Honors Projects in Data Science   Senior Honors Projects

4-2021

Predicting the Outcome of NBA Games

Matthew Houde

Predicting the Outcome of NBA Games

BY Matthew Houde

ADVISOR • Alan Olinsky
EDITORIAL REVEIWER • Suhong Li

_____

Table of Contents

ABSTRACT

The aim of the project is to create a machine learning model to predict NBA games. The purpose is to build upon and improve existing models. Research into other predictive sports models and machine learning techniques was conducted to understand what is currently being done to predict NBA games and how effective it is in doing so. After a thorough literary review, the model was created using Python and a variety of machine learning techniques. The dataset used had an array of team statistics for both the home and away team for each corresponding matchup and two supporting features were feature engineered. Six different models were tested on the training set: Logistic Regression, Random Forest Classifier, K Neighbors Classifier, Support Vector Classifier, Gaussian Naïve Bayes, and XGBoost Classifier. The best performing model from this group was Gaussian Naïve Bayes. An exhaustive grid search was done to tune the hyperparameter and refine the model. The final model reported an average accuracy of 65.1%, which means that it can predict the outcome of an NBA game about 65% of the time. The code for the project is also available on GitHub: https://github.com/mhoude1/NBA_Model

INTRODUCTION

The National Basketball Association is the largest basketball association in the world and brings in an estimated eight billion dollars a year. As a collective whole, the NBA generates immense revenue and data. The data pertains to teams and players alike. This data is beneficial for analytics and game strategy. Sports teams and organizations have adapted data analytics and modeling to gain every competitive advantage they can get. A lot of decisions made in a basketball game are determined by statistics and analytics. These analytics could very well make the difference in the outcome of a game.

Sport predictions have been becoming more relevant in the industry. Before every game, analysts will display a prediction as to who will win and the margin of victory. Another industry that has stemmed from sport predictions is the betting industry. Legal betting platforms have been appearing and expanding for years. An accurate model is imperative to predict games and quantify all metrics in a basketball game to ensure the betting process is equitable. There are many uses to machine learning models in sports and their usefulness is only growing.

This thesis is divided up into a few succinct sections. The Literature Review section will fully detail the machine learning techniques used in this model and offer an overview of existing NBA prediction models. The Methodology section goes into depth on the process of creating the model and follows the Cross-Industry Process for Data Mining (CRISP DM) model for a detailed structure flow. The Results section presents the results of the models, the accuracy reports, and the final confusion matrix on how well it predicts the winners and losers of NBA games. After presenting the results, a Discussion of the results is in the next section. Finally, this is followed by a Conclusion that wraps up the outcome of the project, what was learned, how it could be improved, and some potential next steps.

LITERATURE REVIEW

<u>Machine Learning Techniques</u>

There are a few important machine learning techniques that are fundamental in creating a successful model. One of such methods that is fundamental to understand is cross validation. Cross validation is the dividing of a dataset into k number of groups of sample sizes. For example, if there were ten folds, nine folds would make up the training set and one-fold would comprise the test set. Each individual fold would have the opportunity to be the test set as the model will be trained ten separate times to account for each k number of folds. It is a crucial step in modeling as it creates an output that is less biased than other methods and guarantees the use of the entire dataset as the test data. The user also has the option to stratify the data which is making sure that each fold has the same proportion of observations within a given categorical value. Picture below is a simple model depicting how cross validation is implemented on a set of data.
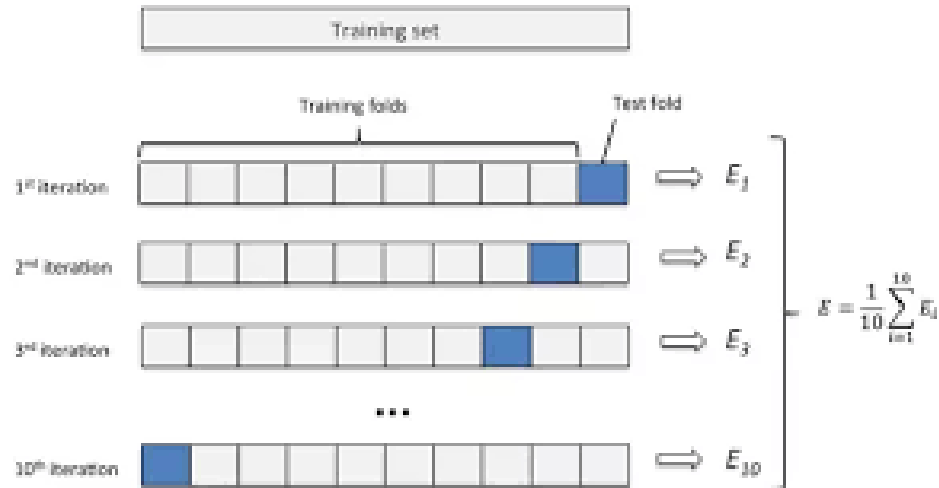

Figure 1 – Cross Validation Diagram

Most of the model will be automated in a ML pipeline. The pipeline is helpful in automating the machine learning workflow. It enables the data to be transformed and correlated into a

model that can be analyzed to achieve outputs. It makes the process of inputting data into the ML model fully automated.

Once the model is created, it needs to be tuned and tested to achieve greater accuracy and results. The hyperparameters are values that are used to control the learning process and need to be optimized and tuned. One effective way of tuning the hyperparameters is through the process of grid search. Grid search is a process that searches exhaustively through a manually specified subset of the hyperparameter space of the targeted algorithm. This allows the user to tune parameters such as the kernel, C, gamma, and more.

All the data must be in a machine-readable format to be appropriately processed by the algorithms. The categorical variables must be one hot encoded and the numerical variables need to be standardized. There are two methods to standardize data to minimize the effects of outliers and make sure the data is internally consistent. The first method is standardization is the process of putting data on a standard scale. The data will have a mean of 0 and a standard deviation of 1. The second one is normalization is the process of making the data normal with a fixed range of 0 – 1. Standardization is good for regression models and normalization is good for models such as k-nearest neighbors or artificial neural network.

Machine Learning Classification Models

The problem at hand is a binary classification problem as there are two potential outcomes, win or loss. There is an array of potential classification algorithms that can be used and that is why six different models will be tested. The six models are Logistic Regression, Random Forest Classifier, K Neighbors Classifier, Support Vector Classifier, Gaussian Naïve Bayes, and XGBoost Classifier. Each model uses a unique approach to classify the outcome of the game. The best performing model form this group will be the final model chosen for deployment.

Logistic Regression –

Logistic regression is used when one is trying to predict a dependent categorical variable. The two outcomes for a binary regression model are 1 and 0. Some things that could potentially be predicted are win or lose, spam or not spam, and so on. There needs to be a decision boundary or threshold for the model which separates the two outcomes. The obtained estimated probability is classified into classes which can be linear or non-linear. The coefficients and beta values must be estimated from the training data. Once the probabilities are set, use the decision boundary to determine the outcome. For example, if probability of male is less than 0.5 then the outcome is 0 and if P(male) >= 0.5 then the outcome is 1.
The formula is X = [x0 x1] = [1 IP-Address].

There are two different regression techniques, LASSO and RIDGE, that can be used to reduce model complexity and prevent over fitting. The Lasso method, L1, stands for least absolute shrinkage and selection operator. It helps reduce overfitting and it can help in proper feature selection. The Ridge method, L2, alters the cost (loss) function by adding a penalty equal to the square of the magnitude of the coefficients. This shrinks the coefficients and helps reduce model complexity and multicollinearity. Multicollinearity refers to the high intercorrelations among two or more independent variables in a regression model and this can undermine the statistical significance of an independent variable (Tutorials Point).
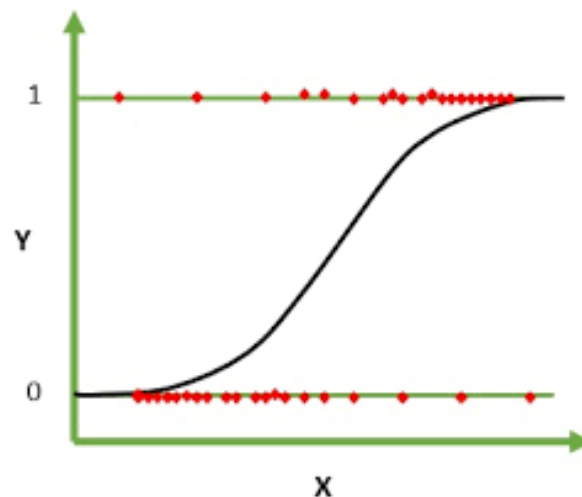

Figure 2 – Logistic Regression Diagram

Random Forest Classifier –

A random forest is an ensemble learning method for classification and regression. It builds multiple decision trees and merges them together to get a more accurate prediction. In a random forest, only a subset of the features are taken into consideration by the algorithm for splitting a node. The model will also rank the importance of each feature in making the final decision. The figure below is a random forest with two trees (Tutorials Point).
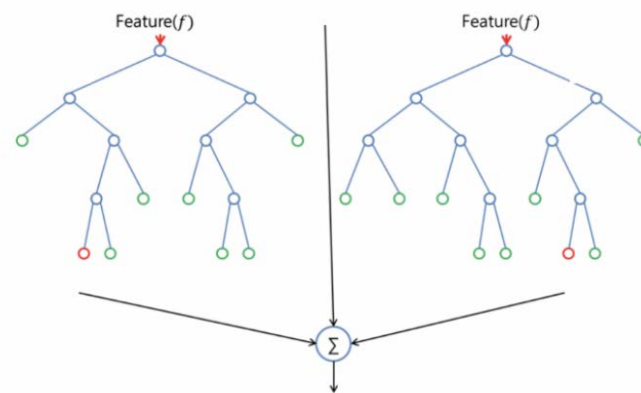

Figure 3 – Random Forest Classifier Diagram

K Neighbors Classifier –

KNeighborsClassifier is a classification algorithm that implements the k-nearest neighbors' vote. The K stands for the number of closest neighbors to the new data point to classify. If K = 3, then the three closest values to the data point would decide which class it is classified as. If there is an equal number of classes close to the data point, the class will be chosen by the lowest average distance. It is an algorithm good at classification and should be successful in prediction the outcome of a match (Srivastava)

Support Vector Classifier –

An SVM model is a representation of different classes in a hyperplane in multidimensional space. The goal of the model is to divide the datasets into classes to find the maximum marginal hyperplane. The support vectors are the data points that are closest to the hyperplane

and the hyperplane is the plane that which is divided to make a decision for the model. SVM is a capable model, and it can handle multiple continuous and categorical variables. (Tutorials Point).

Gaussian Naïve Bayes –

Naïve Bayes is a classification technique with a strong assumption that all the predictors are independent to each other. The main interest is to find the posterior probabilities, or the probability of a label given to some observed features. Gaussian is used when the features have continuous values, and the values follow a gaussian distribution or normal distribution (Sklearn).

XGBoost Classifier –

XGBoost is an optimized distributed gradient boosting library designed to be highly efficient. It uses the Gradient Boosting framework and supplies a parallel tree boosting. Gradient Boosting is a prediction model in the form of an ensemble of weak prediction models, typically decision trees. The model is built in a stage-wise fashion and allows the optimization of an arbitrary differentiable loss function. (Sklearn).

Machine Learning Evaluation Metrics

Confusion Matrix –

A major evaluation metric used for classification models is the confusion matrix. It is used to determine the performance of a classification model on a set of test data. It gives the user an understanding of the actual vs. the predicted values and the overall accuracy. The matrix is formed of True Positive, True Negative, False Positive, and False Negative. True Positive is when the model correctly predicts the positive class and True Negative is when the model correctly predicts the negative class. In a win loss classification model, a True Positive would be represented by the model predicting a team to win and the team does in fact win. True Negative would be the opposite if the team was predicted to lose and does lose. The other two tracked stats are False Positive and False Negative. A False Positive occurs when the team is predicted to win but loses and False Negative would be the reverse of that scenario. The False

Positive would also be classified as a Type I Error and the False Negative would be a Type II Error.

There are a few key metrics that are used to evaluate the model that stem from the confusion matrix. These metrics and how they are calculated is detailed below.

- Accuracy = (TP + FP) / (FP + TP)
- Precision = TP / (TP + FP)
- Recall = TP / (TP + FN)
- F1 = 2 * (Recall * Precision) / (Recall + Precision)

Accuracy is a simple statistic that stands for the overall accuracy of the model. It takes the correctly predicted data points and divides it against the incorrectly predicted values. Precision tells the user how many of the correctly predicted cases turned out to be positive. It is a useful metric when FP (False Positive) is a higher concern than FN (False Negative). Recall tells the user how many of the actual positive cases were predicted correctly with the model and it is useful when FN trumps FP. Finally, the F1 score is a harmonic mean of precision and recall, it captures both trends in a single metric and is scored [0,1] (David Dalisay).

Figure 4 – Confusion Matrix Diagram

AUC - ROC –

AUC - ROC stands for Area Under the Curve - Receiver Operating Characteristics. AUC – ROC is another performance measurement for classification problems. ROC is a probability curve and AUC represent the degree or measure of separability. It tells us how well the model can distinguish between classes.
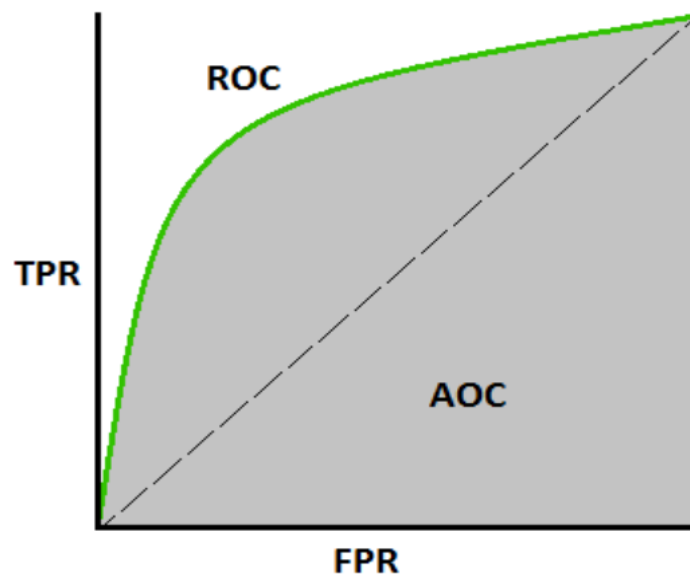
Figure 5 – AUC – ROC Diagram

The ROC curve is plotted with TPR (True Positive Rate) and FPR (False Positive Rate) on the y and x axes. The TPR is like Recall in the Confusion Matrix, but it also goes by Sensitivity. Another metric is the Specificity, which is TN / (TN + FP) and the FPR is calculated by taking 1 – Specificity or FP / (TN + FP). A well performing model would have an AUC around 1, which means it has a good measure of separability. A poor model would have a score near 0.

Figure 6 – AUC – ROC Distribution

The above figure depicts a model with an AUC of 0.7, so it has a 70% chance of accurately separating the two outcomes. The 0.5 is the decision threshold for classification and the False Negatives and False Positives are depicted above it (David Dalisay).

Predict Sport Outcomes
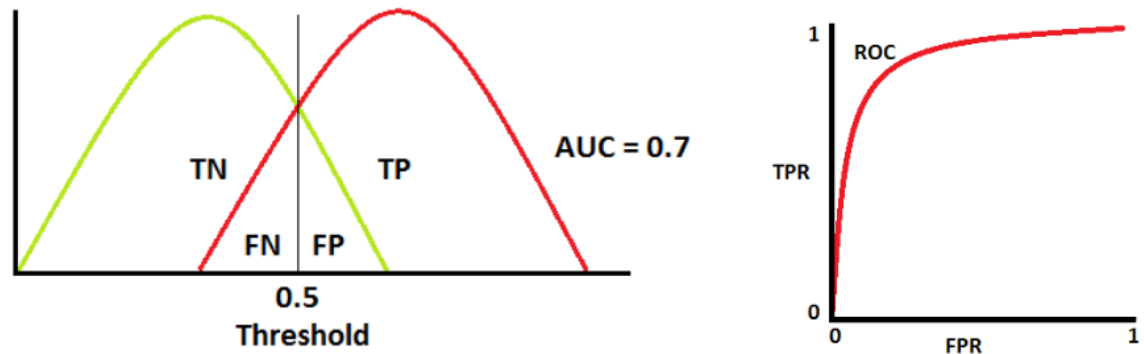
Sport prediction is usually a classification problem to determine wins and losses, but one can also attempt to determine the margin of victory, a continuous target. A lot of unique features play into depicting the outcome such as the historical performance of the teams, results of matches, data on players, and so on. These predictions are pivotal in the betting process as bookmakers, media, fans, bidders, and stakeholders are all deeply involved.

Common models used to predict sports outcomes are Artificial Neural Networks and Logistic Regression. An ANN has interconnected neurons that transform a set of inputs into a desired output. A Logistic Regression is also a common algorithm used to predict sports as it can quantify the wins and losses. When creating a model, it is important to have domain knowledge and a functional understanding of the data. This will go a long way and allow the data scientist to understand what data is important (Bunker, Thabtah).

Current NBA Models

The best performing NBA models achieved an upper bound between 66 - 72% accuracy in predicting the winner of an NBA game. During the regular season of NBA basketball, the upset rate is 32.1%. This means that the non-favorite team will complete an upset about 32%

of the time. This upset rate makes sense, as the best performing models have managed to predict games with about 70% accuracy. In this literature review, NBA prediction models will be examined and compared to one another to understand what goes into creating a successful model.

The most well known model to predict NBA games is the FiveThirtyEight model. They use a metric called Elo ratings in their model. Elo ratings keep track of the final score of each game, where the game was played, and when the game was played. A 1500 Elo rating is the average and starting rating. This score changes as the season progresses adding or subtracting to account for the team's performance. The Elo rating also carries over from season to season as good teams tend to remain good and the same applies for bad teams. The model also uses a metric called RAPTOR, which uses a blend of basic box score stats, player tracking metrics and plus/minus to estimate a player's effect (per 100 possessions) on his team's offensive or defensive efficiency. Both the RAPTOR score and the Elo rating are both updated after each game to reflect the performance of the team and the players. Not only does the model account for team and player metrics, but also other variables such as home court advantage, fatigue (back-to-back games), travel (distance traveled), historical playoff performance, and altitude. The measurements are also adjusted in the playoffs to account for the seeding of teams. The model is complex and uses a variety of feature engineered statistics and has proven to be the gold standard for NBA prediction models. (FiveThirtyEight)

"Which NBA Statistics Actually Translate to Wins" by Chinmay Vayda goes into detail on the specific statistics that are most important to the outcome of a game. The statistics were a team's Offensive Rating, Defensive Rating, Rebound Differential, and 3-Point %. This model got its data from Basketball-Reference.com using the selenium package to interact with the website. This model used all the traditional box score stats as well as Pace and PIE (Player Impact Estimate). The model implemented a variety of machine learning models such as Linear SVC (support vector classifier), KNeighborsClassifier, SVC, Bagging Classifier, Random Forest Classifier, and XGB Classifier. A bagging classifier is an ensemble meta estimator that fits base classifiers each on random subsets of the original dataset and then aggregate their individual predictions to form a final prediction. XGBoost is a decision-based

ensemble machine learning algorithm that uses a gradient boosting framework. The model that showed the best results was the support vector machine SVC classifier. The model performed with great accuracy on the test data but did not perform as well on predicting new games. The model suffered from bias and overfitting on the training set as they had a limited dataset. (Alexander Fayad)

Another model that predicted NBA games was one done by Josh Weiner on Towards Data Science. This was an interesting case as they used a variety of feature engineered columns and player data. They created two separate models: one based on team data and Elo rating and another on player data. This model used the Elo ratings of the teams, the recent team performance of the team (last 10 games), recent player performance, player season performance, and player efficiency ratings. Player season performance is the average player stats over the entire season and player efficiency rating is a rating of a players per minute productivity. The first model created using the team stats and Elo rating used Logistic Regression and RandomForestClassifier as two potential models. The models performed well with 66.95 - 67.15% accuracy. The second model created used individual player statistics and scoring to make their game outcome prediction. It uses stats such as the players average performance over the past 10 games and how many points a player will score in each game. This specific model used Linear Regression model to predict the score of the game based on the points scored by the players. This final game result is then used to predict the outcome of the game. This model did not perform as well and received a 58.66% which was the expected outcome. Aggregated player performance has a high variability and is inconsistent as one of the sole metrics in predicting basketball games. One of the main takeaways from the article is that the time spent optimizing their parameters was not worthwhile, as it was time consuming and computationally costly for only a marginal improvement in the accuracy. (Josh Weiner)

Jake Kandell on GitHub created a high performing model with a unique approach. He got statistics for both the home and the away teams. Then calculated the z score for each statistic based on league means and standard deviations. After those calculations were complete, he took the difference between the two values for each statistic. This allowed the model to

interpret the difference between the two teams based on league averages. Using Logistic Regression, his model accurately predicted games 70% of the time. (Jake Kandell)

The final model that was researched was created by Jaak Uudmae. His approach was to predict games based on previous game score and home/away advantage. The author took advantage of three models: SVM Classifier, Neural Network Classifier, and Linear Regression to predict games. The model achieved accuracies between 62 – 65%. (Jaak Uudmae)

The most common features used in predicting an NBA game were the home team advantage, win percentage, rebounds, assists, turnovers, steals, blocks, plus/minus score, offensive rating, defensive rating, and true shooting percentage. This combination of stats gives a thorough understanding into team performance and can be used to accurately predict games. Most of the models scored from about 58 – 70% range, a score within that range would be acceptable for the model.

METHODOLOGY

The CRISP DM or Cross Industry Process for Data Mining is the industry standard and outlines the proper approach to a machine learning problem. There are seven steps to a machine learning problem.

1. **Problem Definition -** Scope the problem and determine measures for success.
2. **Data Logging and Loading (ETL)** - Identify data, algorithms typically need all data to be numeric, fill missing values, remove duplicates, standardize.
3. **Exploratory Data Analysis -** Summarize and visualize the data to understand it (min, max, median, mean, quartiles, outliers, trends, etc.)
4. **Feature Engineering** - Create new input features from existing ones ex. create interaction features such as height X width which would be area, create dummy variables from categorical variables, create entities from text such as a person's name

5. **Model Selection and Training -** Split the data into train, validation, and test sets. Select the proper model or models to train the dataset.

6. **Model Testing -** Inner loop testing and refining to produce better and better results.

7. **Model Deployment and Monitoring -** Model is deployed and continuously monitored to see how it behaves in the real world, new data is gathered incrementally to improve it.

Define Problem

The goal of the project is to predict daily NBA games. The ultimate measure for success is a model that can accurately predict games greater than 50% of the time.

ETL Data Logging and Loading

The data that is necessary to predict NBA games is freely available on nba.com. The data must be scraped from the website and there are various packages in Python that make this achievable. The nba_api package is a free open-source package that uses the requests package to interact with nba.com. The data that will be necessary to make predictions is an array of extensive NBA team statistics. The statistics were pulled for each team and stored in a Python dictionary. The stats that were scraped are as followed: win percentage, field goal percentage, 3-point field goal percentage, free throw percentage, rebounds, assists, turnovers, steals, blocks, and plus minus (average point differential for games).

The NBA also calculates advanced statistics as well as standard statistics. These statistics are feature engineered and include offensive rating, defensive rating, and true shooting percentage. They offer a better perspective for overall team performance. The offensive rating of a team is the amount of points a team scores per 100 possessions. Defensive rating is the number of points allowed per 100 possessions. Finally, true shooting percentage measures a team's shooting efficiency by evaluating 3-point percentage, field goal percentage, and free throw percentage. The formula for true shooting percentage is PTS / (2*FGA*0.44*FTA) and it is a good indicator of overall team shooting splits.

```
all_stats_dict = {
    'W_PCT': win_percentage,
    'FG_PCT': fg_percentage,
    'FG3_PCT': fg3_percentage,
    'FT_PCT': ft_percentage,
    'REB': rebounds,
    'AST': assists,
    'TOV': turnovers,
    'STL': steals,
    'BLK': blocks,
    'PLUS_MINUS': plus_minus,
    'OFF_RATING': offensive_rating,
    'DEF_RATING': defensive_rating,
    'TS_PCT': true_shooting_percentage
}
```

Figure 7 – Statistics for Model in Dictionary

These 13 NBA stats are the thirteen stats that will be used to predict the outcome of various NBA games. Metrics are tracked for both the home and away teams and are recalculated for every matchup based on team performance. The data was pulled into a dictionary and converted into a DataFrame to use in the model. A few other metrics were tracked to keep track of games and to stay organized such as the date of the game, the current season, the home team's name, the away team's name, the scores of the game, and the game ID. These metrics are important for the user to see the matchup and effectively keep track of the schedule, but these columns will not be used as features in the model. Two other features were added to the dataset as well, Elo rating, and team win percentage the past 10 games. Both of these features will be described thoroughly in the feature engineering section. The final column of the dataset is the target column of which the model is going to predict. This is the outcome of the game and the value is stored in relation to the home team. For example, if the home team wins the game, the data displayed will be a 1 and 0 if they were to lose.

	Home	Away	Game_ID	H_Score	H_W_PCT	H_FG_PCT	H_FG3_PCT	H_FT_PCT	H_REB	H_AST	H_TOV	H_STL	H_BLK	H_PLUS_MINUS
0	Brooklyn Nets	New York Knicks	21800018	107	0.0	0.488	0.185	0.682	38.6	27.7	18.8	8.9	5.0	-3.0
1	LA Clippers	Oklahoma City Thunder	21800025	108	0.0	0.398	0.286	0.833	44.8	20.0	13.3	2.9	8.6	-8.6
2	Orlando Magic	Charlotte Hornets	21800017	88	1.0	0.415	0.280	0.613	49.5	19.4	11.7	6.8	6.8	2.9
3	Utah Jazz	Golden State Warriors	21800024	123	1.0	0.519	0.481	0.737	41.1	19.6	15.9	7.5	3.7	5.6
4	Milwaukee Bucks	Indiana Pacers	21800023	118	1.0	0.494	0.412	0.750	55.3	25.2	20.4	4.9	3.9	1.0

H_OFF_RATING	H_DEF_RATING	H_TS_PCT	A_Score	A_W_PCT	A_FG_PCT	A_FG3_PCT	A_FT_PCT	A_REB	A_AST	A_TOV	A_STL	A_BLK	A_PLUS_MINUS
99.0	103.0	0.545	105	1.0	0.455	0.364	0.774	41.1	18.8	14.3	10.7	5.4	17.0
93.3	102.9	0.497	92	0.0	0.363	0.270	0.649	43.7	20.4	14.6	11.7	5.8	-7.8
101.0	98.1	0.483	120	0.0	0.446	0.421	0.636	39.4	20.2	10.6	7.7	8.7	-1.0
115.0	109.3	0.642	124	1.0	0.442	0.269	0.944	56.3	27.2	20.4	6.8	6.8	7.8
109.7	107.7	0.602	101	1.0	0.566	0.385	0.538	59.4	30.2	20.8	2.1	7.3	29.2

A_OFF_RATING	A_DEF_RATING	A_TS_PCT	Result	Date	Season	Home_W_Pct_10	Away_W_Pct_10	H_Team_Elo_Before	A_Team_Elo_Before
112.5	94.7	0.559	1	2018-10-19	2018-19	0.0	0.0	1500.0	1500.0
97.1	104.9	0.466	1	2018-10-19	2018-19	0.0	0.0	1500.0	1500.0
107.7	109.7	0.551	0	2018-10-19	2018-19	0.0	0.0	1500.0	1500.0
104.9	97.1	0.525	0	2018-10-19	2018-19	0.0	0.0	1500.0	1500.0
115.6	87.4	0.626	1	2018-10-19	2018-19	0.0	0.0	1500.0	1500.0

Figure 8 – Final Dataset

The dataset itself consists of 2,820 rows which is equivalent to 2,820 games collected over three seasons, 2018 – 2021. The exact dates are 10/19/2018 to 3/20/21. The dataset was limited to these three years as it represents a team's recent performance and will aid in the prediction of today's NBA games.

Exploratory Data Analysis

After the data was imported and in a readable format, an exploratory data analysis was conducted to get a better understanding of the data and what kind of results to expect from the model. A thorough EDA will maximize insight into the dataset and uncover the underlying structure of the data. It will also detect outliers, anomalies, and determine the optimal factor settings.

The EDA gave a glimpse into the outcome of an average game and what teams have been performing well the past three years. All null values were One of the goals was to understand which features were going to be used in the model and which were going to be left out. A correlation analysis was conducted on the dataset to see which features interacted with the target variable, the outcome of the game. The three columns with the highest positive correlation analysis were W_PCT, PLUS_MINUS, and ELO rating which is a feature engineered column. The respective ratings were 0.22, 0.23, and 0.25. The Elo rating was the highest indicator of winning a game. These are the key features and will be included in the final model. Some other important metrics were REB, BLK, OFF_RATING, and W_PCT_10. The only features that had a negative correlation was TOV, STL, and DEF_RATING which makes sense because the lower these values are, the better chance of winning a game would be. The exploratory data analysis reassures the importance of each selected feature in the model. All selected columns or features directly correlate to the outcome of a basketball game and are relevant in the prediction analysis. If the feature were to be considered unimportant to the model, it was removed or not used.

Feature Engineering

The NBA feature engineers a few metrics that help determine the effectiveness of a team or a player. These metrics are classified as advanced statistics. These statistics are used in the model and are plus/minus (+/-), offensive rating, defensive rating, and true shooting percentage.

In addition to the advanced statistics, two additional columns were feature engineered to get a better understanding of a team's performance and ability to win basketball games. The first

feature that was feature engineered was Team Win Percentage over the past ten games. This statistic gives the model an understanding as to how the team has been performing recently. It tries to answer and quantify questions such as team chemistry, recent team performance, and any recent changes to the team structure such as trades or injuries. This column was created by selecting the home and away teams earlier ten games, determining their record over this span and returning their win percentage.

The second feature that was feature engineered was the team Elo rating. The Elo rating was calculated after each team performance. The Elo rating is used to gauge team strength and performance and it starts with a median score of 1500. For this model, all teams start with a 1500 and points are either added or subtracted based on point differential, upsets, location, and game outcome. It is a successful metric to represent the quality of the win or loss.

$$R_{i+1} = k * (S_{team} - E_{team} + R_i)$$

Figure 9 – Elo Formula

$$E_{team} = \frac{1}{1 + 10^{\frac{opp\_elo - team\_elo}{400}}}$$

Figure 10 – Elo E team Formula

$$k = 20\frac{(MOV_{winner} + 3)^{0.8}}{7.5 + 0.006(elo\_difference_{winner})}$$

Figure 11 – Elo Constant k Formula

$$(R * 0.75) + (0.25 * 1505)$$

Figure 12 – Elo Season Carryover

The formula above shows how to calculate the Elo rating after each game. R represents the Elo rating, k is a moving constant dependent on the margin of victory and the difference in Elo ratings, E team is the expected win probability of the team, and S team is a state variable (1 for a win and 0 for a loss). After each season, the Elo rating is recalculated using the Season Carryover calculation.

Model Selection & Training

For creating the model, all non-numeric columns were dropped from the dataset. These include Home, Away, Game_ID, H_Score, A_Score, Date, and Season as they will not help in predicting the outcome of the game. The dataset was then split into a training and test set with a 75:25 split. The shape of the training set was (2115, 30) and the test set was (705, 30). This shows the split of the data into training and test. 30 features were used in the training of the model and the features were split for both the home and away teams. There were 15 tracked metrics for each team, which equates to 30 total for each individual matchup.

The training set was then trained using the six different models: Logistic Regression, Random Forest Classifier, K Neighbors Classifier, SVC, Gaussian NB, and XGB Classifier. Five different scoring methods were tracked as well: accuracy, precision_weighted, recall_weighted, f1_weighted, and roc_auc. A crossfold validation was also done fitting five folds. The results for each model were appended to a DataFrame and a confusion matrix was generated for each model.

Model Testing

The six different models were tested and scored with the five scoring methods. The mean and standard deviation is displayed for each scoring method.

| model	test_accuracy		test_f1_weighted		test_precision_weighted		test_roc_auc		test_recall_weighted	
	std	mean	std	mean	std	mean	std	mean	std	mean
GNB	0.040508	0.641153	0.041961	0.642040	0.044508	0.645023	0.032979	0.686600	0.040508	0.641153
KNN	0.019193	0.618980	0.020330	0.616373	0.018472	0.616557	0.019623	0.630484	0.019193	0.618980
LogReg	0.034035	0.639378	0.035705	0.630912	0.032425	0.634730	0.029493	0.680364	0.034035	0.639378
RF	0.028129	0.627279	0.029885	0.622726	0.027518	0.623989	0.033895	0.666171	0.028129	0.627279
SVM	0.023928	0.642155	0.030090	0.605312	0.010787	0.650339	0.030075	0.695869	0.023928	0.642155
XGB	0.025717	0.613276	0.026544	0.609926	0.025384	0.609652	0.023607	0.650003	0.025717	0.613276

Figure 13 – Model Results
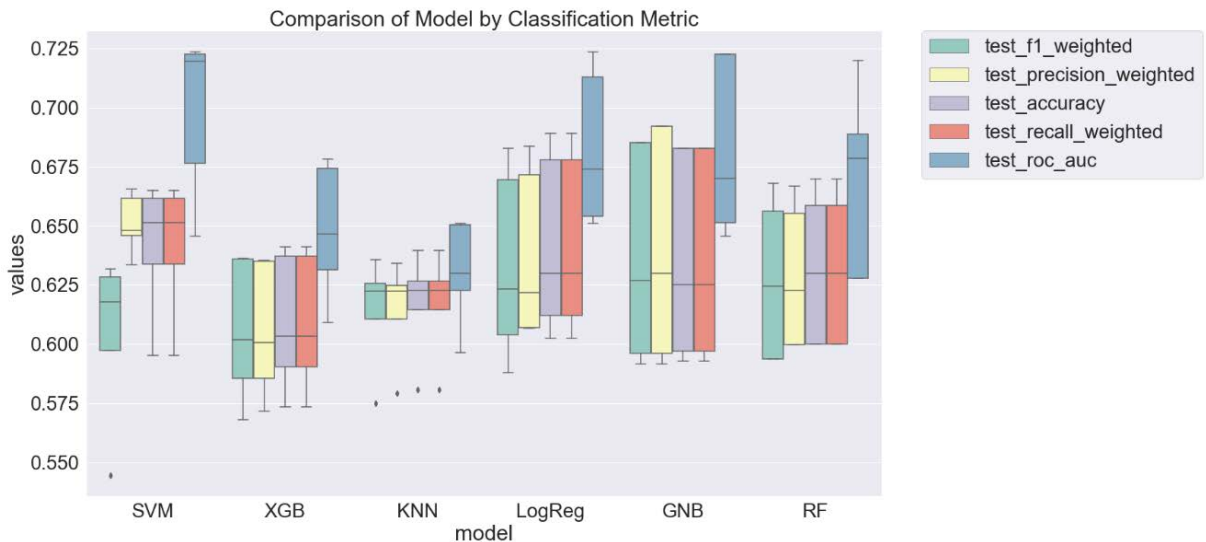
Figure 14 – Comparison of Model by Classification Metric

The figures above display the performance of each model in comparison to one another. The first graph depicts each model's performance with the five-classification metrics. From the

graph, one can see that the top performing models are GNB and Logistic Regression. SVM, XGB, and KNN were ultimately not good fits for the data. The best performing model is the Gaussian NB, it effectively predicts NBA games with an average of 65.1% of the time. The 65.1% is an average based on the five different scoring metrics.
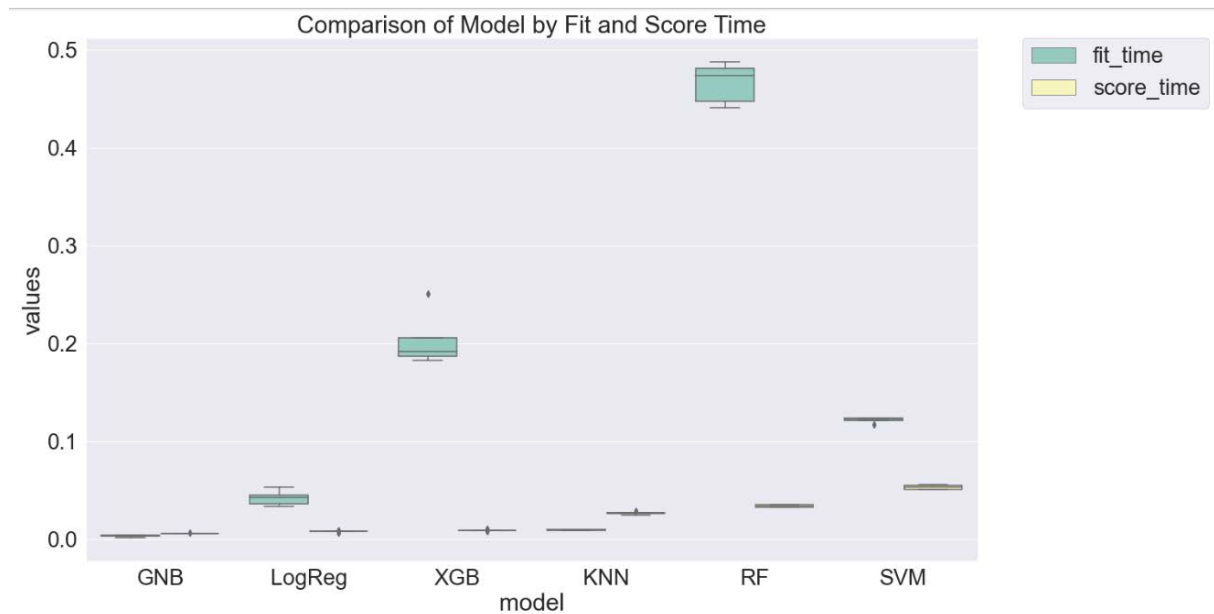

Figure 15 – Comparison of Model by Fit and Score Time

When creating a model, it is also important to consider the computational and processing power of the model. This graph shows the time it took to fit the data to each model. All models scored the data quickly. Random Forest and XGB took a little longer to fit the data than the rest of their competitors. However, the best performing models Logistic Regression and Gaussian NB both achieved a fast time.

Model Deployment

The final model that will be deployed to predict NBA games is the Gaussian Naïve Bayes model as it had the best overall accuracy. It will be used to predict NBA games in the future.

RESULTS

As said, the best model was the Gaussian Naïve Bayes with an average accuracy report of 65.1%. There was an attempt to further improve upon the model by performing an exhaustive grid search to tune the hyperparameters. The only hyperparameter for GNB is 'var_smoothing' which is a variable that is a portion of the largest variance of all features that is added to the variances for calculation stability. 100 different values were inputted for the parameter and a five crossfold validation was done during the grid search. The five folds were fit for 100 candidates, totaling 500 fits. This did find the optimal hyperparameter but did not improve the overall accuracy of the model.

DISCUSSION

Predicting the outcome of sports is a tricky dilemma as there is a level of unpredictability. In basketball, any team can win on any given day and it is extremely difficult to quantify all stats. Metrics such as clutch factor, cold and hot streaks, injuries, referring, trades, and luck all play substantial roles in the outcome of an NBA game. However, these statistics are almost impossible to predict or quantify. A model that can perform and successfully predict the outcome of a game greater than 50% of the time is an achievement. The model successfully predicted 65.1% of NBA games in the dataset. This is a successful model, and it is comparable with the models seen in the Literary Review. The best performing NBA models score in the low 70% range and this is a competitive score in comparison.

CONCLUSION

The final model effectively predicted NBA games with an average accuracy of 65.1%. The model used Gaussian Naïve Bayes to fit the data and make predictions. 30 unique features were used in the creation of the model. Statistics for both the home team and away team were used as features to predict the dependent variable, the outcome.

Some changes that might be made to the model in the future are to calculate the Z score metric of each statistic for both the home and away teams. Indicative of the Literary Review, the best performing model used Z scores to gauge the matchup with great efficacy. Unfortunately, it is a complicated process to implement, as the Z score would have to be

updated continuously to reflect the changes around the league. Furthermore, another potential future implementation would be to fully deploy the model to use in a daily fashion to predict the current day's NBA games. This would require the data to be updated concurrently, but it would be a realistic use case for the model.

APPENDICES

<u>Appendix A – Notes</u>
There are a variety of metrics used to evaluate the accuracy and effectiveness of a model. Common regression models use metrics such as mean absolute percentage error and root mean squared error. MAPE or mean absolute percentage error measures the prediction accuracy.

MAPE – mean absolute percentage error, measure of prediction accuracy of a forecasting method in statistics, the mean or average of the absolute percentage errors of forecasts. It gives us how far the predictions were from the actual output.

RMSE – the standard deviation of the residuals (prediction errors), measure of how far from the regression line data points are, RMSLE (log version) difference between the original values and predicted values and it is easier to compute the gradient compared to MAPE

Statistics -

Hypothesis testing – one tail two tails, null vs alternate hypothesis

T test – used for samples, statistic that measures if two means are reliably different from one another, make inferences about entire population, each t value has a corresponding p value, independent samples, paired sample, one sample

P value – probability that the pattern data in the sample could be produced by random data p =.05 5% chance, .1 10% chance

Confidence intervals - 1 std 68% 2 std 95% 3 std 99.7%

Model complexity – complexity of the function you are trying to learn, ex. Degree of polynomial

Bias variance tradeoff

The gradient is another important concept. It is a vector of partial derivatives and points in the direction of the greatest rate of increase of the function. The gradient is calculated by finding the derivative of a function. The user will use gradient descent for minimization and optimization problems and gradient ascent for maximization. It can also be used to find the roots of a function. The gradient would equal 0 at a local maximum or minimum.
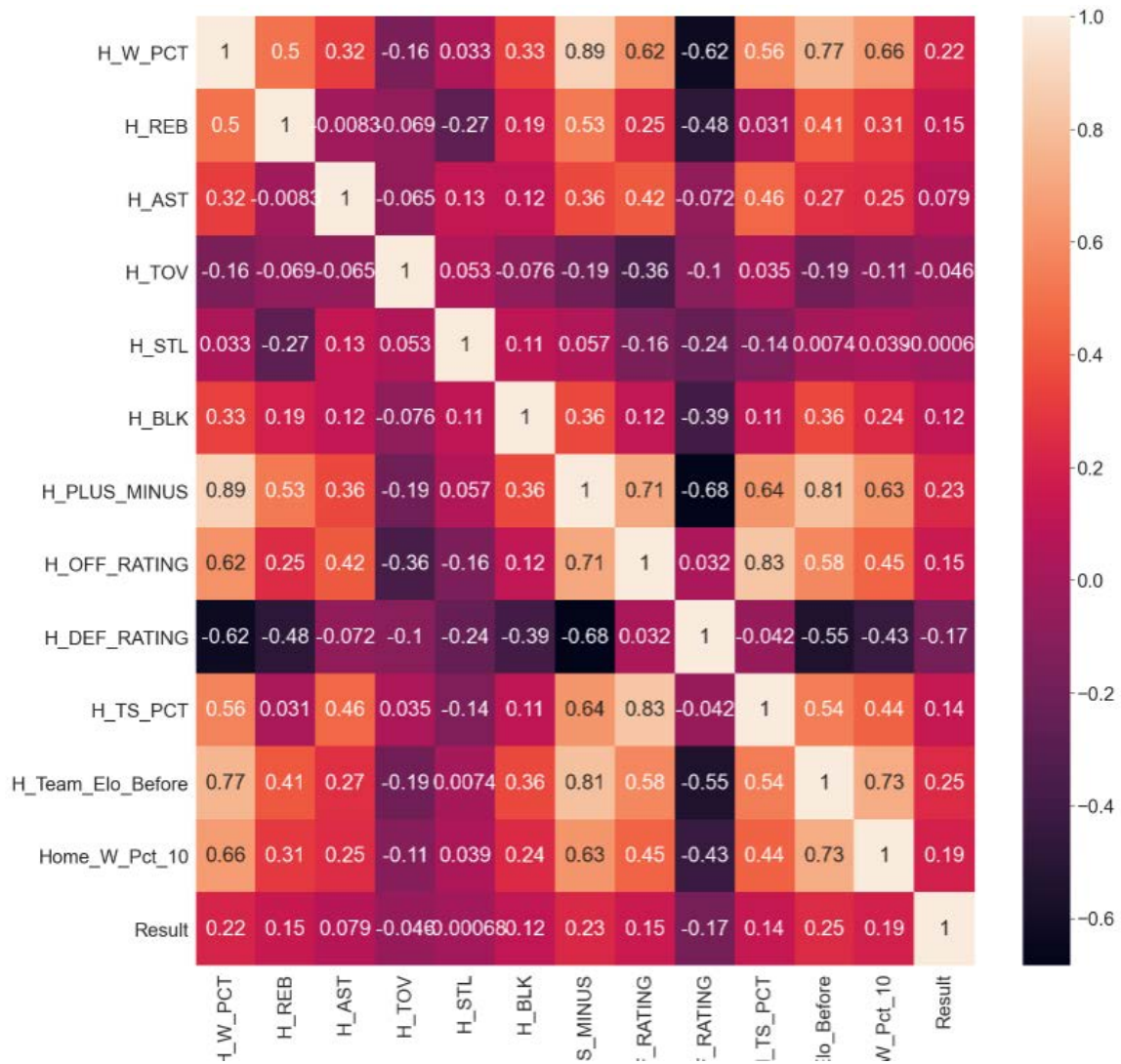
Figure – Correlation Analysis

REFERENCES

Papadopoulos, Aris. "Machine Learning with the NBA's 'Shooting Machine.'" *Medium*, Towards Data Science, 5 Sept. 2018, towardsdatascience.com/nba-data-science-93e0314bb45e.

Silver, Nate. "2019-20 NBA Predictions." *FiveThirtyEight*, FiveThirtyEight, 12 Feb. 2020, projects.fivethirtyeight.com/2020-nba-predictions/.

Neil_Paine. "How Our NBA Predictions Work." FiveThirtyEight, FiveThirtyEight, 18 Dec. 2018, fivethirtyeight.com/methodology/how-our-nba-predictions-work/.

"Machine Learning Systems in Sports." *12th Anniversary*, NBAstuffer, 2020, www.nbastuffer.com/analytics101/machine-learning-systems-in-sports/.

Torres, Renato Amorim. "Prediction of NBA games based on Machine Learning Methods." *University of Wisconsin, Madison* (2013).

Wang, Kuan-Chieh, and Richard Zemel. "Classifying NBA offensive plays using neural networks." *Proceedings of MIT Sloan Sports Analytics Conference*. Vol. 4. 2016.

"The Official Site of the NBA for the Latest NBA Scores, Stats & News." The Official Site of the NBA for the Latest NBA Scores, Stats & News. | NBA.com, National Basketball Association, 2021, www.nba.com/.

Swar. "Swar/nba_api." *GitHub*, GitHub, 22 Feb. 2020, github.com/swar/nba_api.

Franks, Alexander, et al. "Counterpoints: Advanced defensive metrics for nba basketball." *9th Annual MIT Sloan Sports Analytics Conference, Boston, MA*. 2015.

Wheeler, Kevin. "Predicting NBA Player Performance." (2012).

Caliwag, Jasmin A., et al. "Predicting Basketball Results Using Cascading Algorithm." *Proceedings of the 2018 International Conference on Information Science and System*. 2018.

Haghighat, Maral, Hamid Rastegari, and Nasim Nourafza. "A review of data mining techniques for result prediction in sports." Advances in Computer Science: an International Journal 2.5 (2013): 7-12.

Schumaker, Robert P., Osama K. Solieman, and Hsinchun Chen. "Predictive modeling for sports and gaming." Sports Data Mining. Springer, Boston, MA, 2010. 55-63.

Jaagu. "Jaagu/CS229FinalProject." GitHub, GitHub, 2017, github.com/jaagu/CS229FinalProject.

Kandell, Jake. "JakeKandell/NBA-Predict." GitHub, GitHub, 2019,
github.com/JakeKandell/NBA-Predict.

"Machine Learning Tutorial: Machine Learning with Python - Javatpoint."
Www.javatpoint.com, JavaTPoint, 2018.

Bunker, Rory P., and Fadi Thabtah. "A machine learning framework for sport result
prediction." *Applied computing and informatics* 15.1 (2019): 27-33.

Engineer, Kevin Beaulieu Software, and David Dalisay. *Machine Learning Mastery*, MLM,
19 Sept. 2019, machinelearningmastery.com/.

"Machine Learning with Python Tutorial." *Tutorialspoint*, Tutorials Point, 2021,
www.tutorialspoint.com/machine_learning_with_python/index.htm.

"Learn." *Scikit*, Scikit - Learn, 2021, scikit-learn.org/stable/index.html.

Srivastava, Tavish SrivastavaTavish. "K Nearest Neighbor: KNN Algorithm: KNN in Python
& R." *Analytics Vidhya*, Analytics Vidhya, 18 Oct. 2020,
www.analyticsvidhya.com/blog/2018/03/introduction-k-neighbours-algorithm-
clustering/.