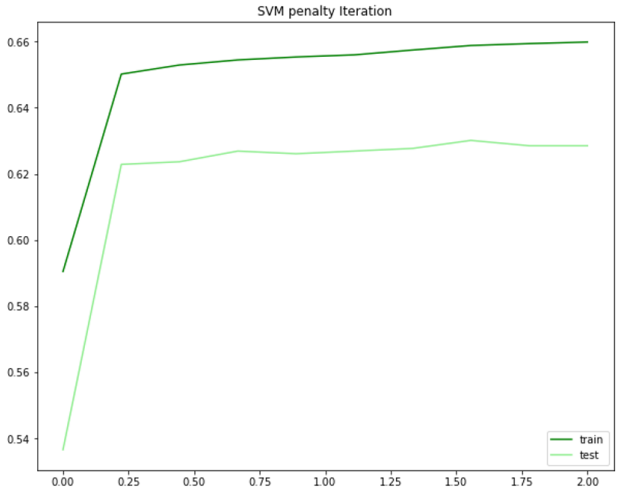

Fig 7. Penalty iteration of SVM

As can be seen from Figure 8, the penalty make the machine perform best when set around 1.5. With a 1.5 penalty we got the highest testing accuracy by SVM at 63.0%.

d) Naive Bayes

Given the assumption of independence, Naive Bayes is expected to achieve the same accuracy with less training data and computational power. We tried the Gaussian, multinomial and Bernoulli Naive Bayes, among which the multinomial model performed especially poorly, only recording the accuracy slightly higher than random guess: 53.8%. Gaussian Naive Bayes turned out to be the best performer of the three, details can be seen in Fig 9:

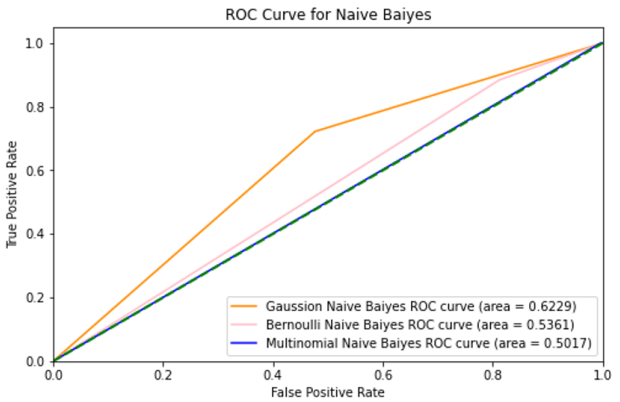

Fig 8. ROC curves of NB methods

However, the Naive Bayes model still performed fairly subpar compared to other models. This is highly likely due to the existence of collinearity in our datasets.

e) Gradient Boosting

Gradient Boosting is another model that converts weak classifiers into strong classifiers. The three tunable hyper parameters are: depth, learning rate and number of estimators. We first look at tuning the depth parameter.

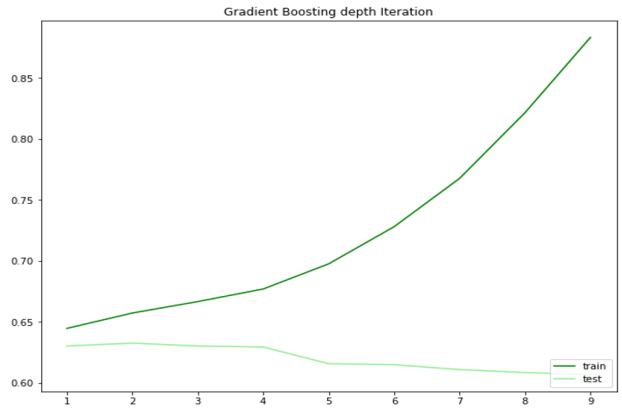

Fig 9. Depth iteration of GB

Surprisingly, the best depth is actually achieved at 2. With the depth parameter relatively fixed, we iterate through possible learning rates and plot it against the accuracy achieved:

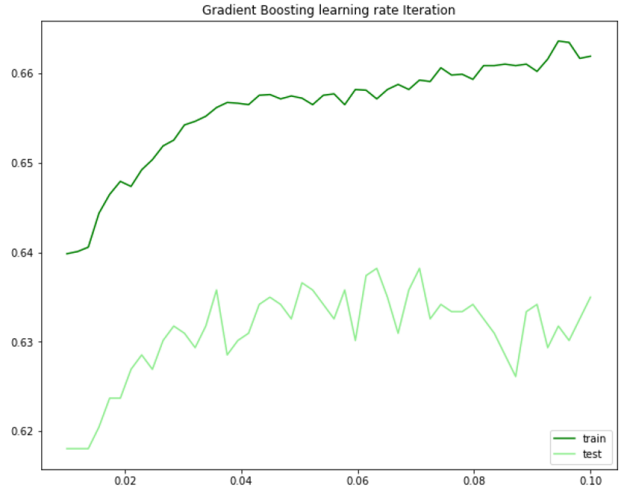

Fig 10. Learning Rate iteration of GB

We then found that the best learning rate is set at 0.0633. With the same iteration technique, we found the best number of estimators to be 105. After tuning all three hyper parameters, we found that Gradient Boosting actually gives the second best testing accuracy, right after Random Forest at 64.3%.

IV. RESULTS

After testing out and training all five models, we drew the aggregated ROC curve for all the models in Fig 12:

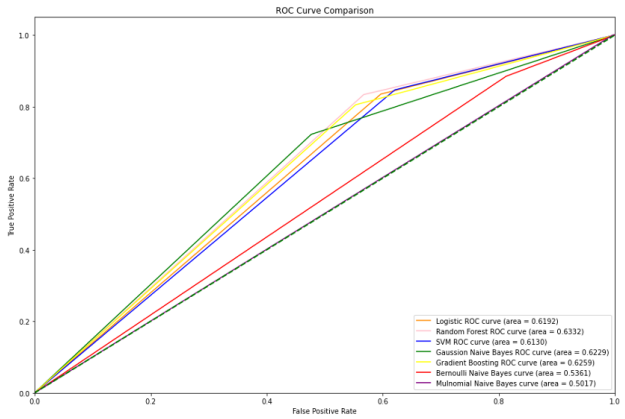

Fig 11.ROC curves of all models

As discussed above, Random Forest yields the biggest area under curve, with Gradient Boosting coming in second, and Naive Bayes ranking last. Taking into consideration the characteristics of our special dataset, it is likely that the regulation disparities in the early NBA seasons resulted in a chronological imbalance. As Random Forest handles empty values quite smooth and automatically, this feature also contributed to it being crowned the most effective in solving our problem. Similarly, although Naive Bayes can be sufficiently effective in predicting values where the training features are independent, our problem does not turn out to fit that requirement. As a result, one is better off simply betting on the nome team to win than utilizing the Naive Bayes model.

V. CONCLUSION AND FUTURE WORK

With the aim of correctly predicting the winner of a given NBA matchup, we applied five different machine learning models and found that Random Forest came out as the best performer on our dataset. While the eventual outcome accuracy is not astonishing, given that the best accuracy achieved by professional NBA data scientists is still hovering the 70% mark, our models also did not disappoint. More importantly, the biggest innovation made in this project is during the data preprocessing part: instead of the conventional approach which is bundling the entire previous season as a training set, our aggregate functions made it possible to predict game outcomes on a rolling basis. In this way, future modifications can be made to work towards a new form of predicting pipeline that is more sensitive to roster changes in the NBA. In an ever-more fast-paced league, this sensitivity to the most minor changes in team power ranking can boast long-term influences to the way organizations make decisions.

Together with our efforts to make predictions on a team-based approach, we also started out with a player-based approach, which is even more sensitive to changes in player roster to the point where it allows

game-by-game adjustments. However, after writing out the functions to retrieve from raw datas the most recent statistics of each player, we ran the function for weeks on end trying to generate the player-based data frame on which we can train. Eventually we downscaled the training sets and still were not able to train the data due to limited computing power. We plan to keep exploring more efficient ways to enable our player-based algorithm in the summer.

VI. REFERENCES

[1] Avalon, Grant, et al. *Various Machine Learning Approaches to Predicting NBA Score Margins*. , 2016.

[2] "Basketball Statistics and History | Basketball-Reference.com." *Basketball-Reference.com*, 2000, www.basketball-reference.com.

[3] "Case Study: Machine Learning Applications - Making NBA Predictions." *Oursky Posts*, 26 Nov. 2019, blog.oursky.com/2019/11/26/machine-learning-applications-nba-predictions/#training%20model. Accessed 15 May 2021.

[4] "NBA Game Prediction Model V1." *Kaggle.com*, www.kaggle.com/jbthornt02/nba-game-prediction-model-v1. Accessed 15 May 2021.

[5] Paine, Neil. "How Our NBA Predictions Work." *FiveThirtyEight*, 18 Dec. 2018, fivethirtyeight.com/methodology/how-our-nba-predictions-work/. Accessed 15 May 2021.

[6] Singh, Harshdeep. "Understanding Gradient Boosting Machines." *Towards Data Science*, Towards Data Science, 3 Nov. 2018, towardsdatascience.com/understanding-gradient-boosting-machines-9be756fe76ab

[7] Singh, Prastuti, and Bai Wang. *NBA Game Predictions Based on Player Chemistry*.