Using Cumulative Statistics to Predict the Spread, Total, and OREB of NBA Games

Authors:

Jacob Andrews, Srinidhi Ekkurthi, Morgan Hall, Nishtha Mukherji, Adam Parks, Yanchen Xie

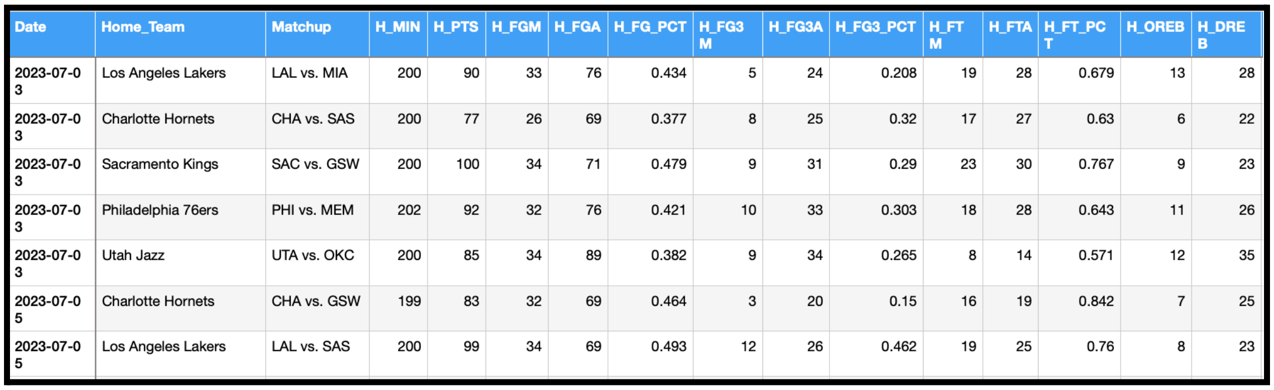
INTRODUCTION TO THE DATA

In the world of sports betting, sportsbooks use extremely advanced predictive models to set their own lines for the public to place their bets on. These predictions are scarily accurate on a game-to-game basis and help evenly distribute the money between the bettors and the book. For this project, our group used various sources of NBA statistics as well as our own variables to create our own predictive models to project different outcomes for NBA games such as point total, spread, and offensive rebounding.

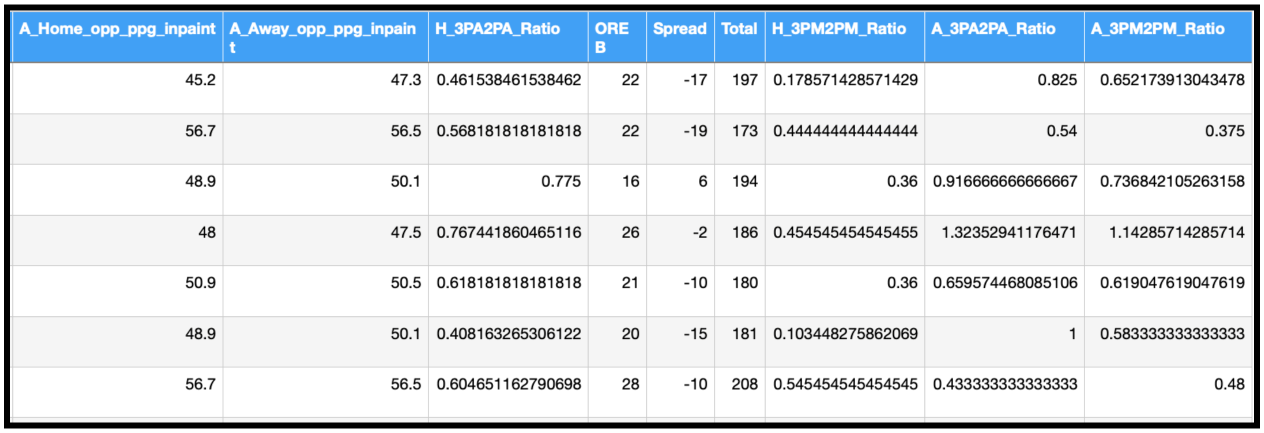
The primary dataset for this project was created through scraping web data off of the NBA Stats portion of Team Rankings' website. The variables scraped for each team were: Defensive Efficiency, Opponent Defensive Efficiency, Offensive Efficiency, Opponent Offensive Efficiency, Free Throws Made per 100 baskets, Opponent Free Throws Made per 100 baskets, Offensive Rebound Percent, Opponent Offensive Rebound Percent, Offensive Rebounds per Game, Opponent Offensive Rebounds per Game, Points per Game, Opponent Points per Game, Points per Game in Paint, Opponent Points per Game in Paint, Steals per Defensive Play, Opponent Steals per Defensive Play, Steals per Game, and Opponent Steals per Game from each day between 7/3/2023 and 4/9/2024. Each variable was scraped with the season average, home game average, away game average, and the past three game average for each team. The scraping of the data was done through Python using the packages pandas, requests, BeautifulSoup, datetime, and time. The "pandas" package was used for storing the data, "requests" was used for sending the GET requests to the website to retrieve the HTML content, "BeautifulSoup" was used to extract the data elements, "datetime" was used to format the dates, and "time" was used to include the delay so that the requests to the website wouldn't crash the model. The scraping process involved iterating through a range of dates starting from 7/3/2023 up until the current date. Through each iteration, the date within the specified range was formatted and appended to the base URL to the specific statistic to construct the target URL for data retrieval. The HTML content of the target URL was then parsed using BeautifulSoup to extract the relevant data table and the extracted data rows were converted into pandas DataFrames for each date, with an additional column indicating the date for which the data were retrieved. This process was done 18 times for each of the aforementioned variables to create 18 datasets with the season average, home game average, away game average, and the past three game average of each variable for each team.

After all the data was scraped into 18 different csv files, they were each loaded into RStudio where they were cleaned and merged. The merging process was conducted using the “Reduce” function, which iteratively merged the datasets based on their common columns of “Team” and “Date”. This merging led to all of the files being in one dataframe that contained a comprehensive set of our variables for each NBA team across each day. After merging, the dataset was transformed to standardize column names and ensure consistency across different metrics. Column prefixes such as “H_” and “A_” were added to distinguish between home and away game statistics and “Current_Season” and “Past3_” for the current season and past 3 game statistics. Additionally, inconsistent team names were standardized to ensure uniformity and accuracy in data analysis. The final dataset underwent validation and quality control measures to remove rows with missing or invalid data before downloading as our finished csv.

The other dataset used was the one which was created by using nba_api in Python which scraped the box scores from 7/3/2023 up until the present. This code was given to us and originally only scraping up until 12/31/2023, but we modified it to not have date restrictions and scrape up until the present. To create our final dataset for modeling, we needed to merge the game dataset with the cumulative statistics dataset so that for each game from 3/7/2023 to 4/9/2024 we have the cumulative statistics for the home and away team from that date. This process was done through an iterative loop in RStudio. To start, we initialize an empty list (merged_game_data_list) to store the merged game data for each game in the dataset. A loop was then implemented to iterate over each game in the dataset, where for each game, the corresponding cumulative statistics data is filtered based on the game date and then filtered to the specific teams for each game. The filtered cumulative statistics data is then joined with the game-specific data for the current game using left joins with “A_” for the away team and “H_” for the home team. This loop iterated through each game to create our final merged dataset.



Date	Home_Team	Matchup	H_MIN	H_PTS	H_FGM	H_FGA	H_FG_PCT	H_FG3M	H_FG3A	H_FG3_PCT	H_FTM	H_FTA	H_FT_PCT	H_OREB	H_DREB
2023-07-03	Los Angeles Lakers	LAL vs. MIA	200	90	33	76	0.434	5	24	0.208	19	28	0.679	13	28
2023-07-03	Charlotte Hornets	CHA vs. SAS	200	77	26	69	0.377	8	25	0.32	17	27	0.63	6	22
2023-07-03	Sacramento Kings	SAC vs. GSW	200	100	34	71	0.479	9	31	0.29	23	30	0.767	9	23
2023-07-03	Philadelphia 76ers	PHI vs. MEM	202	92	32	76	0.421	10	33	0.303	18	28	0.643	11	26
2023-07-03	Utah Jazz	UTA vs. OKC	200	85	34	89	0.382	9	34	0.265	8	14	0.571	12	35
2023-07-05	Charlotte Hornets	CHA vs. GSW	199	83	32	69	0.464	3	20	0.15	16	19	0.842	7	25
2023-07-05	Los Angeles Lakers	LAL vs. SAS	200	99	34	69	0.493	12	26	0.462	19	25	0.76	8	23



A_Home_opp_ppg_inpaint	A_Away_opp_ppg_inpaint	H_3PA2PA_Ratio	OREB	Spread	Total	H_3PM2PM_Ratio	A_3PA2PA_Ratio	A_3PM2PM_Ratio
45.2	47.3	0.461538461538462	22	-17	197	0.178571428571429	0.825	0.652173913043478
56.7	56.5	0.568181818181818	22	-19	173	0.444444444444444	0.54	0.375
48.9	50.1	0.775	16	6	194	0.36	0.916666666666667	0.736842105263158
48	47.5	0.767441860465116	26	-2	186	0.454545454545455	1.32352941176471	1.14285714285714
50.9	50.5	0.618181818181818	21	-10	180	0.36	0.659574468085106	0.619047619047619
48.9	50.1	0.408163265306122	20	-15	181	0.103448275862069	1	0.583333333333333
56.7	56.5	0.604651162790698	28	-10	208	0.545454545454545	0.433333333333333	0.48

The above figures show the beginning columns and end columns of our 195-column dataset, respectively

The Defensive/Offensive efficiency variables were chosen to better predict the spread and the total for these games as the efficiency on both sides of the ball for these teams can impact the points that they score and allow. We also chose Free Throws Made per 100 baskets to see if teams who shoot better from the line and have a higher frequency of free throws would have an impact on their points scored which would impact our predictions for both spread and total. Points per Game and Points per Game in Paint were obvious choices for us to include as these variables show the trend of how many points a team is putting up per night which can help us greatly in predicting the spread and total. The inclusion of variables Offensive Rebound Percent and Offensive Rebounds per Game was intended for our OREB predictions, but we were also interested to see how these two variables could impact the total and spread as well since teams with better offensive rebounding numbers will likely put more shots up resulting in more points. Steals per defensive play and Steals per Game show the frequency of steals and the average steals a team will collect in a game, respectively. We decided to add these variables in our dataset because we figured a team with more steals would be more likely to prevent their opponents from scoring, thus producing a lower total and altered spread.

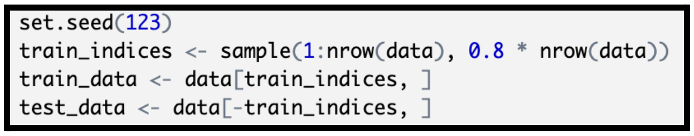
We also created two variables in 3PA/2PA Ratio and 3PM/2PM Ratio to help predict spreads, totals, and offensive rebounds for our desired games. What we hoped this would accomplish is to show what each team is better at between two-point shots and three-point shots as well as show what each team favors when it comes to the two types of shots. In addition, to analyze a potential trend of offensive rebounds and points in the paint made. The calculations for these variables are pretty self-explanatory as they are simply the 3-point attempts or makes divided by the 2-point attempts or makes. In terms of predictions, these two variables were intended to influence the spreads, totals, and offensive rebounds as teams who may prefer or be more successful at shooting three-pointers may average more points per game than teams who don't shoot as many three-pointers.

METHODOLOGY FOR SPREAD

In our project, one of our goals was to construct a reliable predictive model for determining the spread in NBA games. The spread, a critical measure in sports analytics, reflects the expected difference in performance between the Home and Away team (Home points - Away points). Achieving accurate predictions for game spreads requires a nuanced understanding of numerous variables and their interactions. We approached this challenge by utilizing statistical techniques in R.

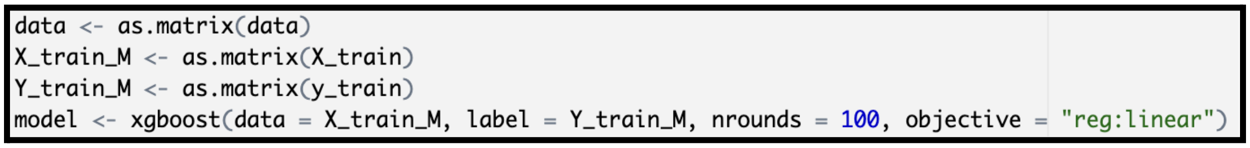
We initiated our modeling process by loading in our final dataset. In order for our dataset to work with XGBoost modeling software we wanted to use, we had to transform all variables to a numeric format. This transformation involved assigning numeric codes to teams, converting dates into separate components (month, day, year, day of the week), and translating percentage values into decimals. We also excluded actual game outcomes since these wouldn't be available for future game predictions and therefore would not work with predicting with our model.

In the development phase of our model, we started by guaranteeing the consistency and reproducibility of our results by setting a fixed random seed. Following this, we split our dataset into 80% for training and 20% for testing. This division was for the evaluation of our model's predictive capabilities on unseen data.



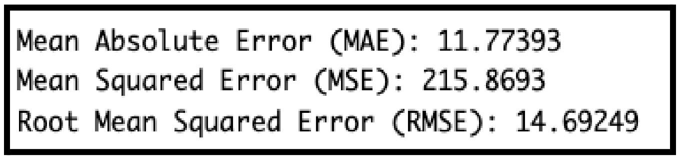
```
set.seed(123)
train_indices <- sample(1:nrow(data), 0.8 * nrow(data))
train_data <- data[train_indices, ]
test_data <- data[-train_indices, ]
```

Before finally creating the model, we converted the dataset into a matrix format, which is the only format of data that is accepted by XGBoost. Once the training data was usable, we initialized and created our initial model.



```
data <- as.matrix(data)
X_train_M <- as.matrix(X_train)
Y_train_M <- as.matrix(y_train)
model <- xgboost(data = X_train_M, label = Y_train_M, nrounds = 100, objective = "reg:linear")
```

To use XGBoost effectively, we configured our model to undergo 100 boosting rounds. Increasing the nrounds parameter can potentially improve the model's performance by allowing it to learn more complex patterns in the data. However, a higher number of rounds also increases the training time and gives the potential for overfitting, which is why we settled at the number 100. We initially implemented a linear regression approach for the XGBoost model, but the XGBoost software told us once running the code that a squared error model would be more effective and it automatically changed the approach. After our model was complete, we tested it on the test set to assess its predictive performance. To measure its accuracy, we calculated its MAE, MSE, and RMSE, which is seen to the right:



```
Mean Absolute Error (MAE): 11.77393
Mean Squared Error (MSE): 215.8693
Root Mean Squared Error (RMSE): 14.69249
```

In the optimization phase of our model, we attempted techniques such as cross-validation, Bayesian optimization, and random tuning. In Bayesian Optimization and random tuning, we encountered several errors that we could not figure out, which led to us abandoning those efforts. In using cross-validation methods, we obtained a model that was slightly worse than our initial one. These issues and results led us to a decision to maintain our initial model configuration with default hyperparameters. This choice was informed by our limited experience with the intricate software tools, reflecting a balance between ambition and the practical learning curve we navigated throughout this process.

In a separate analysis, we also looked into stepwise regression, a method aimed at identifying which specific variables significantly influence the Spread. This model provided insights by selecting each and every variable in our dataset, showing that each statistic we decided to scrape was a valuable one. When analysing the error margin for the stepwise regression, we concluded that it was much higher than the other models we had previously attempted, concluding that it did not lead to the development of a model that outperformed our initial XGBoost model.

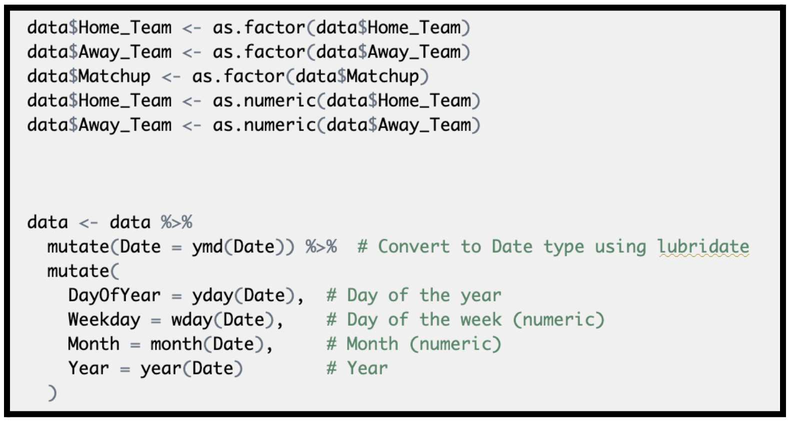
To project the Spreads for upcoming games, we constructed a dataset that directly mirrored the one used during our model's training phase. As we obviously can't update the cumulative statistics as the games we want to predict are completed, we decided to just use the updated statistics for the season, home, away, and last 3 games as of 4/9/2024 to predict each of the 57 games assigned to us. Scraping this data was done by a slightly modified scraper from the one we used before where, in contrast to our previous approach, we iterated through every statistic for each team, rather than iterating through each date and team for each statistic individually. Ensuring the structural and format consistency of this new dataset with our original training dataset was very important as any discrepancies in the dataset structure could lead to the rejection of the input by the model. To achieve this, we verified that column names and their order were precisely aligned with those of the training dataset. Furthermore, we utilized a data merging technique similar to the one in our initial data compilation phase, guaranteeing accurate association of each game with the relevant home and away team statistics.

The final step involved iterating through the prediction dataset, provided by Dr. Mario, to identify the teams involved in each game. We then filtered the recently scraped data to extract the specific statistics for these teams, subsequently appending this information to the Predictions dataset. This ensured that each game was equipped with the latest team performance data.

Upon completing the dataset preparation and ensuring its compatibility with our model, we proceeded to generate our Spread predictions. These predictions were then added to the prediction dataset for finalization.

METHODOLOGY FOR TOTAL

In our project, we aimed to construct a predictive model for estimating the Total points scored in NBA games, utilizing the same approach and methodology that proved successful in predicting game spreads. Total points, a fundamental metric in sports analytics, encapsulates the overall scoring dynamics between the Home and Away teams (Home points + Away points). Achieving precise predictions for total points necessitates a comprehensive analysis of various factors and their interplay. To tackle this challenge, we employed advanced statistical techniques in R.

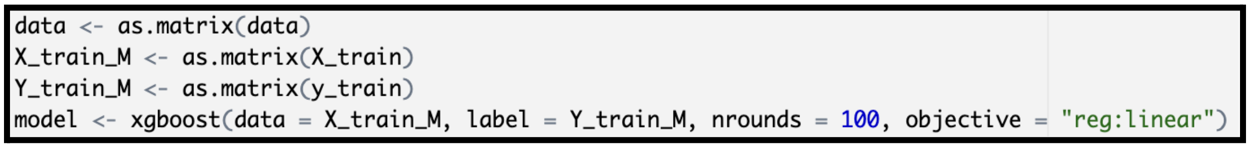


```r
data$Home_Team <- as.factor(data$Home_Team)
data$Away_Team <- as.factor(data$Away_Team)
data$Matchup <- as.factor(data$Matchup)
data$Home_Team <- as.numeric(data$Home_Team)
data$Away_Team <- as.numeric(data$Away_Team)



data <- data %>%
  mutate(Date = ymd(Date)) %>%  # Convert to Date type using lubridate
  mutate(
    DayOfYear = yday(Date),  # Day of the year
    Weekday = wday(Date),    # Day of the week (numeric)
    Month = month(Date),     # Month (numeric)
    Year = year(Date)        # Year
  )
```

Our creation of our model commenced with importing our finalized dataset. We transformed all variables into a numeric format to align with the XGBoost modeling software's requirements. This transformation encompassed assigning numeric codes to teams, breaking down dates into distinct components (month, day, year, day of the week), and converting percentage values into decimals. Additionally, we excluded actual game outcomes from our dataset since they would not be available for future game predictions and would thus not be compatible with our predictive model.

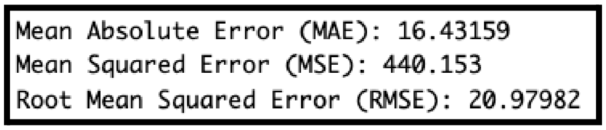
Moving into developing the model, we ensured result consistency and reproducibility by setting a fixed random seed. Subsequently, we divided our dataset into an 80% portion for training and a 20% portion for testing. This partitioning enabled us to evaluate our model's predictive performance on unseen data. Before model creation, we transformed the dataset into a matrix format, as XGBoost only accepts data in this specific format. Once the training data was appropriately structured, we proceeded to initialize and create our initial model.

For effective utilization of XGBoost, we configured our model to undergo 100 boosting rounds. This parameter represents the number of boosting iterations the model undertakes to refine its predictions. While increasing the number of rounds could potentially enhance performance by capturing intricate data patterns, it also introduces risks such as longer training times and potential overfitting. Therefore, we settled on 100 rounds to strike a balance between model complexity and training efficiency. After initially using a linear regression approach, we received feedback from XGBoost software recommending a squared error model for improved effectiveness, prompting us to adjust our approach accordingly.



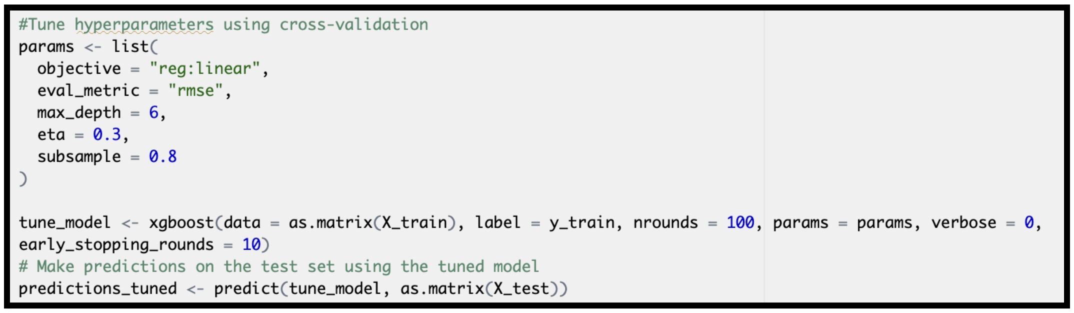
```
data <- as.matrix(data)
X_train_M <- as.matrix(X_train)
Y_train_M <- as.matrix(y_train)
model <- xgboost(data = X_train_M, label = Y_train_M, nrounds = 100, objective = "reg:linear")
```

After completing the model, we conducted t testing on the test set to evaluate its predictive accuracy. We calculated evaluation metrics such as MAE, MSE, and RMSE to gauge the model's performance, which are pictured below:



```
Mean Absolute Error (MAE): 16.43159
Mean Squared Error (MSE): 440.153
Root Mean Squared Error (RMSE): 20.97982
```

During the optimization phase, we explored techniques like cross-validation, Bayesian optimization, and random tuning and again encountered technical challenges leading to



```
#Tune hyperparameters using cross-validation
params <- list(
  objective = "reg:linear",
  eval_metric = "rmse",
  max_depth = 6,
  eta = 0.3,
  subsample = 0.8
)

tune_model <- xgboost(data = as.matrix(X_train), label = y_train, nrounds = 100, params = params, verbose = 0,
early_stopping_rounds = 10)
# Make predictions on the test set using the tuned model
predictions_tuned <- predict(tune_model, as.matrix(X_test))
```

discontinuation of Bayesian and random tuning. Cross-validation also similarly yielded a model slightly inferior to our initial one. Considering these results and challenges, we opted to retain the default hyperparameters for our model configuration.

A model that we also attempted using was the Naive Bayes model. At first, this seemed like a good idea given the model's general success in sorting data into different groups and its overall simplicity. We attempted to see if we could sort games into being high-scoring or low-scoring based on certain factors, turning our continuous challenge of predicting scores into a simpler grouping problem. However, the assumption Naive Bayes makes, that all the factors we're considering (like how good a team is on offense or defense, or their scoring in recent games) work independently from each other, didn't hold up. In sports, these factors are often related. A team's offense can affect its defense, and recent performance can influence a game's outcome in complex ways. Further, Naive Bayes is better at telling categories apart, not predicting specific numbers like total scores. Because of these issues and challenges, this particular model didn't do well in predicting game scores accurately, highlighting the need to choose a model that fits both the kind of data you have and the exact problem you're trying to solve, or in this case, what you are trying to predict.

To predict Total points for upcoming games, we constructed a dataset mirroring the training data format. Utilizing a modified data scraper approach, we gathered updated statistics for the current season, home, away, and last 3 games for each team. As we obviously can't update the cumulative statistics as the games we want to predict are completed, we decided to just use the updated statistics for the season, home, away, and last 3 games as of 4/9/2024 to predict each of the 57 games assigned to us. Ensuring structural and format consistency between this dataset and our original training dataset was crucial to avoid model rejection of input data. A data merging technique similar to our initial phase ensured accurate association of each game with relevant team statistics.

In the final stage, we iterated through the prediction dataset to match teams for each game. We extracted the statistics for these teams from the recently scraped data, appending this information to the Predictions dataset. This preparation and alignment ensured each game was equipped with the cumulative data as of 4/9/24 for the home and away team in each game.

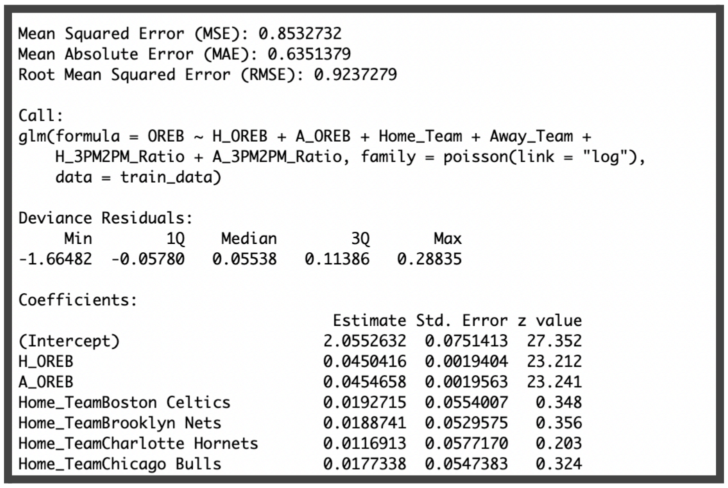
Finally, we leveraged our model to generate predictions for Total points, incorporating these predictions into the final prediction dataset along with the Spread predictions.

METHODOLOGY FOR OREB

In our project, our aim was to develop a strong predictive model for estimating offensive rebounds (OREB) in NBA games. OREB is a significant metric in basketball that will significantly impact a team's offensive performance. In order to achieve our prediction goals, we used many statistical models techniques to understand the relationships between various factors and OREB.

As previously stated, we collected comprehensive data on NBA games over the past year and curated the dataset to ensure consistency. Intuitively looked at mostly offensive variables when selecting variables. Starting with home team, away team, home offensive rebounds, away offensive rebounds, shooting percentage, three pointers, two pointers, and free throws made. In addition, we also derived new variables from our scraped data. For offensive rebounds, we utilized a new metric, our new three points to 2 points made ratio (H_3PM2PM_Ratio, A_3PM2PM_Ratio) when making our model. We found after using the Pearson correlation coefficient with our new variable and OREB, it showed a high correlation. We found that the ratio can be a helpful predictor during offensive rebounds for several reasons. Number one being that teams that excel in 3-point shooting tend to spread the floor more efficiently, creating additional space for offensive rebounds. On the other hand, defending teams may prioritize contesting 3-point shots, leaving them vulnerable to offensive rebounds if they commit too many defenders to perimeter defense. Which can reflect on offensive rebounds and is essential because our OREB predictions accounts for both the home and away team.

Knowing the complexity of our statistic, we knew the discrete numeric variable needed some sort of nonlinear transformation model. We started by trying to run a Gradient Boosting Machine (GBM) model, however, even after optimizing it slightly, our MSE stayed at eleven which would not suffice. After trying to get that number down, we inferred that our data may have been too complex and dense to effectively run our model.

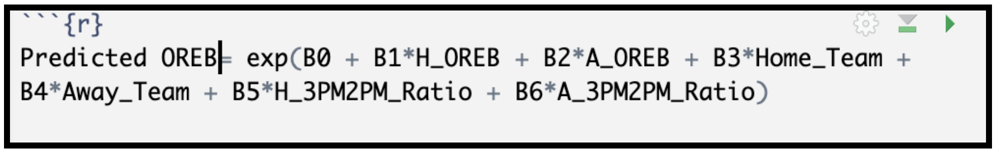


```
Mean Squared Error (MSE): 0.8532732
Mean Absolute Error (MAE): 0.6351379
Root Mean Squared Error (RMSE): 0.9237279

Call:
glm(formula = OREB ~ H_OREB + A_OREB + Home_Team + Away_Team +
    H_3PM2PM_Ratio + A_3PM2PM_Ratio, family = poisson(link = "log"),
    data = train_data)

Deviance Residuals:
     Min        1Q    Median        3Q       Max
-1.66482  -0.05780   0.05538   0.11386   0.28835

Coefficients:
                              Estimate Std. Error z value
(Intercept)                  2.0552632  0.0751413  27.352
H_OREB                       0.0450416  0.0019404  23.212
A_OREB                       0.0454658  0.0019563  23.241
Home_TeamBoston Celtics      0.0192715  0.0554007   0.348
Home_TeamBrooklyn Nets       0.0188741  0.0529575   0.356
Home_TeamCharlotte Hornets   0.0116913  0.0577170   0.203
Home_TeamChicago Bulls       0.0177338  0.0547383   0.324
```

Subsequently, we implemented a basic basic poisson regression model for offensive rebounds by first running a train-test split for randomness. Then, we fitted a poisson regression model by using the ‘glm’ function. At first when we ran this model, it showed us NA values for the MSE, MAE, and RMSE. While this can be normal because these statistics are not normally used in poisson models we still optimized it more to get these values. Following the optimization process, our notebook displayed the following outcomes.

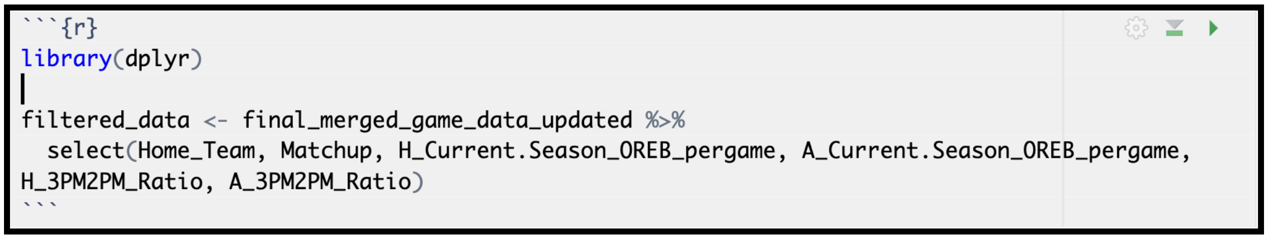
Looking at the notebook summary, we have now obtained our values and, compared to our data we are trying to collect, the numbers are low enough to use and generate good predictions. To double-check, because of the nature of this model, it is also important to examine the null deviance and residual deviance. Our null and residual deviance are recorded as 1227.66 and 34.43, respectively. In our case, the substantial decrease from a null deviance of 1227 to a residual deviance of 34 indicates that the predictors included in your model significantly improve its ability to explain the variation in the response variable (OREB). Therefore, the model seems to fit the data well.

The most challenging part was conducting our predictions. Predictions models rely heavily on unknown data. Our job was to figure out a way to find these unknowns in order to use the Poisson model to make the best predictions. The model equation worked to predict the offensive rebounds using the coefficients obtained from the regression model.



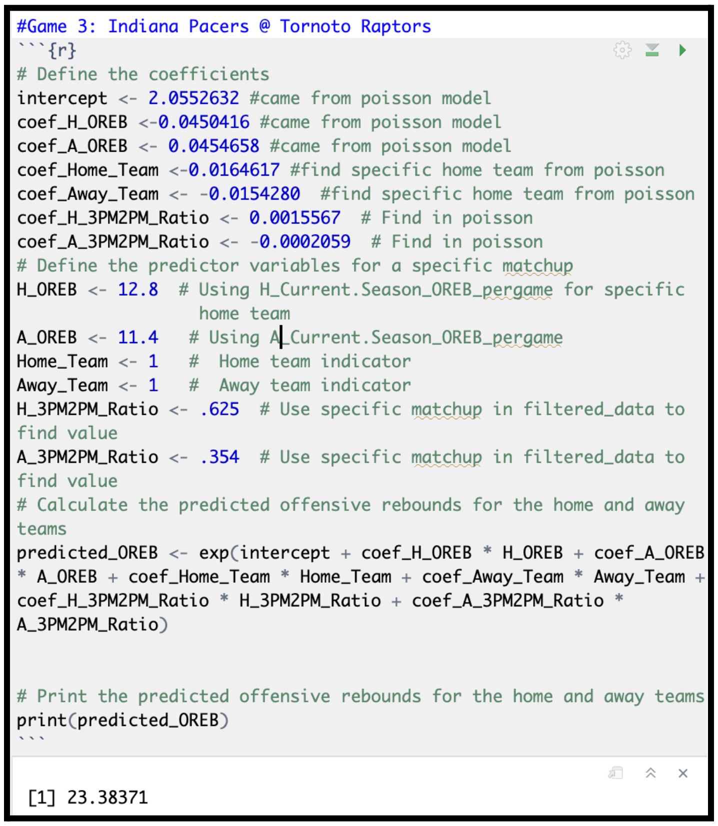
```{r}
Predicted OREB = exp(B0 + B1*H_OREB + B2*A_OREB + B3*Home_Team +
B4*Away_Team + B5*H_3PM2PM_Ratio + B6*A_3PM2PM_Ratio)
```

Next we filtered our data to only contain home team, away team, home OREB, away OREB, and home and away three to two points made ratio.



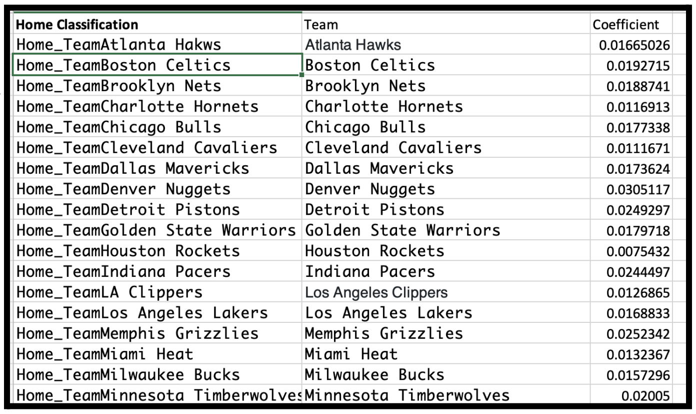
```{r}
library(dplyr)

filtered_data <- final_merged_game_data_updated %>%
  select(Home_Team, Matchup, H_Current.Season_OREB_pergame, A_Current.Season_OREB_pergame,
H_3PM2PM_Ratio, A_3PM2PM_Ratio)
```

Using the intercepts and coefficients from the poisson model and historical data from our dataset, we initially used this equation to make a prediction for each model. The intercept, coef_H_OREB, and coef_A_OREB all come from the Poisson model notebook summary to show on average what home and away teams are calculating. Looking further down, we also utilized coef_Home_Team and coef_Away_Team that we found from looking at the specific coefficients under each team name. In addition, H_OREB and A_OREB has come from our historical data using the current running average of home/away offensive rebounds per game for each team. For our ratios, we looked at specific matchups we were trying to predict in the last season to see the ratio of 3 to 2 pointers in that game. Combining all that data to calculate the expected OREB for both teams gives us a number recorded at the bottom of the image below.



~~~
#Game 3: Indiana Pacers @ Tornoto Raptors
```{r}
# Define the coefficients
intercept <- 2.0552632 #came from poisson model
coef_H_OREB <-0.0450416 #came from poisson model
coef_A_OREB <- 0.0454658 #came from poisson model
coef_Home_Team <-0.0164617 #find specific home team from poisson
coef_Away_Team <- -0.0154280  #find specific home team from poisson
coef_H_3PM2PM_Ratio <- 0.0015567  # Find in poisson
coef_A_3PM2PM_Ratio <- -0.0002059  # Find in poisson
# Define the predictor variables for a specific matchup
H_OREB <- 12.8  # Using H_Current.Season_OREB_pergame for specific
                 home team
A_OREB <- 11.4   # Using A_Current.Season_OREB_pergame
Home_Team <- 1   #  Home team indicator
Away_Team <- 1   #  Away team indicator
H_3PM2PM_Ratio <- .625  # Use specific matchup in filtered_data to
find value
A_3PM2PM_Ratio <- .354  # Use specific matchup in filtered_data to
find value
# Calculate the predicted offensive rebounds for the home and away
teams
predicted_OREB <- exp(intercept + coef_H_OREB * H_OREB + coef_A_OREB
* A_OREB + coef_Home_Team * Home_Team + coef_Away_Team * Away_Team +
coef_H_3PM2PM_Ratio * H_3PM2PM_Ratio + coef_A_3PM2PM_Ratio *
A_3PM2PM_Ratio)


# Print the predicted offensive rebounds for the home and away teams
print(predicted_OREB)
```

 [1] 23.38371
~~~

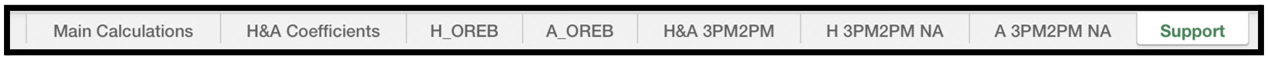
However, we quickly discovered that our model utilized different coefficients pulling from different datasets due to the nature of our variables. One example is seen in the H/A_3PM2PM_Ratio where it pulls from a specific matchup while the H/A_OREB pulls for the specific team. With this challenge, the team pivoted to Excel with a broader base of data manipulation and prediction tools utilizing primarily pivot tables, Vlookups, and sorting functions. Below we will outline the steps we took to come up with our



Home Classification	Team	Coefficient
Home_TeamAtlanta Hakws	Atlanta Hawks	0.01665026
Home_TeamBoston Celtics	Boston Celtics	0.0192715
Home_TeamBrooklyn Nets	Brooklyn Nets	0.0188741
Home_TeamCharlotte Hornets	Charlotte Hornets	0.0116913
Home_TeamChicago Bulls	Chicago Bulls	0.0177338
Home_TeamCleveland Cavaliers	Cleveland Cavaliers	0.0111671
Home_TeamDallas Mavericks	Dallas Mavericks	0.0173624
Home_TeamDenver Nuggets	Denver Nuggets	0.0305117
Home_TeamDetroit Pistons	Detroit Pistons	0.0249297
Home_TeamGolden State Warriors	Golden State Warriors	0.0179718
Home_TeamHouston Rockets	Houston Rockets	0.0075432
Home_TeamIndiana Pacers	Indiana Pacers	0.0244497
Home_TeamLA Clippers	Los Angeles Clippers	0.0126865
Home_TeamLos Angeles Lakers	Los Angeles Lakers	0.0168833
Home_TeamMemphis Grizzlies	Memphis Grizzlies	0.0252342
Home_TeamMiami Heat	Miami Heat	0.0132367
Home_TeamMilwaukee Bucks	Milwaukee Bucks	0.0157296
Home_TeamMinnesota Timberwolves	Minnesota Timberwolves	0.02005

predictions as well as some of the challenges we faced along the process.

First, we exported data that we needed from R's dataset to Excel, filtering out items that were no longer relevant and eventually coming down to coef_Home/Away team, H/A_OREB, and H/A_3PM2PM_Ratio. These were the main variables that changed from each matchup, while the other components of the outlined equation were fixed. See below for the setup of the Excel file, with the Main Calculations sheet being where the data was compiled and calculated for our final predictions for OREB. The other sheets act as support to help set up the data in a way that utilizing the VLOOKUP function, specific information can be pulled using the home team/away team name. We preferred this method of utilizing Excel since it was less time intensive, compiling 57 different calculations seen above, and manually pulling each number that had to be changed. Furthermore, the accuracy of the data can be protected by having formulas, eliminating human error in the process.

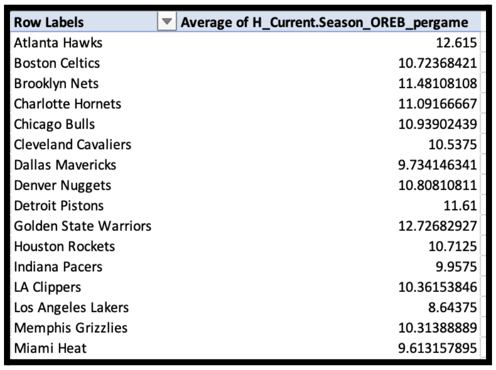


Specifically regarding the VLOOKUP function, H&A coefficients are obtained by matching the pair of teams and their home/away coefficient. The data is pulled from the notebook summary shown above. The same table below is also constructed for away teams.

Then, using the following as an example, we see that the VLOOKUP searches for the team name in cell C9 of the main calculation sheet, and pulls the value one column to the right, making sure to match exactly as the team names are.

=VLOOKUP(C9,'H&ACoefficients'!$B$2:$C$31,2,FALSE)

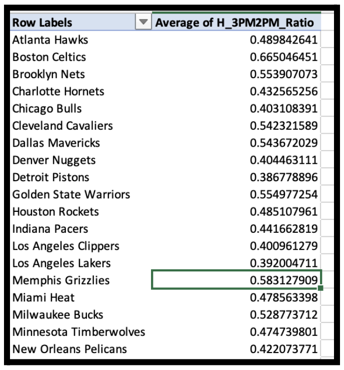
For H/A_OREB, our team pulled data from the filtered_data reference above (more specifically columns 3 and 4), and we utilized the pivot table function to calculate the average of a specific home/away team's offensive rebound in a game and used the VLOOKUP function as detailed above to pull in the data based on each specific team and their home/away status. See the image associated for a snapshot of an average of a team's offensive rebounds in a game. This helps to account that some teams have an advantage in rebounding at home or disadvantage away.



Row Labels	Average of H_Current.Season_OREB_pergame
Atlanta Hawks	12.615
Boston Celtics	10.72368421
Brooklyn Nets	11.48108108
Charlotte Hornets	11.09166667
Chicago Bulls	10.93902439
Cleveland Cavaliers	10.5375
Dallas Mavericks	9.734146341
Denver Nuggets	10.80810811
Detroit Pistons	11.61
Golden State Warriors	12.72682927
Houston Rockets	10.7125
Indiana Pacers	9.9575
LA Clippers	10.36153846
Los Angeles Lakers	8.64375
Memphis Grizzlies	10.31388889
Miami Heat	9.613157895

Lastly, H/A_3PM2PM_Ratio helps to account for a difference in a team's playstyle, specifically shooting more 3 points than 2 points, resulting in a much higher opportunity for OREBs throughout the game. While incorporating the data to account for the specific matchups, we realized that we did not have specific matchup data for certain games. To combat this issue, we first made sure to pull all specific matchup data that matched with our prediction matchups. Then, we were left with ~20 matchups without specific historical matchup data.

Our solution was to first separate the filter_data set out such that we have the Home/Away team in one column and H/A_3PM2PM _Ratio in the other for all the specific matchup data we have. Then we utilized pivot tables to calculate the average of a specific team's 3PM2PM ratio when they were H/A. See the table associated for reference. These altered data would then be individually fitted to the specific matchups without historical data based on the home/away team featured. While these solutions do help plug some gaps in the predictions, the invariable difference in consistency across the prediction might have some effect on the eventual outcome.



Row Labels	Average of H_3PM2PM_Ratio
Atlanta Hawks	0.489842641
Boston Celtics	0.665046451
Brooklyn Nets	0.553907073
Charlotte Hornets	0.432565256
Chicago Bulls	0.403108391
Cleveland Cavaliers	0.542321589
Dallas Mavericks	0.543672029
Denver Nuggets	0.404463111
Detroit Pistons	0.386778896
Golden State Warriors	0.554977254
Houston Rockets	0.485107961
Indiana Pacers	0.441662819
Los Angeles Clippers	0.400961279
Los Angeles Lakers	0.392004711
Memphis Grizzlies	0.583127909
Miami Heat	0.478563398
Milwaukee Bucks	0.528773712
Minnesota Timberwolves	0.474739801
New Orleans Pelicans	0.422073771