OR 568 Final Project: Predicting NBA Wins

Sravani Varma Gottumukkala
Nebojsa Hrnjez
Tanishq Ravi
Chris Smith
Ben Zevin

Table of Contents

Project Description ***3***

Data Set ***3***

Data Source **3**

Data Description **3**

Data Preparation & Final Dataset **4**

Final Dataset: **5**

Data Exploration ***6***

Exploratory Analysis **6**

Data Analysis ***6***

Pre-Model Code **6**

Standardized Model Sequence **7**

Models, Tuning & Results **7**

Base Case: **9**

Alternate Cases: **10**

Conclusions ***11***

Challenges and Further Analysis **12**

Works Cited ***13***

Tables ***14***

Figures ***15***

Code Snippets ***22***

Project Description

The number one objective of any sports team is to win games and this fact holds true in the NBA. In the past teams would often rely on the "eye-test" to determine whether a factor like adding a player, changing an offensive scheme, or focusing on a different aspect of the game would lead to more wins. This process is based solely on feeling and personal biases and is often unreliable.

In recent years data analysis and science has become an industry of its own within sports. Teams will use machine learning models and predictive analytics to determine the best possible outcome with varying degrees of success. These methods and analyses are often closely guarded secrets, allowing teams to have an advantage over one another. Other stake holders utilizing sports analytics include sports gambling companies to determine betting odds like DraftKings and FanDuel as well as sports media such as ESPN to provide insight to their programming.

Utilizing the techniques taught in OR 568 on NBA game statistics, various models will be trained to determine which game statistics, both basic and advanced, lead to a win and then these models will be used to predict which team would win in the future based on game stats. There are two research questions that this analysis will explore. The primary question is related to the performance of the classifier: Can a model be built to predict which team will win? Additionally, through further analysis of classifier performance, the key input variables will be examined and a determination of which statistics most affect a team's chances to win.

Data Set

Data Source

This data set was originally sourced from Kaggle (Lauga, 2022) on February 22nd, 2022. This dataset was collected by the author, Nathan Lauga, from the NBA official statistics website. Upon further investigation it appears that this dataset was used by the author for a similar project, predicting the winner of an NBA game, but at the time of writing this report no results have been posted.

Data Description

This dataset is an amalgamation of five different csv files all related to NBA games data. Below is a detailed description of each CSV file.

games.csv:

The games.csv dataset contains several team-based statistics for each game from the 2004 season up to last update (this data was pulled February 22nd, 2022). For each game there are numerical statistical totals for the home and away team, for the following stats: Points, Field Goal Percentage, Free Throw Percentage, 3-PT Percentage, Assists and Rebounds. Additionally, there is a binary variable called "HOME_TEAM_WINS" which indicates whether the home team won (1) or lost (0). Finally, there is a unique game id for each game as well as a home and away team id for each team which is unique to the season, i.e., the 2020 Washington Wizards will have a different id than the 2021 Wizards. A short sample of this dataset can be seen in Figure 1.

games_details.csv:

The games_details.csv dataset would be considered a more in-depth statistical view of each game, consisting of the individual player statistics for each team. The same previously mentioned game and team ids are used in addition to a player id to identify data points. The numerical statistics represented in this dataset for players are Minutes Played, Field Goals/Free Throws/3-PTs Made, Field Goals/Free Throws/3-PTs Attempted, Field Goal/Free Throw/3-PT Percentage, Offensive/Defensive Rebounds, Assists, Steals, Blocks, Turn-Overs, Personal Fouls, Points and Plus/Minus. Additionally, this dataset contains information like the player's name, nickname, position. A short sample of this dataset can be seen in Figure 2.

players.csv:

The players.csv dataset is a purely informational dataset meant to act as a link between other datasets. This dataset contains the player's name and their team and player id for a particular season. A short sample of this dataset can be seen in Figure 3.

ranking.csv:

The ranking.csv is a high-level dataset that looks at a team's wins at various dates in a season. This data set details games played, wins and losses as well as a home and away record grouped by team id and season id. Additionally, there is a date to identify the record at a certain period. A short sample of this dataset can be seen in Figure 4.

teams.csv:

The teams.csv contains what is simply explained as trivia information about each of the teams. Some information includes the arena the team plays in, the owner and the year they were formed. This data had no useful information and was never considered for this project.

Data Preparation & Final Dataset

Problem Statement:

To completely explain how this data was prepared and how the final dataset was reached, first the development of the problem statement must be discussed. Before this group settled on the current goal, originally two possibilities for the project were considered:

1) A regression problem: Predicting the total number of wins a team has at the end of the season
2) A classification problem: Predicting a win or a loss on a single game

Ultimately the classification problem was chosen because of the potential problems perceived in predicting the total number of wins at the end of a season such as: the point of the time in a season affecting wins (hot-streaks and lulls), whether to using a running average statistics or season totals and other time-series related issues. Within the classification problem the nature of the dataset was also changed after the results at the midterm presentation proved to be unsatisfactory.

Originally, the plan was to predict a win or loss based on a single team's game statistics. This was motivated by a potential real world use case where a team could look at their season averages going into a game and predict if they win and what statistics would be important to win. The main problem with this method of logic was that basketball is a game played by two teams and the probability of winning changes based on who you are playing. Without comparing one teams performance to the opposing teams any potential model loses out on a significant portion of important information. Furthermore, this was validated by the models developed for the midterm presentation, with neither logistic regression nor classification trees performing well for numerous reasons.

Data Preparation Process:

The dataset was created using the attached R code "Classification Problem_2 Teams Data Set.R" and ultimately ends up with 70 variables with 25009 observations. Outside of some identifier columns, the response variable, "HOME_TEAM_WINS", the final dataset consists of duplicates of predictors, one for the home team and another for the away team. The predictors can be categorized three ways:

Raw NBA game statistics:

These predictors are directly from the games.csv and games_details.csv datasets sourced from Kaggle and are considered original predictors in the sense that they are not derived and are "facts" of the game. The predictors include Points, Rebounds, Assists, Field Goals/Free Throws/3-PT Percentage, Field Goals/Free Throws/3-PTs Made, Field Goals/ Free Throws/3-PTs Attempted, Offensive/Defensive Rebounds, Steals, Blocks, Turn-overs, and Personal Fouls.

These predictors cover the entirety of statistics available in an NBA game without performing calculations or otherwise transforming them. Raw statistics are vital to the model because all other predictors are derived from these values. Expanded upon in later sections, certain predictors and their derivatives are too highly correlated to be used in modeling but are included to derive other statistics that do not suffer from this problem.

A large part of the pre-processing or preparation for this project was done for these predictors. As detailed previously the games.csv and games_details.csv each contain the desired dataset. To combine these data sets mainly the "dplry" package (Hadley Wickham, 2022) was used in conjunction with base R.

Prior to any transformation or combination, the datasets were inspected for missing values. Looking at Figure 5, several predictors have many missing values. As previously mentioned games_details.csv contains individual player statistics for each game, and during an NBA game some players do not play for a variety of reasons (injury, coaches' decision). To deal with this all missing or NA values were changed to 0, logically if you do not record a statistic, it's the same as recording a 0 for that statistic. Code 1 details the use of the "mutate_if" function from "dplyr" to replace all NA values with 0.

Once the datasets were prepared, the two datasets were combined in the most logical way. Since games.csv already had some game total statistics already it was treated as a base dataset with additional statistics added to it. First game total statistics were calculated by grouping the "GAME_ID" and "TEAM_ID" and then summed into a single value (Code 2) using the "group_by" and "summarize" functions from "dplry". This new dataset, add_game_stat, was then added to the games dataset using the "merge" function from base R, this created the new dataset all_games_raw (Code 3).

Statistical Spreads:

Next the spread between home and away predictors was calculated and added to the dataset. For all spread stats the formula is:

$$Home\ Stat - Away\ Stat = Stat\ Spread$$

This was done because the response variable is a binary value based on whether the home team wins or not. Logically, the value of a statistic is only important in comparison to the opposing team's same statistics. While a model might be able to infer this from the home and away values of a stat, providing this value ensures the comparison is included. Additionally, this may be used to reduce the overall number of predictors down from 70 as it is a derivative of home and away raw statistics. The calculations for this can be seen in Code 4.

Advanced NBA statistics:

Finally advanced statistics were calculated for each game for both the home and away team. Advanced statistics are derived statistics using formulas (Ugur, Analytics101 - Basketball Team Evaluation Metrics, 2007) that help to quantify nuances of the game not expressed in regular stats. These advance statistics are formulated by statisticians such as ESPN's John Hollinger (Ugur, John Hollinger, 2007). The advanced statistics are explained as follows and can also be seen in Code 5:

Possession: A calculation of the number of times the team has the ball during a game.

Offensive Efficiency: Equal to an opposing teams "Defensive Efficiency", offensive efficiency is what the name states, how efficient the offense is. Given as the number of points a team scores per 100 possessions.

Effective Field Goal Percentage (eFG): A metric that combines the impact of 3-Pt and 2-Pt field goals into a single field goal percentage.

True Shooting Percentage: Similar to eFG but considers the impact of free throws as well.

Assist Ratio: the percentage of a team's possessions that ends in an assist.

Turnover Ratio: the percentage of a team's possessions that ends in a turnover.

Final Dataset:

The final dataset can be seen in full in the "all_games.csv" file. This data set contains 3 unique identifiers for home team, away team, and game, 1 response variable; "HOME_TEAM_WINS", and 66 predictor variables. There are 25009 unique observations across this dataset with no missing values.

While this dataset may seem like it has too many predictors there are many techniques for feature selection and reduction and an important aspect of data science is not removing information until necessary. The R file

“Classification Problem_2 Teams Data Set.R” also contains different datasets that contain only some predictors, like a one with only the spreads. Ultimately there may be high correlation between predictors, especially those that are derived from others, but at this step it is not necessary to do any additional data cleaning or “wrangling”.

Data Exploration

Exploratory Analysis

An exploratory analysis was done on the “all_games.csv” dataset to determine patterns in the predictors and to highlight any unique circumstances that may need pre-processed prior to model implementation. This section can be seen in the R file “all_games_data_exploration.R”

Distribution:

Looking at Figures 6 to 9 we see that the entire dataset appears to be normally distributed (game and team ids are not but they are not predictors just identifiers). This is to be expected as you are averaging game stats over the course of 22 years and while there are certain changes that have occurred to the game (i.e., the importance of the 3-point shot, less emphasis on defense) overall it is much the same. Certain statistics appear skewed, but the mean is simply shifted, and the graph shows the entire range of data points that appear. This normal distribution in the raw stats is carried through to all other stats.

Scale:

The scale of our predictors is naturally different due to the nature of the game, for example there will be almost 4 times as many points in a game in comparison to assists. All models will be centered and scaled to ensure consistency and to make sure no one variable outweighs the others.

Near Zero Variance Predictors:

Looking at Code 6, the “nearZeroVar” function from the “caret” package (Kuhn, 2022) was used to determine if any of the predictors had near zero variance. There are no predictors with near zero variance as is to be expected from real sports data.

Data Analysis

Pre-Model Code

The full sequence of code used to prepare the data sets for each model can be seen in Code 7. All models are run on these datasets to ensure comparability and consistency in results.

Train/Test Dataset:

For this project each model was trained and tested on the “all_games.csv” dataset that was described in the data preparation process section of this report. A sample variable was created from the original data set used to later split the dataset into training/test split. The data was split to a ratio of 80% training and 20% test. Before creating the datasets, the problem was examined from a top-down perspective. It was clear that including the predictors HOME_TEAM_PTS, AWAY_TEAM_PTS, PTS_SPREAD would lead to problems of overfitting and in general are too related to the response variable HOME_TEAM_WINS. This problem would continue to plague this project as predictors related to points in anyway were often highest in variable importance.

An initial run of this dataset showed that the predictors FG_PCT_SPREAD and FGM_SPREAD were consistently the important variables seen in Figure 10. The decision was made to remove them as one of the goals was to determine which statistics are important to win basketball games. It is relatively obvious that if you score more efficiently and more baskets than your opponent you will win, and while it is not a certain fact it is highly unlikely you will lose. This problem would persist, but the decision was made not to remove any more predictors as to maintain the integrity of our dataset and the information we are trying to extract. Additionally, identifier columns were also removed like GAME_ID, HOME_TEAM_ID and TEAM_ID_away.

Once the predictors were removed, the dataset was split into a train and test data set using the sample variable. Additionally, “train_x” and “test_x” which is simply the dataset with the response variable removed were

created for ease of input into the models. Finally, the response variable was converted into a factor to allow for comparison between observed and predicted response in a confusion matrix.

Final Test Dataset:

To truly test whether our model could predict whether a game is won or lost a final test dataset was created based on the 2022 NBA playoffs. Initially this dataset was created to simulate a real-world use-case where professional sports teams would predict whether they would win or lose an upcoming game. To do this the season averages of two teams playing were taken and varied by 95%-115% for the home team's stats and 90%-105% for the away team's stats. The ranges were selected to impart the effect of home court advantage and this dataset can be seen in "final2_test_data.csv"

Detailed later in the challenges and further analysis section of this report, the above process is flawed and produced poor results. In a real-world setting, there would be teams of data scientists and statisticians applying several techniques and comparing countless variables to determine the input statistics to predict future games. To determine the performance of our models we took the actual statistics from playoff games that occurred and used those inputs to test our models and this dataset can be seen in "final3_test_data.csv".
A similar process to the train/test dataset was performed in terms of removing required predictors and converting the response variable into a factor.

Standardized Model Sequence

For all models in this project a standardized approach was taken to ensure consistency and comparability amongst results as seen in Code 8. The pre-model code seen in Code 7 creates a trainControl variable "ctrl, with the method selected being LGOCV or leave group out cross validation.

All models were created using the "train" function from the R package "caret" (Kuhn, 2022) with the method argument being changed as needed and any necessary tuning parameters called out as required. Once the model was trained, the accuracy and kappa were extracted using the "confusionMatrix" function from the "caret" package and the "kappa" function from base R respectively. Finally, the most important variables for that model were found using the "varImp" function.

Once the model was fitted it was tested using the test set from our train/test split, the predict function was used for all models. Again, the accuracy and kappa were extracted as well as other important performance statistics like sensitivity and specificity. Finally, the models were tested on the final test set and the again the "confusionMatrix" function was used. There is only one piece of code for the final test portion and the actual vs generated final test data set is change in the pre-model code. Visuals were created at the end as required for each model.

Models, Tuning & Results

Logistic Regression:

Logistic Regression predicts the probability of one response variable with two outcomes or classes by having the log odds for the event be a linear combination of one or more predictor variables. Logistic Regression was completed using the "train" function in the "caret" R package with the "method" parameter being "glm." The data would also be centered and scaled in the function using the "preProc" parameter. Logistic Regression doesn't have any tunning parameters. The results from the model received an ROC score of 0.9998 and an accuracy of 0.9982 with the test set from the original file. Using the generated 2022 test set, it performed poorly with an accuracy of 0.44 and a kappa of 0. The actual 2022 playoff set didn't perform much better, with an accuracy of 0.57 and a kappa of 0.18.

Sparse LDA:

Sparse Linear Discriminant Analysis is a penalized version of Linear Discriminant Analysis which is another model that attempts to find a linear combination of predictors that labels a response variable of two or more classes. The package and function used are the same as Logistic Regression, except the "method" parameter is now "sparseLDA." This method does have two tuning parameters, "lambda" and "NumVars." "Lambda" ranged

from 0.001 to 1, and "NumVars" ranged from 1 to 1000. After running the model, the optimal tuning parameters were NumVars = 50 and lambda = 1. Using the optimal parameters, the ROC score of the model is 0.997, and the accuracy using the original test set is 0.972. When using both of the 2022 test datasets, the model generated the same results with an accuracy of 0.44 and a kappa of 0.

Partial Least Squares:

Partial Least Squares is a statistical model that projects response and predictor variables to a new space. The "method" parameter for the train function this time is "pls," and the tuning grid for Partial Least Squares only has one tuning parameter, "ncomp," which stands for the number of components. The model was tested over a range of 1 to 60 components to discover that the optimal number of components is 24. The model with 24 components has an ROC score of 0.997 and an accuracy of 0.9714 using the original test set. When using both 2022 test datasets, the model generated the same results with an accuracy of 0.44 and a kappa of 0.

Penalized GLM:

Penalized Generalized Linear Model is a model that uses a lasso-like penalty on a binomial likelihood function. The method "glmnet" in the train functions uses ridge and lasso penalties simultaneously. This tuning grid has two parameters, "alpha" and "lambda." "alpha" ranges from 0 to 1, and "lambda" ranges from 0.0001 to 2. The optimal parameters were alpha equal to one and lambda equal to 0.0005. This model has a ROC value of 0.997 and an accuracy of 0.9944 with the original test set. This model performed better than the previous linear models with an accuracy of 0.49 and a kappa of .07 using the generated 2022 test set. It had an accuracy of .68 and a kappa of .40 using the actual 2022 test set.

Nearest Shrunken Centroids:

Nearest Shrunken Centroid model finds the centroid for each class by taking the average value of each predictor in the training set. Then, the overall centroid is computed using all the classes' data. The method parameter must be "pam" in the train function to use this technique, and this model only has one parameter, "threshold." The optimal threshold was discovered to be zero, and using the optimal parameters, the ROC score of the model was 0.9744, and the model's accuracy using the test set was 0.9118. This model performed the best out of all of the linear classification models we used during this project. Even though it had the lowest accuracy with the original dataset, it had the highest accuracy and kappa for the generated and actual 2022 dataset. It has an accuracy of 0.65 and a kappa of 0.27 with the generated set, and it had an accuracy of 0.79 and a kappa of .58 with the actual set.

Neural Network:

Neural networks use interconnected nodes to mimic the functions of a brain. These nodes can act as miniature regression equations. As such they are prone to overfitting. The base construct of a neural network is an input layer, output layer, and up to -n numbers of hidden layers. A loss function like cross-entropy is used to optimize the parameters. With enough nodes and data neural networks will learn relationships between predictors. This also makes them vulnerable to collinearity. The trained model resulted in three hidden layers and a weight decay of 1. This resulted in an area under the ROC curve of 0.999 and poor performance on the 2022 playoff data. Due to the nature of our predictors, the model is likely overfit.

Support Vector Machine:

Support Vector machine uses a linear hyperplane to separate the two classes. In the case of highly clustered data like this dataset, the model transforms the data to a higher order to create a linear hyperplane. Points are then determined by how far they are from this hyperplane to determine class. The optimal tuning parameter on the

training data came to sigma = 0.01 and C equal to 64. The model had an accuracy of 0.99 on the test data. The model performed poorly on the 2022 playoff data, predicting only one class for all values.

K-Nearest Neighbor:

K nearest neighbors is a distance-based model. On the training data the optimal parameter came to K equal to 94. Of the approximately 20,000 neighbors the model relies on the closest 87 neighbors to determine the classification. The model fit with an accuracy of 0.9334 on the test split of the data with an area under the ROC curve of 0.985.

Naïve Bayes:

Unlike other models that rely on distance, Naïve Bayes relies on the prior probability of the predictors. Given the likelihood of a predictor with this value, what is the probability that this is a win or a loss? Naïve Bayes does assume independence between predictors, which in this case is not a valid assumption. Despite this the model performed well on the train/test split with an area under the ROC curve of 0.96 and an accuracy on the test set of 0.88. The model used a Laplacian correction of 1 and an adjust of 2. Naïve Bayes performed well on the 2022 playoff data with an accuracy of 0.89 and poorly on the noisy data set. This is likely due to the range of some of the noisy values.

MARS:

The Multivariate Adaptive Regression –Splines use piecewise functions to prune a tree. Like in regression distance from the splines determines which branches to trim. The model performed well on the train and test data with accuracy and area under the ROC curve of 0.99. It also performed very well on the 2022 playoff data with an accuracy of 0.91.

Stochastic Gradient Boosting:

It is a boosting approach, wherein each iteration a randomly selected sample of training data and the best subset is used to train the base learning model. Then it eventually tries to find other well-trained data samples and integrates them to finalize the training model. When trained with a 75% split for training data with 25 repetitions the best tuning parameters were the number of trees of 150, and interaction depth of 3. The model stochastic gradient boosting is performed well with an accuracy of 97% and the area under the curve for ROC of 0.97.

Extreme Gradient Boosting:

Similar to other boosting models, Extreme gradient boosting commonly known as XGBoost is also a supervised machine learning algorithm. It uses advanced regularization methods that improve the generalization capabilities of the model, which will greatly reduce the model overfitting tendency. Regularization takes care of the decision of whether the leaf node in a tree is participating in further growth. When trained with a 75% split for training data with 25 repetitions the best tuning parameters were the number of trees of 150, and interaction depth of 3. The model's performance is great, as it is performing slightly better with an accuracy of 98.27% and the area under the curve for ROC of 0.98.

Base Case:

Accuracy and Kappa

The accuracy and kappa results for this project can be seen in Table 1. As discussed in the individual model sections all the models performed well on the training/test split. The models begin to differentiate themselves when looking at the final test using actual playoff stats.

We notice that the best performing models were: Logistic Regression, Penalized GLM, Nearest Shrunken Centroids, K-Nearest Neighbor, Naïve Bayes, MARS and both types of boosting models. Of these NSC, Naïve

Bayes and MARS performed especially well with a Kappa higher than 50%. The boosting models were not included in this list as we believe they might still be suffering from over fitting.

The models suffering from overfitting is most likely due to predictors related to points in some way. As previously discussed in this report, looking at this problem from a macro perspective it made sense to remove predictors that were directly related to points and after an initial run of our dataset revealed a couple other predictors worth removing, the group decided this would be sufficient for the base case.

The results do hold some value though as we were able to determine through the actual stats final test which models performed the best.

ROC, Sensitivity and Specificity

The ROC, Sensitivity and Specificity was also recorded for our base case and the results seem the agree with the initial assumption of overfitting. Table 2 shows that every statistic was over 85% and majority were over 90%. Results are difficult to quantify as

Important Variables

As mentioned previously the important variables for each model were extracted using the "varImp" function. The top 5 variables were recorded for each model and then ranked; Table 3 shows the results of this process.

As our group expected predictors related to points appear frequently on this list such as: away and home offensive efficiency, away and home true shooting, away and home effective field goal percentage and many others. This verified our previously stated hypothesis that the models were likely cheating their way to a correct answer through extrapolation of the points statistic.

This process did yield some insights, defensive rebounding spread was the third most important variable even with all the points/scoring related predictors included. This is especially interesting considering that this predictor is higher than total rebound and offensive rebound spreads. One reason may be the increased frequency a team will recover defensive rebounds at versus offensive rebounds. Assist spread was also an important statistic, this is to be expected as one of the fundamentals of basketball is ball movement and passing.

Alternate Cases:

To determine the overall effect of predictors related to points another dataset was run with HOME_OFF_EFF, AWAY_OFF_EFF, HOME_TS, AWAY_TS, HOME_POSS, AWAY_POSS, HOME_eFG, and AWAY_eFG predictors removed, this will be called alternate case 1. Alternate case 2 with all predictors related to shooting and points removed was also tested to fully validate if predictors related to points/scoring were adversely affecting our models. We re-ran only the best performing models from the base case to speed up computational time for both the alternate cases.

Accuracy and Kappa

Looking at Tables 4 and 5, the difference in values for accuracy and kappa between alternate case 1 and 2 seem to support the previously mentioned hypothesis that predictors related to scoring were causing model overfitting. Looking specifically at Table 5, the accuracy and kappa values are not nearly as high as the base and even alternate case 1, this is the first thing to look for when determining if models are overfitting. This is again validated when looking at the actual stat final test accuracy and kappa where the alternate case 1 models appear to perform similarly to the base case whereas the results for alternate case 2 are much more consistent with each other.

From a performance perspective Penalized GLM, NSC and Naïve Bayes and both boosting models perform best for alternate case 1. For alternate case 2, the results were relatively standardized with all models having an accuracy between 80-85%, with Stochastic Gradient Boosting having the highest overall accuracy and kappa.

ROC, Sensitivity and Specificity

For the base case, ROC, Sensitivity and Specificity were all so high due to overfitting that these metrics were not very valuable for analysis. Looking at Table 6 and 7, again the results seem to agree with the overfitting hypothesis. Table 6 shows the results for alternate case 1 and like the base case the values are all high with all of them being higher than 85% and a majority being higher than 90%.

Looking at the alternate case 2 results in Table 7 there are some patterns to note. It can be noted that for all models the specificity is higher than the sensitivity meaning that these models are better at predicting cases the home team loses than if they win. This is particularly the case for the NSC and K-Nearest Neighbor models where the difference is 15-20%.

Correctly predicting a true positive or true negative does not have any severe consequences in this scenario as these are just basketball games.

Important Variables

Table 8 and 9 show to results of the "varImp" function for both cases. Again, for alternate case 1, predictors related to scoring/points dominate this table. With the consistency and similarity of the results to the base case it can be determined that predictors related to scoring or points will cause overfitting.

It is interesting to note that for both Table 8 and 9, defensive rebounding spread is at the top of both lists, it was ranked 3rd for the base case. This is especially surprising considering that alternate case 1 still includes some predictors related to shooting/scoring. Additionally, assist spread which appeared in the top 5 for the base case is also in the top 5 for both alternate cases. Other highly ranked predictors include turnover and personal foul spread.

There are number of patterns to note from the variable importance results. It is clear to see that the spread is more important than the individual statistic and this makes sense when considering that basketball is played by two teams and your goal is to outperform your opponent.

Additionally, defensive rebounding is one of the most important components to the game outside of scoring, higher than offensive or total rebounding and many other aspects like assists or steals. There are two potential reasons for this; defensive rebounds are much more abundant than other statistics and defensive rebounds are one of the most important factors to a team's defense. Looking at figure 11 we can note that for both home and away the mean value for defensive rebounds is the highest among the non-scoring predictors.

Finally, defensive predictors in general do not appear very highly in any list. Predictors like steals and blocks do not appear on the base case or alternate case 1 important variable tables and appear at the bottom of the alternate case 2 table. This could be due to an overall trend in the NBA the last few years that over emphasizes offense as well as the lack of numbers for those statistics (steals has a mean of about 7.6 and blocks a mean of about 5)

Conclusions

From this project we can conclude the following. Based on the results of this project, the best models for predicting NBA wins Penalized GLM, NSC and Naïve Bayes as well boosting based models. It is clear based on the predictors and their effect on models that shooting efficiency and scoring points remain the most important factors in winning a basketball game. Outside of scoring, defensive rebounding and assists seem to be the most important factors a team focuses on to win a game. Teams should specifically focus on increasing their teams value over the other teams value for these stats as the spread for these stats was deemed more important that the individual stat.

Challenges and Further Analysis

Future Prediction:

One of the biggest issues was generating a final test data set to determine whether our model truly had predictive capabilities. For this project our model was trained on previous game statistics and while it can accurately determine a winner it can only do this when realistic statistics are inputted.

In a real-world use case, a basketball team wanting to use predictive analytics would have an entire team of data scientists and analysts performing these calculations and building models. When looking at our project from this perspective the key difference would be the statistics used for the "Final Test" of the models. Where this group used seasons averages and randomly varied them within a set range, the data science team for a professional team would likely have a much more robust and analytically created input dataset. Their dataset would likely generate statistics based on several factors such as statistics relative to opponent, potential player matchups within the game, win/loss streaks, and many more. These statistics would also be randomly generated and tested countless times. All these techniques are outside of the capabilities of this group and the scope of this project.

While this does limit the evaluation of the performance of our models based on the training and test results it is believed that the model would correctly predict the winner. As a final test the actual statistics for the games used in "final3_test_data.csv" were inputted as the models performed in accordance with their training and test performances.

Computational Cost:

Another issue that affected this group and limited in the testing of possible "what-if" scenarios is the time it took to run all the models. Certain models are fitted and tested very quickly but models like MARS, SVM, and Naïve Bayes can take upwards of 45 minutes each to run. Only a "base case" as mentioned in the methods used section of this report was extensively test and evaluated.

Alternative cases such as one with only one teams statistics inputted (as originally proposed) and one where all predictors related to points were removed were examined but not to the level required for a proper write up and through inclusion in this report. In a real-world use case, data science teams would have access to more powerful computers as well as a larger number of them to allow for running in parallel.

Removing Predictors:

As mentioned previously this project struggled with which predictors to removed. Certain predictors directly related to points like HOME_PTS, AWAY_PTS and PTS_SPREAD was easy to remove as they quickly indicated a win or a loss. Where the difficulty arises is the predictors that are points adjacent like shooting percentages, shooting attempts and the advanced statistics related to those raw statistics. There is a large amount of information in these predictors outside of the obvious score more points win more games. One example of potential analysis is the emphasis NBA teams have placed on 3-point shooting in recent years. As recently as 2010 the consensus in the NBA was that to win a championship you needed a dominant center, but this changed around 2015 with more teams focusing solely on shooting and making more 3 pointers. This change in the game wasn't discovered by chance but through an examination of shooting analytics and factors affecting game wins.

Works Cited

Hadley Wickham, R. F. (2022, February). *dplyr: A grammar of data manipulation*. Retrieved from CRAN: https://cran.r-project.org/web/packages/dplyr/index.html

Kuhn, M. (2022, April 19). *caret: Classification and Regression Training v6.0-92*. Retrieved from CRAN: https://cran.r-project.org/web/packages/caret/index.html

Lauga, N. (2022, March). *NBA games data*. Retrieved from Kaggle: https://www.kaggle.com/datasets/nathanlauga/nba-games?select=games.csv

Ugur, S. (2007). *Analytics101 - Basketball Team Evaluation Metrics*. Retrieved from NBAstuffer: https://www.nbastuffer.com/analytics-101/

Ugur, S. (2007). *John Hollinger*. Retrieved from NBAstuffer: https://www.nbastuffer.com/analytics101/john-hollinger/

Tables

Method	Training Accuracy	Training Kappa	Test Accuracy	Test Kappa	Generated' Test Accuracy	Generated' Test Kappa	Actual' Test Accuracy	Actual' Test Kappa
Linear Classification								
Logistic Regression	98.56%	87.11%	93.44%	86.23%	44.19%	0.00%	57.45%	17.69%
Sparse LDA	97.24%	94.29%	97.06%	93.83%	44.19%	0.00%	44.19%	0.00%
Partial Least Squares	97.28%	94.35%	97.10%	93.92%	44.19%	0.00%	44.68%	0.00%
Penalized GLM	99.48%	98.92%	99.38%	98.70%	48.84%	6.52%	68.09%	39.59%
Nearest Shrunken Centroids	91.24%	81.78%	91.10%	81.24%	65.12%	27.28%	78.72%	58.11%
Non-Linear Classification								
Neural Net	99.84%	99.67%	99.90%	99.79%	44.19%	0.00%	44.68%	0.00%
Support Vector Machine	99.55%	99.08%	99.52%	99.00%	55.81%	0.00%	55.32%	0.00%
K-Nearest Neighbor	93.51%	86.44%	93.46%	86.13%	55.81%	0.00%	68.09%	32.66%
Naïve Bayes	91.11%	81.78%	90.96%	81.18%	53.49%	6.72%	89.36%	78.77%
MARS	99.37%	98.70%	99.92%	99.83%	41.86%	-21.20%	91.49%	83.09%
Classification Trees & Rule Based								
Stochastic Gradient Boosting	96.99%	93.77%	96.70%	93.07%	58.14%	13.23%	95.74%	91.39%
Extreme Gradient Boosting	98.59%	97.09%	98.78%	97.45%	51.16%	-5.37%	97.87%	95.68%

Table 1: Base case Accuracy and Kappa

Method	Train ROC	Train SENS	Train SPEC	Test SENS	Test SPEC
Linear Classification					
Logistic Regression	98.57%	92.13%	94.91%	90.87%	95.11%
Sparse LDA	99.71%	96.13%	98.02%	95.59%	98.02%
Partial Least Squares	99.73%	96.23%	97.99%	95.74%	97.99%
Penalized GLM	99.97%	99.30%	99.60%	99.04%	99.60%
Nearest Shrunken Centroids	97.42%	87.49%	93.84%	86.76%	93.92%

Non-Linear Classification					
Neural Net	99.71%	99.62%	99.46%	99.80%	99.97%
Support Vector Machine	99.98%	99.46%	99.62%	99.44%	99.57%
K-Nearest Neighbor	98.59%	88.60%	96.92%	87.98%	97.03%
Naïve Bayes	97.33%	91.37%	90.93%	90.06%	91.55%
MARS	99.94%	99.19%	99.49%	99.90%	99.93%
Classification Trees & Rule Based					
Stochastic Gradient Boosting	99.65%	95.76%	97.85%	94.98%	97.82%
Extreme Gradient Boosting	99.92%	98.16%	98.89%	98.43%	99.01%

Table 2: Base case ROC, Sensitivity and Specificity

Predictor	Sum of Value
AWAY_OFF_EFF	48
HOME_OFF_EFF	42
DREB_SPREAD	33
AWAY_TS	16
AST_SPREAD	6
HOME_TS	5
HOME_eFG	5
AWAY_POSS	4
HOME_POSS	4
AWAY_eFG	4
PF_SPREAD	3
AWAY_PF	3
HOME_FGM	3
HOME_FG3A	2
AWAY_FGM	1
AWAY_FG3A	1

Table 3: Base case important variables

Method	Training Accuracy	Training Kappa	Test Accuracy	Test Kappa	Actual' Test Accuracy	Actual' Test Kappa
Linear Classification						
Logistic Regression	99.14%	98.22%	99.08%	98.10%	44.68%	0.00%
Penalized GLM	98.79%	97.50%	98,8%	97.52%	80.85%	62.13%
Nearest Shrunken Centroids	89.72%	78.45%	89.64%	78.40%	78.72%	58.11%
Non-Linear Classification						
K-Nearest Neighbor	91.33%	81.77%	91.48%	82.16%	65.96%	27.83%
Naïve Bayes	89.71%	78.74%	89.44%	78.27%	80.85%	62.80%
MARS	98.29%	96.46%	98.42%	96.73%	63.83%	20.66%
Classification Trees & Rule Based						

Stochastic Gradient Boosting	94.96%	89.52%	94.74%	89.08%	97.87%	95.72%
Extreme Gradient Boosting	97.70%	95.23%	97.80%	95.45%	95.74%	91.39%

Table 4: Alternate Case 1 Accuracy and Kappa

Method	Training Accuracy	Training Kappa	Test Accuracy	Test Kappa	Actual' Test Accuracy	Actual' Test Kappa
Linear Classification						
Logistic Regression	86.95%	72.80%	86.72%	72.69%	82.98%	65.57%
Penalized GLM	86.91%	72.72%	86.74%	72.73%	82.98%	65.57%
Nearest Shrunken Centroids	83.20%	64.18%	83.16%	64.76%	78.72%	56.96%
Non-Linear Classification						
K-Nearest Neighbor	85.16%	68.57%	85.00%	68.78%	82.98%	65.25%
Naïve Bayes	83.35%	65.53%	83.14%	65.51%	80.85%	61.79%
MARS	86.83%	72.54%	86.96%	82.64%	82.98%	65.57%
Classification Trees & Rule Based						
Stochastic Gradient Boosting	86.55%	71.91%	86.76%	72.76%	85.11%	70.01%
Extreme Gradient Boosting	86.68%	72.22%	86.66%	72.54%	82.98%	65.57%

Table 5: Alternate Case 2 Accuracy and Kappa

Method	Train ROC	Train SENS	Train SPEC	Test SENS	Test SPEC
Linear Classification					
Logistic Regression	99.86%	98.87%	99.32%	98.78%	99.29%
Penalized GLM	99.85%	98.32%	99.12%	98.29%	99.15%
Nearest Shrunken Centroids	96.47%	83.42%	94.06%	90.48%	69.23%
Non-Linear Classification					
K-Nearest Neighbor	97.68%	84.37%	96.13%	85.33%	95.76%
Naïve Bayes	96.38%	88.32%	90.63%	88.55%	90.06%
MARS	99.76%	97.81%	98.63%	98.00%	98.71%
Classification Trees & Rule Based					
Stochastic Gradient Boosting	99.10%	92.55%	96.62%	92.20%	96.51%
Extreme Gradient Boosting	99.79%	96.81%	98.31%	97.12%	98.27%

Table 6: Alternate Case 1 ROC, Sensitivity and Specificity

Method	Train ROC	Train SENS	Train SPEC	Test SENS	Test SPEC
Linear Classification					
Logistic Regression	94.36%	82.47%	90.00%	82.64%	89.72%
Penalized GLM	94.30%	82.33%	90.02%	82.58%	89.79%
Nearest Shrunken Centroids	91.59%	70.92%	91.54%	71.93%	91.42%
Non-Linear Classification					
K-Nearest Neighbor	93.16%	75.39%	91.80%	76.08%	91.56%

Naïve Bayes	91.42%	80.17%	85.51%	80.38%	85.17%
MARS	94.26%	82.32%	89.89%	82.64%	90.14%
Classification Trees & Rule Based					
Stochastic Gradient Boosting	93.98%	81.42%	90.04%	82.50%	89.90%
Extreme Gradient Boosting	94.06%	81.90%	89.94%	82.22%	89.93%

Table 7: Alternate Case 2 ROC, Sensitivity and Specificity

Row Labels	Sum of Value
DREB_SPREAD	30
HOME_FG_PCT	15
AWAY_FG_PCT	11
TO_SPREAD	9
AST_SPREAD	8
FTM_SPREAD	7
FG3_PCT_SPREAD	6
HOME_FGM	5
REB_SPREAD	5
HOME_FG3M	5
AWAY_FG3M	4
AWAY_FG	4
AWAY_FGM	4
HOME_FTM	3
HOME_REB	2
AWAY_FTM	1
FG3M_SPREAD	1

Table 8: Alternate Case 1 Important Variables

Row Labels	Sum of Value
DREB_SPREAD	35
AST_SPREAD	21
TO_SPREAD	15
REB_SPREAD	13
AWAY_DREB	7
PF_SPREAD	6
HOME_REB	5
HOME_PF	5
AWAY_PF	4
HOME_AST_RAT	4
HOME_DREB	2
HOME_OREB	1
BLK_SPREAD	1
STL_SPREAD	1

Table 9: Alternate Case 2 Important Variables

Figures

games

GAME_DATE_EST	GAME_ID	GAME_STATUS_TEXT	HOME_TEAM_ID	VISITOR_TEAM_ID	SEASON	TEAM_ID_home	PTS_home	FG_PCT_home	FT_PCT_home	FG3_PCT_home	AST_home	REB_home	TEAM_ID_away	PTS_away	FG_PCT_away	FT_PCT_away	FG3_PCT_away	AST_away	REB_away	HOME_TEAM_WINS
11/17/21	22100213	Final	1610612766	1610612764	2021	1610612766	97	0.438	0.5	0.313	30	59	1610612764	87	0.367	0.813	0.19	23	48	1
11/17/21	22100214	Final	1610612765	1610612754	2021	1610612765	97	0.425	0.75	0.286	16	42	1610612754	89	0.418	0.737	0.243	14	43	1
11/17/21	22100215	Final	1610612737	1610612738	2021	1610612737	110	0.506	0.833	0.351	28	40	1610612738	99	0.44	0.824	0.268	24	42	1
11/17/21	22100216	Final	1610612751	1610612739	2021	1610612751	109	0.458	0.84	0.375	29	47	1610612739	99	0.393	0.857	0.25	20	50	1
11/17/21	22100217	Final	1610612748	1610612740	2021	1610612748	113	0.483	0.824	0.375	29	39	1610612740	98	0.44	0.786	0.286	18	38	1
11/17/21	22100218	Final	1610612752	1610612753	2021	1610612752	98	0.42	0.667	0.327	25	45	1610612753	104	0.406	0.833	0.289	20	51	0
11/17/21	22100219	Final	1610612749	1610612747	2021	1610612749	109	0.46	0.708	0.324	22	52	1610612747	102	0.417	0.833	0.279	24	48	1
11/17/21	22100220	Final	1610612750	1610612758	2021	1610612750	107	0.5	0.833	0.278	20	35	1610612758	97	0.366	0.76	0.263	19	47	1
11/17/21	22100221	Final	1610612760	1610612745	2021	1610612760	101	0.446	0.563	0.256	18	56	1610612745	89	0.356	0.684	0.286	16	46	1
11/17/21	22100222	Final	1610612756	1610612742	2021	1610612756	105	0.441	0.714	0.333	27	49	1610612742	98	0.413	1	0.5	28	50	1
11/17/21	22100223	Final	1610612757	1610612741	2021	1610612757	112	0.44	0.964	0.355	26	44	1610612741	107	0.47	0.8	0.448	23	36	1
11/16/21	22100210	Final	1610612751	1610612744	2021	1610612751	99	0.386	0.778	0.278	21	38	1610612744	117	0.519	0.786	0.325	25	53	0
11/16/21	22100211	Final	1610612762	1610612755	2021	1610612762	120	0.517	0.632	0.421	27	56	1610612755	85	0.367	0.765	0.207	19	42	1
11/16/21	22100212	Final	1610612746	1610612759	2021	1610612746	106	0.435	0.909	0.316	16	52	1610612759	92	0.432	0.714	0.227	26	41	1
11/15/21	22100199	Final	1610612739	1610612738	2021	1610612739	92	0.386	0.75	0.4	18	42	1610612738	98	0.462	0.8	0.345	23	46	0
11/15/21	22100200	Final	1610612765	1610612758	2021	1610612765	107	0.406	0.941	0.277	23	47	1610612758	129	0.531	0.818	0.485	32	47	0
11/15/21	22100201	Final	1610612764	1610612740	2021	1610612764	105	0.464	0.68	0.357	23	42	1610612740	100	0.405	0.853	0.524	20	45	1
11/15/21	22100202	Final	1610612737	1610612753	2021	1610612737	129	0.553	0.656	0.467	32	46	1610612753	111	0.453	0.643	0.372	30	41	1

Figure 1: games.csv dataset

games_details

GAME_ID	TEAM_ID	TEAM_ABBREVIATION	TEAM_CITY	PLAYER_ID	PLAYER_NAME	NICKNAME	START_POSITION	COMMENT	MIN	FGM	FGA	FG_PCT	FG3M	FG3A	FG3_PCT	FTM	FTA	FT_PCT	OREB	DREB	REB	AST	STL	BLK	TO	PF	PTS	PLUS_MINUS
22100213	1610612764	WAS	Washington	203484	Kentavious Caldwell-Pope	Kentavious	F		27:41:00	1	6	0.167	0	5	0	1	1	1	1	5	6	2	1	0	1	0	3	2
22100213	1610612764	WAS	Washington	1628398	Kyle Kuzma	Kyle	F		30:28:00	2	12	0.167	1	8	0.125	0	0	0	1	4	5	3	1	2	1	1	5	-14
22100213	1610612764	WAS	Washington	1629655	Daniel Gafford	Daniel	C		24:21:00	9	12	0.75	0	0	0	2	5	0.4	2	7	9	1	2	1	1	4	20	-2
22100213	1610612764	WAS	Washington	203078	Bradley Beal	Bradley	G		35:07:00	9	20	0.45	5	11	0.455	1	1	1	0	3	3	7	2	0	2	3	24	-9
22100213	1610612764	WAS	Washington	203915	Spencer Dinwiddie	Spencer	G		28:34:00	0	5	0	0	4	0	0	0	0	0	3	3	2	0	0	2	1	0	-5
22100213	1610612764	WAS	Washington	203526	Raul Neto	Raul			17:59	1	6	0.167	0	2	0	0	0	0	0	0	0	3	0	0	3	2	2	-8
22100213	1610612764	WAS	Washington	1626149	Montrezl Harrell	Montrezl			22:12	6	9	0.667	0	0	0	3	3	1	5	4	9	2	0	1	0	0	15	-11
22100213	1610612764	WAS	Washington	1630166	Deni Avdija	Deni			24:59:00	2	9	0.222	1	7	0.143	4	4	1	2	9	11	0	0	1	0	1	9	-10
22100213	1610612764	WAS	Washington	1630557	Corey Kispert	Corey			11:25	2	4	0.5	0	2	0	2	2	1	0	1	1	1	0	0	0	0	6	-1
22100213	1610612764	WAS	Washington	1628988	Aaron Holiday	Aaron			12:53	1	6	0.167	1	3	0.333	0	0	0	0	0	0	1	1	0	0	0	3	-1
22100213	1610612764	WAS	Washington	1630264	Anthony Gill	Anthony			1:27	0	0	0	0	0	0	0	0	0	0	1	1	0	0	0	0	0	3
22100213	1610612764	WAS	Washington	1630225	Isaiah Todd	Isaiah			1:27	0	1	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	3
22100213	1610612764	WAS	Washington	1630555	Joel Ayayi	Joel			1:27	0	0	0	0	0	0	0	0	0	0	0	0	0	1	0	0	0	0	3

Figure 2: games_details.csv dataset

players

PLAYER_NAME	TEAM_ID	PLAYER_ID	SEASON
Royce O'Neale	1610612762	1626220	2019
Bojan Bogdanovic	1610612762	202711	2019
Rudy Gobert	1610612762	203497	2019
Donovan Mitchell	1610612762	1628378	2019
Mike Conley	1610612762	201144	2019
Joe Ingles	1610612762	204060	2019

Figure 3: players.csv dataset

ranking

TEAM_ID	LEAGUE_ID	SEASON_ID	STANDINGSDATE	CONFERENCE	TEAM	G	W	L	W_PCT	HOME_RECORD	ROAD_RECORD	RETURNTOPLAY
1610612744	0	22021	2021-11-17	West	Golden State	14	12	2	0.857	8-1	4-1	
1610612756	0	22021	2021-11-17	West	Phoenix	14	11	3	0.786	6-2	5-1	
1610612742	0	22021	2021-11-17	West	Dallas	14	9	5	0.643	6-1	3-4	
1610612743	0	22021	2021-11-17	West	Denver	14	9	5	0.643	7-1	2-4	
1610612746	0	22021	2021-11-17	West	LA Clippers	14	9	5	0.643	7-3	2-2	
1610612762	0	22021	2021-11-17	West	Utah	14	9	5	0.643	5-2	4-3	
1610612747	0	22021	2021-11-17	West	L.A. Lakers	16	8	8	0.5	7-5	1-3	
1610612757	0	22021	2021-11-17	West	Portland	16	8	8	0.5	7-1	1-7	
1610612763	0	22021	2021-11-17	West	Memphis	14	7	7	0.5	5-3	2-4	
1610612760	0	22021	2021-11-17	West	Oklahoma City	14	6	8	0.429	4-4	2-4	
1610612758	0	22021	2021-11-17	West	Sacramento	15	6	9	0.4	2-4	4-5	
1610612750	0	22021	2021-11-17	West	Minnesota	14	5	9	0.357	3-6	2-3	
1610612759	0	22021	2021-11-17	West	San Antonio	14	4	10	0.286	2-4	2-6	

Figure 4: ranking.csv dataset

```
> summary(games_details)
    GAME_ID            TEAM_ID          TEAM_ABBREVIATION   TEAM_CITY           PLAYER_ID         PLAYER_NAME
 Min.   :10300001   Min.   :1.611e+09   Length:626111      Length:626111      Min.   :1.500e+01   Length:626111
 1st Qu.:20600922   1st Qu.:1.611e+09   Class :character   Class :character   1st Qu.:2.419e+03   Class :character
 Median :21200224   Median :1.611e+09   Mode  :character   Mode  :character   Median :2.011e+05   Mode  :character
 Mean   :21645531   Mean   :1.611e+09                                         Mean   :3.452e+05
 3rd Qu.:21700202   3rd Qu.:1.611e+09                                         3rd Qu.:2.031e+05
 Max.   :52000211   Max.   :1.611e+09                                         Max.   :1.963e+09

   NICKNAME         START_POSITION       COMMENT              MIN                FGM              FGA
 Length:626111      Length:626111      Length:626111      Length:626111      Min.   : 0.00    Min.   : 0.00
 Class :character   Class :character   Class :character   Class :character   1st Qu.: 1.00    1st Qu.: 3.00
 Mode  :character   Mode  :character   Mode  :character   Mode  :character   Median : 3.00    Median : 7.00
                                                                             Mean   : 3.57    Mean   : 7.88
                                                                             3rd Qu.: 5.00    3rd Qu.:11.00
                                                                             Max.   :28.00    Max.   :50.00
                                                                             NA's   :102360   NA's   :102360
     FG_PCT            FG3M             FG3A           FG3_PCT            FTM              FTA             FT_PCT
 Min.   :0.00     Min.   : 0.00    Min.   : 0.00    Min.   :0.00     Min.   : 0.00    Min.   : 0.00    Min.   :0.00
 1st Qu.:0.25     1st Qu.: 0.00    1st Qu.: 0.00    1st Qu.:0.00     1st Qu.: 0.00    1st Qu.: 0.00    1st Qu.:0.00
 Median :0.43     Median : 0.00    Median : 1.00    Median :0.00     Median : 1.00    Median : 2.00    Median :0.50
 Mean   :0.42     Mean   : 0.75    Mean   : 2.12    Mean   :0.20     Mean   : 1.74    Mean   : 2.29    Mean   :0.44
 3rd Qu.:0.57     3rd Qu.: 1.00    3rd Qu.: 3.00    3rd Qu.:0.38     3rd Qu.: 3.00    3rd Qu.: 4.00    3rd Qu.:0.91
 Max.   :1.00     Max.   :14.00    Max.   :24.00    Max.   :1.00     Max.   :26.00    Max.   :39.00    Max.   :1.00
 NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360
      OREB             DREB             REB              AST              STL              BLK              TO
 Min.   : 0.00    Min.   : 0.00    Min.   : 0.00    Min.   : 0.00    Min.   : 0.00    Min.   : 0.00    Min.   : 0.00
 1st Qu.: 0.00    1st Qu.: 1.00    1st Qu.: 1.00    1st Qu.: 0.00    1st Qu.: 0.00    1st Qu.: 0.00    1st Qu.: 0.00
 Median : 1.00    Median : 2.00    Median : 3.00    Median : 1.00    Median : 0.00    Median : 0.00    Median : 1.00
 Mean   : 1.03    Mean   : 3.03    Mean   : 4.06    Mean   : 2.09    Mean   : 0.72    Mean   : 0.46    Mean   : 1.33
 3rd Qu.: 2.00    3rd Qu.: 4.00    3rd Qu.: 6.00    3rd Qu.: 3.00    3rd Qu.: 1.00    3rd Qu.: 1.00    3rd Qu.: 2.00
 Max.   :18.00    Max.   :25.00    Max.   :31.00    Max.   :25.00    Max.   :10.00    Max.   :12.00    Max.   :12.00
 NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360   NA's   :102360
       PF               PTS            PLUS_MINUS
 Min.   : 0.00    Min.   : 0.00    Min.   :-57
 1st Qu.: 1.00    1st Qu.: 3.00    1st Qu.: -7
 Median : 2.00    Median : 8.00    Median :  0
 Mean   : 2.01    Mean   : 9.64    Mean   :  0
 3rd Qu.: 3.00    3rd Qu.:14.00    3rd Qu.:  6
 Max.   :15.00    Max.   :81.00    Max.   : 57
 NA's   :102360   NA's   :102360   NA's   :126021
```

Figure 5: Summary of games_details.csv dataset

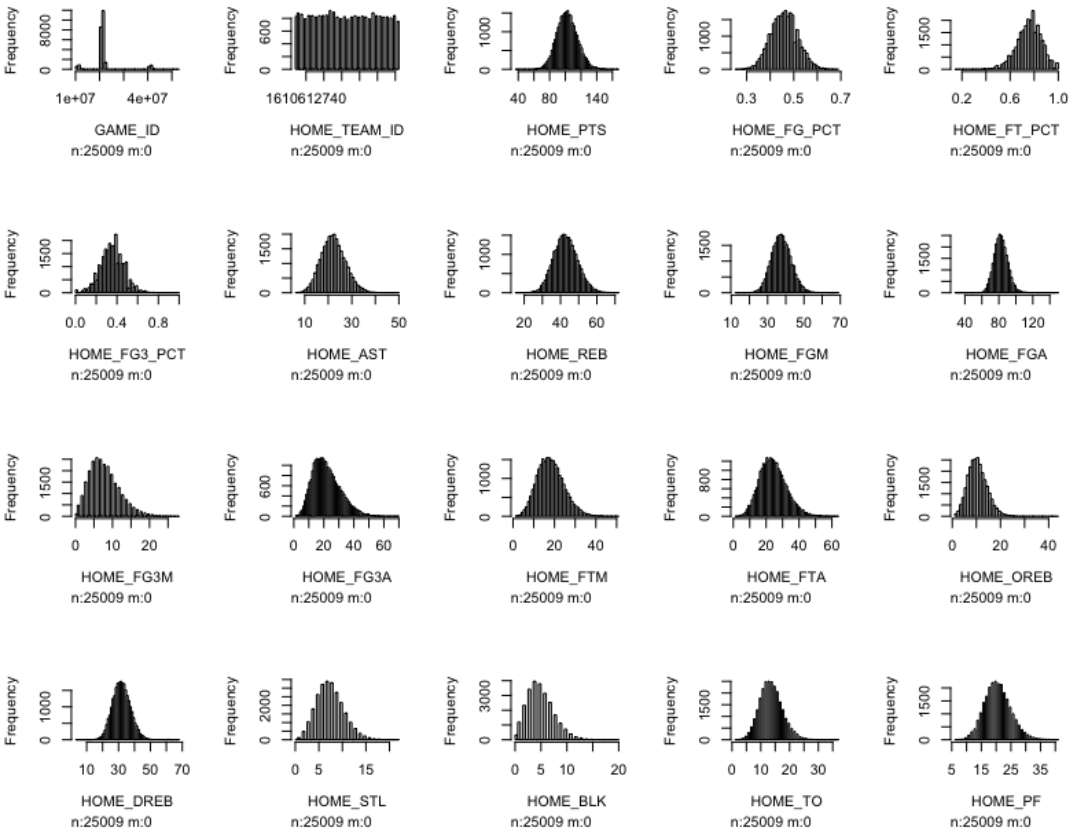

Figure 6: Histogram 1 of all_games.csv

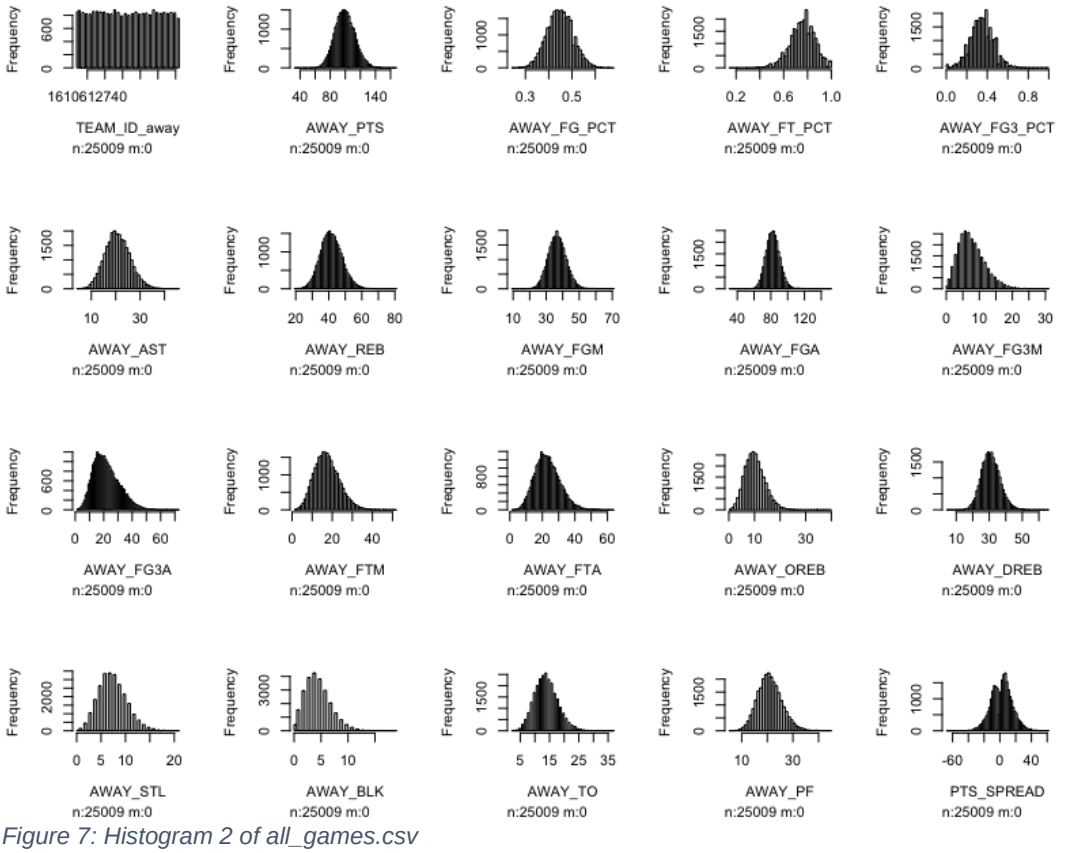

Figure 7: Histogram 2 of all_games.csv

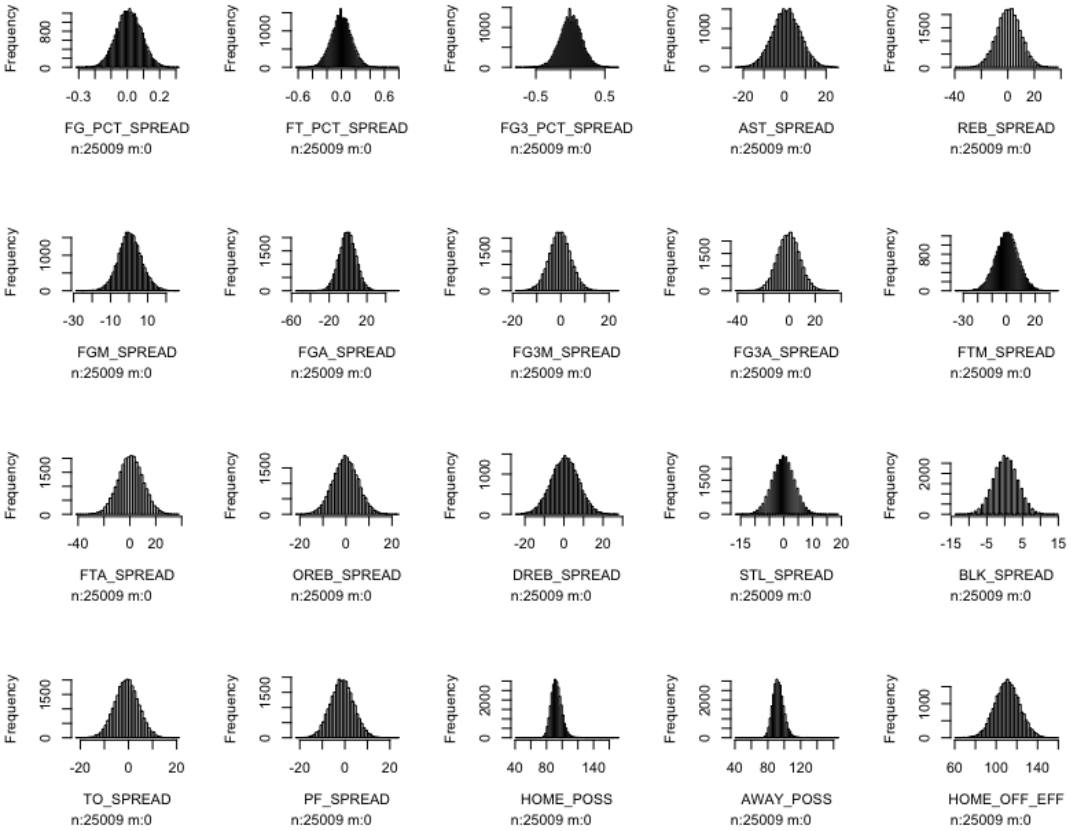

Figure 8: Histogram 3 of all_games.csv

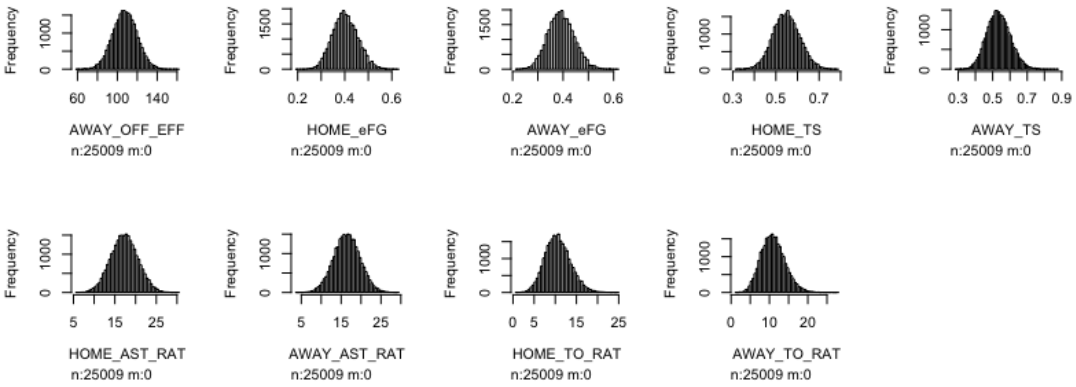

Figure 9: Histogram 4 of all_games.csv

```
> varImp(knnFit)
ROC curve variable importance

  only 20 most important vari

               Importance
FG_PCT_SPREAD      100.00
FGM_SPREAD          92.74
HOME_OFF_EFF        78.80
DREB_SPREAD         78.52
AWAY_OFF_EFF        77.92
AWAY_TS             76.33
HOME_TS             70.57
AST_SPREAD          69.30
HOME_FG_PCT         66.37
HOME_eFG            65.97
AWAY_FG_PCT         65.86
FG3_PCT_SPREAD      64.77
AWAY_eFG            62.54
REB_SPREAD          54.20
AWAY_FGM            50.61
HOME_FGM            50.53
AWAY_DREB           49.53
FG3M_SPREAD         48.09
HOME_AST_RAT        48.01
HOME_FG3_PCT        46.01
```

```
> varImp(plsFit)
pls variable importance

  only 20 most important

               Overall
FG_PCT_SPREAD   100.00
FGM_SPREAD       93.37
HOME_OFF_EFF     81.52
AWAY_OFF_EFF     81.21
DREB_SPREAD      80.46
AWAY_TS          78.89
AST_SPREAD       73.75
HOME_TS          71.95
HOME_FG_PCT      68.75
HOME_eFG         68.34
AWAY_FG_PCT      67.89
FG3_PCT_SPREAD   64.90
AWAY_eFG         64.34
REB_SPREAD       57.81
AWAY_FGM         52.49
AWAY_DREB        52.27
HOME_AST_RAT     51.89
HOME_FGM         51.69
FG3M_SPREAD      50.84
AWAY_AST_RAT     48.99
```

```
> varImp(spLDAFit)
ROC curve variable importance

  only 20 most important vari

               Importance
FG_PCT_SPREAD      100.00
FGM_SPREAD          93.07
DREB_SPREAD         78.93
HOME_OFF_EFF        78.89
AWAY_OFF_EFF        78.11
AWAY_TS             76.23
HOME_TS             70.68
AST_SPREAD          70.14
HOME_FG_PCT         66.82
HOME_eFG            66.50
AWAY_FG_PCT         65.65
FG3_PCT_SPREAD      64.59
AWAY_eFG            62.22
REB_SPREAD          55.12
AWAY_FGM            51.17
HOME_FGM            50.41
AWAY_DREB           50.09
HOME_AST_RAT        48.59
FG3M_SPREAD         47.97
HOME_FG3_PCT        46.53
```

Figure 10: variable importance for KNN, PLS and SparseLDA models

```
> summary(data)
    HOME_PTS      HOME_FG_PCT      HOME_FT_PCT      HOME_FG3_PCT      HOME_AST        HOME_REB        HOME_FGM        HOME_FGA        HOME_FG3M       HOME_FG3A       HOME_FTM        HOME_FTA
 Min.   : 36.0   Min.   :0.2500   Min.   :0.1430   Min.   :0.000   Min.   : 6.00   Min.   :15.00   Min.   :12.00   Min.   : 29.00   Min.   : 0.000   Min.   : 1.00   Min.   : 1.00   Min.   : 1.00
 1st Qu.: 94.0   1st Qu.:0.4210   1st Qu.:0.6960   1st Qu.:0.286   1st Qu.:19.00   1st Qu.:39.00   1st Qu.:34.00   1st Qu.: 77.00   1st Qu.: 5.000   1st Qu.:16.00   1st Qu.:14.00   1st Qu.:19.00
 Median :102.0   Median :0.4590   Median :0.7650   Median :0.357   Median :22.00   Median :43.00   Median :38.00   Median : 83.00   Median : 8.000   Median :21.00   Median :18.00   Median :24.00
 Mean   :102.9   Mean   :0.4602   Mean   :0.7591   Mean   :0.356   Mean   :22.67   Mean   :43.31   Mean   :38.15   Mean   : 83.08   Mean   : 8.034   Mean   :22.36   Mean   :18.74   Mean   :24.72
 3rd Qu.:111.0   3rd Qu.:0.5000   3rd Qu.:0.8280   3rd Qu.:0.429   3rd Qu.:26.00   3rd Qu.:48.00   3rd Qu.:42.00   3rd Qu.: 88.00   3rd Qu.:10.000   3rd Qu.:28.00   3rd Qu.:23.00   3rd Qu.:30.00
 Max.   :168.0   Max.   :0.6840   Max.   :1.0000   Max.   :1.000   Max.   :50.00   Max.   :72.00   Max.   :69.00   Max.   :149.00   Max.   :28.000   Max.   :70.00   Max.   :51.00   Max.   :64.00
   HOME_OREB       HOME_DREB       HOME_STL        HOME_BLK        HOME_TO         HOME_PF        AWAY_PTS      AWAY_FG_PCT     AWAY_FT_PCT     AWAY_FG3_PCT     AWAY_AST        AWAY_REB
 Min.   : 1.00   Min.   : 3.0   Min.   : 0.000   Min.   : 0.000   Min.   : 1.00   Min.   : 6.00   Min.   : 33   Min.   :0.244   Min.   :0.1430   Min.   :0.0000   Min.   : 4.00   Min.   :19.00
 1st Qu.: 8.00   1st Qu.:29.0   1st Qu.: 6.000   1st Qu.: 3.000   1st Qu.:11.00   1st Qu.:18.00   1st Qu.: 91   1st Qu.:0.412   1st Qu.:0.6920   1st Qu.:0.2780   1st Qu.:18.00   1st Qu.:38.00
 Median :11.00   Median :32.0   Median : 7.000   Median : 5.000   Median :14.00   Median :21.00   Median :100   Median :0.448   Median :0.7630   Median :0.3500   Median :21.00   Median :42.00
 Mean   :10.99   Mean   :32.4   Mean   : 7.643   Mean   : 5.117   Mean   :13.83   Mean   :20.79   Mean   :100   Mean   :0.449   Mean   :0.7577   Mean   :0.3495   Mean   :21.32   Mean   :42.03
 3rd Qu.:13.00   3rd Qu.:36.0   3rd Qu.: 9.000   3rd Qu.: 7.000   3rd Qu.:16.00   3rd Qu.:24.00   3rd Qu.:109   3rd Qu.:0.486   3rd Qu.:0.8290   3rd Qu.:0.4210   3rd Qu.:25.00   3rd Qu.:46.00
 Max.   :44.00   Max.   :68.0   Max.   :22.000   Max.   :20.000   Max.   :37.00   Max.   :41.00   Max.   :168   Max.   :0.674   Max.   :1.0000   Max.   :1.0000   Max.   :46.00   Max.   :81.00
    AWAY_FGM        AWAY_FGA        AWAY_FG3M       AWAY_FG3A       AWAY_FTM        AWAY_FTA       AWAY_OREB       AWAY_DREB       AWAY_STL        AWAY_BLK        AWAY_TO         AWAY_PF
 Min.   : 9.00   Min.   : 31.00   Min.   : 0.000   Min.   : 1.00   Min.   : 1.00   Min.   : 1.00   Min.   : 0.00   Min.   : 4.00   Min.   : 0.000   Min.   : 0.000   Min.   : 2.00   Min.   : 5.0
 1st Qu.:34.00   1st Qu.: 77.00   1st Qu.: 5.000   1st Qu.:16.00   1st Qu.:13.00   1st Qu.:18.00   1st Qu.: 8.00   1st Qu.:28.00   1st Qu.: 6.000   1st Qu.: 3.000   1st Qu.:11.00   1st Qu.:18.0
 Median :37.00   Median : 83.00   Median : 7.000   Median :21.00   Median :17.00   Median :23.00   Median :10.00   Median :31.00   Median : 7.000   Median : 4.000   Median :14.00   Median :21.0
 Mean   :37.21   Mean   : 83.03   Mean   : 7.895   Mean   :22.39   Mean   :17.89   Mean   :23.63   Mean   :10.67   Mean   :31.43   Mean   : 7.597   Mean   : 4.614   Mean   :14.11   Mean   :21.5
 3rd Qu.:41.00   3rd Qu.: 88.00   3rd Qu.:10.000   3rd Qu.:28.00   3rd Qu.:22.00   3rd Qu.:28.00   3rd Qu.:13.00   3rd Qu.:35.00   3rd Qu.: 9.000   3rd Qu.: 6.000   3rd Qu.:17.00   3rd Qu.:24.0
 Max.   :71.00   Max.   :152.00   Max.   :31.000   Max.   :73.00   Max.   :52.00   Max.   :64.00   Max.   :40.00   Max.   :66.00   Max.   :21.000   Max.   :19.000   Max.   :37.00   Max.   :45.0
```

Figure 11: Summary of Raw NBA Statistics

Code Snippets

```
games_details = games_details %>%
  mutate_if(is.numeric, ~replace_na(., 0))
```

Code 1: Replacing all "NA" with 0 in games_details dataset

```
add_game_stat = games_details %>% group_by (GAME_ID, TEAM_ID) %>%
  summarise(FGM = sum(FGM), FGA = sum(FGA),FG3M = sum(FG3M), FG3A = sum(FG3A),
            FTM = sum(FTM), FTA = sum(FTA),OREB = sum(OREB), DREB = sum(DREB),
            STL = sum(STL), BLK = sum(BLK),TO = sum(TO), PF = sum(PF))
```

Code 2: Combining individual player statistics into single game totals

```
games1 = merge(games, add_game_stat, by.x = c("GAME_ID","HOME_TEAM_ID"), by.y = c("GAME_ID","TEAM_ID"))
colnames(games1)[22:33] = paste("HOME", colnames(games1)[22:33], sep = "_")

all_games = merge(games1, add_game_stat, by.x = c("GAME_ID","VISITOR_TEAM_ID"), by.y = c("GAME_ID","TEAM_ID"))
colnames(all_games)[34:45] = paste("AWAY", colnames(all_games)[34:45], sep = "_")
```

Code 3: Merging the calculated game stat totals with the already existing ones

```
all_games$PTS_SPREAD = all_games$HOME_PTS - all_games$AWAY_PTS
all_games$FG_PCT_SPREAD = all_games$HOME_FG_PCT - all_games$AWAY_FG_PCT
all_games$FT_PCT_SPREAD = all_games$HOME_FT_PCT - all_games$AWAY_FT_PCT
all_games$FG3_PCT_SPREAD = all_games$HOME_FG3_PCT - all_games$AWAY_FG3_PCT
all_games$AST_SPREAD = all_games$HOME_AST - all_games$AWAY_AST
all_games$REB_SPREAD = all_games$HOME_REB - all_games$AWAY_REB
all_games$FGM_SPREAD = all_games$HOME_FGM - all_games$AWAY_FGM
all_games$FGA_SPREAD = all_games$HOME_FGA - all_games$AWAY_FGA
all_games$FG3M_SPREAD = all_games$HOME_FG3M - all_games$AWAY_FG3M
all_games$FG3A_SPREAD = all_games$HOME_FG3A - all_games$AWAY_FG3A
all_games$FTM_SPREAD = all_games$HOME_FTM - all_games$AWAY_FTM
all_games$FTA_SPREAD = all_games$HOME_FTA - all_games$AWAY_FTA
all_games$OREB_SPREAD = all_games$HOME_OREB - all_games$AWAY_OREB
all_games$DREB_SPREAD = all_games$HOME_DREB - all_games$AWAY_DREB
all_games$STL_SPREAD = all_games$HOME_STL - all_games$AWAY_STL
all_games$BLK_SPREAD = all_games$HOME_BLK - all_games$AWAY_BLK
all_games$TO_SPREAD = all_games$HOME_TO - all_games$AWAY_TO
all_games$PF_SPREAD = all_games$HOME_PF - all_games$AWAY_PF
```

Code 4: Stat spread formulas

```
all_games_adv$HOME_POSS = 0.96 * (x$HOME_FGA + (0.44 * x$HOME_FTA) - x$HOME_OREB + x$HOME_TO)
all_games_adv$AWAY_POSS = 0.96 * (x$AWAY_FGA + (0.44 * x$AWAY_FTA) - x$AWAY_OREB + x$AWAY_TO)
#TEAM POSSESSION

all_games_adv$HOME_OFF_EFF = (x$HOME_PTS * 100)/all_games_adv$HOME_POSS
all_games_adv$AWAY_OFF_EFF = (x$AWAY_PTS * 100)/all_games_adv$AWAY_POSS
#TEAM OFFENSIVE EFFCIENCY

all_games_adv$HOME_eFG = (x$HOME_FGM + (0.5*x$HOME_FG3M))/(x$HOME_FGA+x$HOME_FG3A)
all_games_adv$AWAY_eFG = (x$AWAY_FGM + (0.5*x$AWAY_FG3M))/(x$AWAY_FGA+x$AWAY_FG3A)
#EFFECTIVE FIELD GOAL

all_games_adv$HOME_TS = (0.5*x$HOME_PTS)/(x$HOME_FGA+(0.44*x$HOME_FTA))
all_games_adv$AWAY_TS = (0.5*x$AWAY_PTS)/(x$AWAY_FGA+(0.44*x$HOME_FTA))
#TRUE SHOOTING

all_games_adv$HOME_AST_RAT = (x$HOME_AST*100)/(x$HOME_FGA+(x$HOME_FTA*0.44)+x$HOME_AST+x$HOME_TO)
all_games_adv$AWAY_AST_RAT = (x$AWAY_AST*100)/(x$AWAY_FGA+(x$AWAY_FTA*0.44)+x$AWAY_AST+x$AWAY_TO)
#ASSIST RATIO

all_games_adv$HOME_TO_RAT = (x$HOME_TO*100)/(x$HOME_FGA+(x$HOME_FTA*0.44)+x$HOME_AST+x$HOME_TO)
all_games_adv$AWAY_TO_RAT = (x$AWAY_TO*100)/(x$AWAY_FGA+(x$AWAY_FTA*0.44)+x$AWAY_AST+x$AWAY_TO)
#TURNOVER RATIO
```

Code 5: Advanced stat calculations

```
data_nzv = nearZeroVar(data, freqCut = 9/1)
data_nzv
#near zero variance
```

Code 6: Near Zero Variance exploration code

```r
#Load data
cleaned_data <- read.csv("all_games.csv")
names(which(colSums(is.na(cleaned_data))>0))
colSums(is.na(cleaned_data))

#Split data 80/20
sample = createDataPartition(cleaned_data$GAME_ID, p=0.8, list=FALSE)

#Remove ID columns and points columns
no_points =  subset(cleaned_data, select=-c(GAME_ID,HOME_TEAM_ID,HOME_PTS,AWAY_PTS,TEAM_ID_away,PTS_SPREAD,FG_PCT_SPREAD,FGM_SPREAD))#,GAME_DATE_EST))
spread_all_games = no_points[35:62]#If you only want to use spread variables

train_sp <- no_points[sample,]
test_sp <- no_points[-sample,]

train_x <- train_sp[,-35]
test_x <- test_sp[,-35]

#Turn win loss column into factor for classification
train_sp$HOME_TEAM_WINS <- as.factor(train_sp$HOME_TEAM_WINS)
levels(train_sp$HOME_TEAM_WINS) <- c("Loss", "Win")
test_sp$HOME_TEAM_WINS <- as.factor(test_sp$HOME_TEAM_WINS)
levels(test_sp$HOME_TEAM_WINS) <- c("Loss", "Win")
names(which(colSums(is.na(cleaned_data))>0))

#Train Control
ctrl <- trainControl(method = "LGOCV",
                     summaryFunction = twoClassSummary,
                     classProbs = TRUE,
                     savePredictions = TRUE)

#Data for Final Test
data_2022 <- read.csv("final2_test_data.csv")
no_points_2022 =  subset(data_2022, select=-c(HOME_PTS,AWAY_PTS,PTS_SPREAD,FG_PCT_SPREAD,FGM_SPREAD,Teams))
no_points_2022$HOME_WINS <- as.factor(no_points_2022$HOME_WINS)
levels(no_points_2022$HOME_WINS) <- c("Loss", "Win")
no_points_2022_x = no_points_2022[,-35]

#Set Consistent Seed
set.seed(1056)
```

Code 7: The pre-model code used to prepare datasets, train control and setting seed for all models

```r
############### LOGISTIC REGRESSION ###############
#Model Tuning
lrTune = train(
  x = train_x,
  y = train_sp$HOME_TEAM_WINS,
  method = "glm",
  preProc = c("center","scale"),
  metric = "ROC",
  trControl = ctrl
)
lrTune

#Model Tune Results
lrTuneCM = confusionMatrix(lrTune, norm = "none")
lrTuneCM
lrTuneKappa = kappa(lrTuneCM$table)
lrTuneKappa

#Variable Importance
varImp(lrTune)

#Model Testing and Results
lrPred = predict(lrTune, test_x)

lrCM = confusionMatrix(lrPred, reference = test_sp$HOME_TEAM_WINS)
lrCM

#Model Final Test
lrPred_2022 = predict(lrTune, no_points_2022_x)
lrPred_2022

lrPred_2022CM = confusionMatrix(lrPred_2022, reference = no_points_2022$HOME_WINS)
lrPred_2022CM

############### END LOGISTIC REGRESSION ##################
```

Code 8: Standardized approach for training and testing all models, some models may differ slightly