NBA MVP Predictor

Praveen Manimaran

12/10/2022

Contents

Introduction

Goal of Project

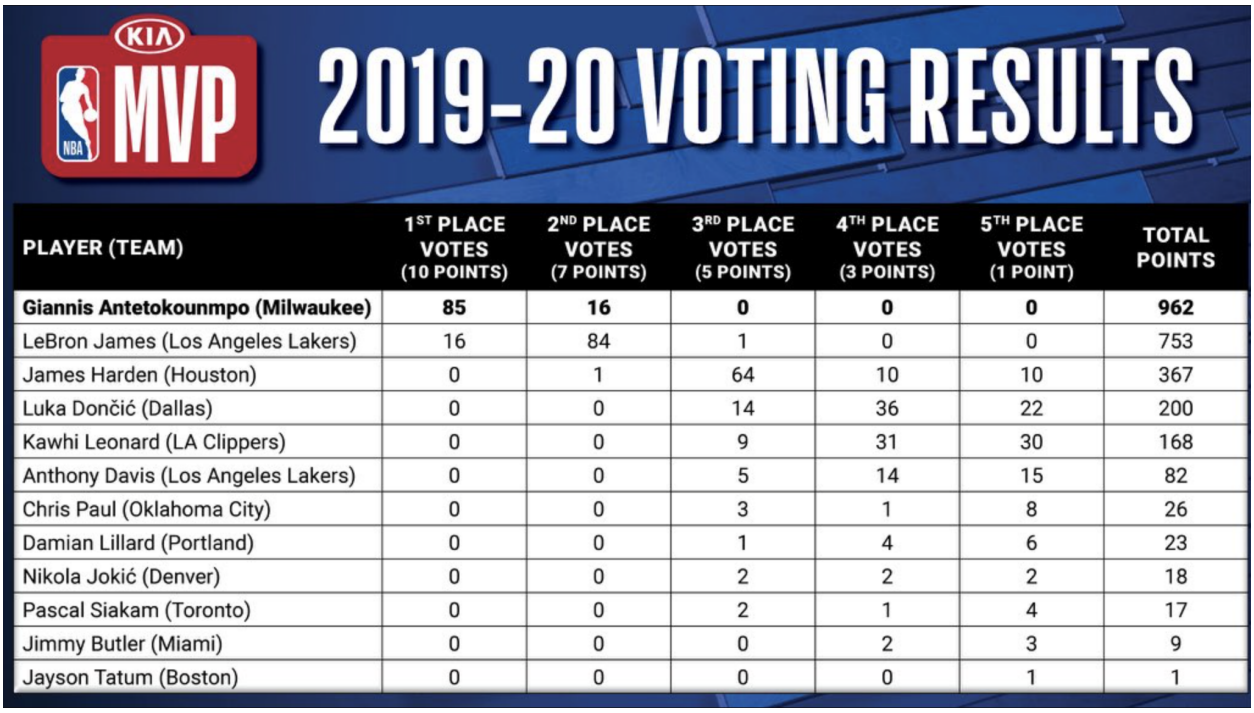
The goal of this project is to build a machine learning model that can predict the NBA's Most Valuable Player (MVP) of any NBA season.

History of the NBA MVP

The NBA's Most Valuable Player award is given annually to to the best performing player of the regular season since the start of the 1955–56 season. At the start of the first NBA season, the MVP award would be selected by NBA players through voting. This was until the 1979-80 season, in which the award is now decided by various of sportswriters/broadcasters in the United States. At the start of the 2010 season, fans were allowed to cast their vote for one ballot.

Award Voting Process

Each member from the list of voters casts a vote for ranking their top 5 candidates in order. A 1st place vote is worth 10 pts, a 2nd place vote is worth 7 pts, a 3rd place vote is worth 5 pts, a 4th place is worth 3 pts, and the 5th place vote is worth 1 pt. A ballot conducted through online voting from fans is also included. The player who has the highest point total wins the MVP.



PLAYER (TEAM)	1ST PLACE VOTES (10 POINTS)	2ND PLACE VOTES (7 POINTS)	3RD PLACE VOTES (5 POINTS)	4TH PLACE VOTES (3 POINTS)	5TH PLACE VOTES (1 POINT)	TOTAL POINTS
Giannis Antetokounmpo (Milwaukee)	**85**	**16**	**0**	**0**	**0**	**962**
LeBron James (Los Angeles Lakers)	16	84	1	0	0	753
James Harden (Houston)	0	1	64	10	10	367
Luka Dončić (Dallas)	0	0	14	36	22	200
Kawhi Leonard (LA Clippers)	0	0	9	31	30	168
Anthony Davis (Los Angeles Lakers)	0	0	5	14	15	82
Chris Paul (Oklahoma City)	0	0	3	1	8	26
Damian Lillard (Portland)	0	0	1	4	6	23
Nikola Jokić (Denver)	0	0	2	2	2	18
Pascal Siakam (Toronto)	0	0	2	1	4	17
Jimmy Butler (Miami)	0	0	0	2	3	9
Jayson Tatum (Boston)	0	0	0	0	1	1

Why Would This Model Be Useful?

This model could help NBA fans understand how the MVP of the league is actually chosen and whether or not there is bias involved when sportswriters and broadcasters are included in the voting process. This model can give insight to which statistics are the most important in determining who the most valuable player is and if there is a pattern among past players who were awarded this honor.

A lot of NBA fans believe that the MVP is simply based on whoever has the best stats, specifically Points, Rebounds, and Assists. However, this has been proven to not be the case as there are underlying factors involved in the voting process with the criteria for MVP changing every year. Hopefully, this model can help us understand the main reasons for selecting the MVP and give us a better understanding of the statistical requirements (criteria) to get the award.

Project Outline

Let us now talk about how we will approach this problem. We will use several features to help predict the mvp_share metric which will then be used to predict the MVP.

The mvp_share metric is calculated by this simple formula: MVP Share = (MVP points for player) / (Total MVP points)

				Voting						
Rank	**Player**	**Age**	**Tm**	**First**	**Pts Won**	**Pts Max**	**Share**	**G**	**MP**	**PTS**
1	Nikola Jokić	26	DEN	65.0	875.0	1000	0.875	74	33.5	27.1
2	Joel Embiid	27	PHI	26.0	706.0	1000	0.706	68	33.8	30.6
3	Giannis Antetokounmpo	27	MIL	9.0	595.0	1000	0.595	67	32.9	29.9

The player with the largest MVP Share metric for a particular year is the winner of the award.

I will be performing a training/test split on the data, making a recipe, and setting folds for the 5-fold cross validation. I will be using 4 models: Linear Regression, Ridge Regression, Random Forest, Boosted Model on the training data. Based of the RMSE(Residual Mean Squared Error) value of each model, I will pick the model with the lowest RMSE and then fit that model to our testing data to see how well the model performs.

What is RMSE?

RMSE is essentially the standard deviation of the residuals (prediction errors). Residuals are a measure of how far from the regression line data points are and the RMSE tells us how spread out the residuals are. The lower the RMSE, the better because then you know your model did a good job predicting the outcome variable as the errors are low.

Data and Packages

Data Source

This project uses a data scrape of the yearly NBA MVP Awards Voting from https://www.basketball-reference.com/, which records basketball stats and history statistics, scores, and history for the NBA, ABA, WNBA, and top European competition. I was able to download the yearly data from the start of the NBA in the 1955-1956 season to the last NBA season 2021-2022.

Loading Packages

```
library(tidymodels)
library(tidyverse)
library(ggthemes)
library(ISLR)
library(ggplot2)
library(discrim)
library(poissonreg)
library(corrr)
library(corrplot)
library(janitor)
library(glmnet)
library(rpart.plot)
library(vip)
library(randomForest)
library(xgboost)
library(ranger)
tidymodels_prefer()
```

Reading in the data file

```
mvpList <-read.csv('/Users/praveenmanimaran/Desktop/PSTAT131_FinalProject/data/NBA_MVP_TOP5.csv')

mvpList <- mvpList %>%
```

```
  clean_names() #clean names

#creating a copy of data set to add column:pra(total avg of points rebounds assists)
#this copied dataset will be used for EDA to see if this variable can be useful to test my prediction
mvpList2 <- mvpList
mvpList2$pra <- c(mvpList$pts + mvpList$ast + mvpList$trb)

set.seed(2324) #set seed to reproduce results
```

Splitting the data

The data was split in a 70% training, 30% testing split. Stratified sampling was used with the strata being mvp_share, which is going to be our response variable to help determine the MVP. I am using stratified sampling because it helps me ensure that specific subgroups are present in the sample.

```
mvp_split <- initial_split(mvpList, prop = 0.70, strata = "mvp_share")
mvp_train <- training(mvp_split)
mvp_test <- testing(mvp_split)
```

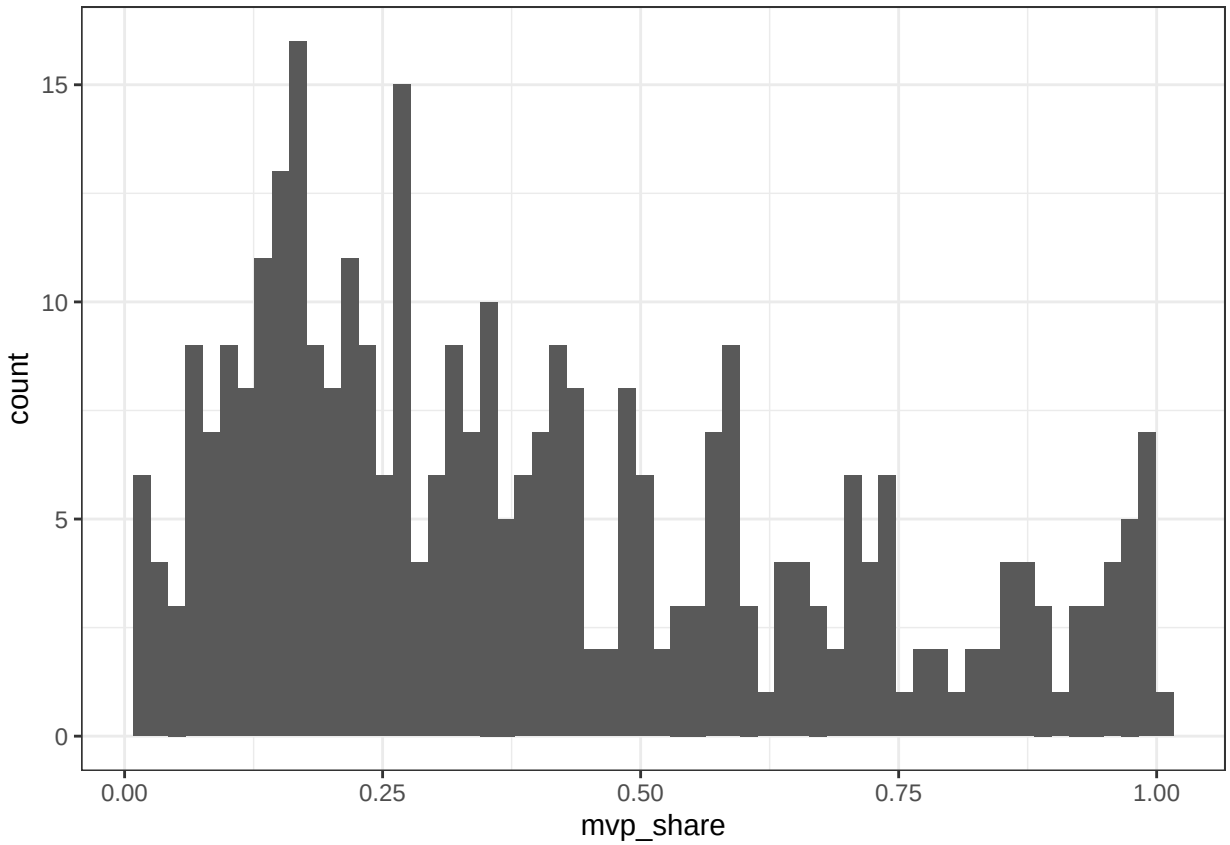
Exploratory Data Analysis

I will be performing Exploratory Data Analysis to see how the distribution of my data set is and see which predictors are the most important and which ones are the least important. I also want to know if any of the predictors have a relationship with one another. Like most NBA fans, I believe that the most important metric used to determine the MVP is Points, Rebounds, Assists and Win Shares. I will be testing these predictions below!

I first want to understand what the distribution of our outcome variable looks like and see if it is skewed.

Histogram of mvp_share to Understand Distribution

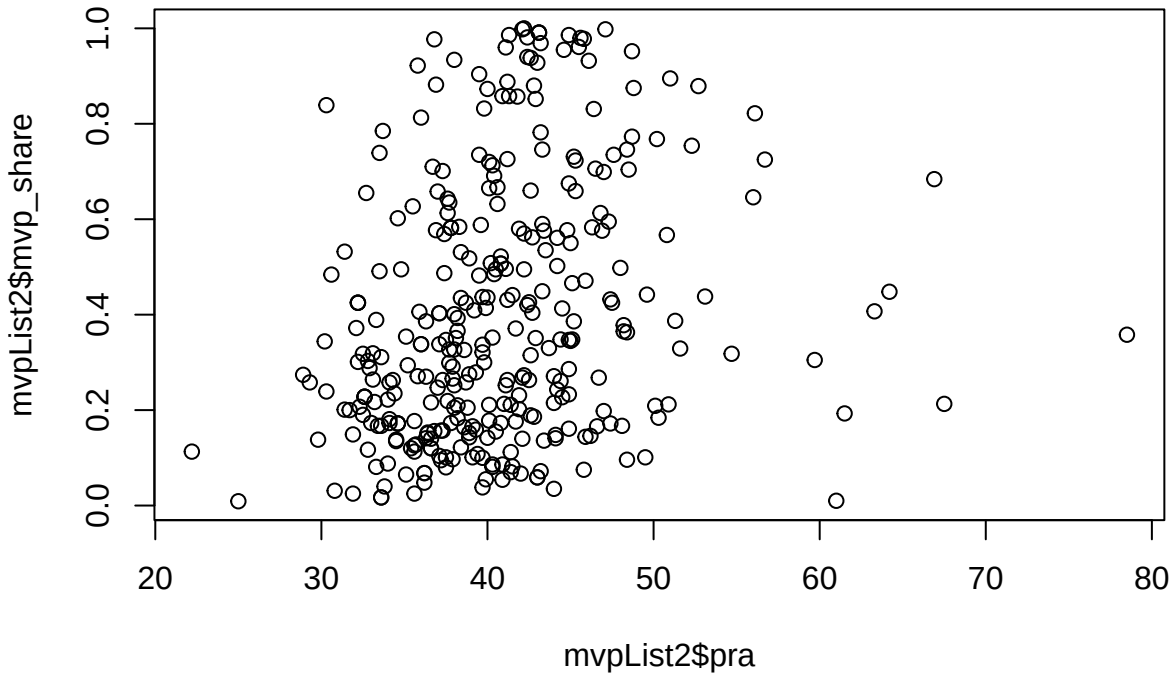
```
mvpList %>%
  ggplot(aes(x = mvp_share)) +
  geom_histogram(bins = 60) +
  theme_bw()
```



I expected that the distribution of mvp_share would resemble a bell curve but this graph makes sense because there is usually only 1-2 players that are truly in contention of winning the award so majority of the votes would go to them. This would mean that the rest of the 3 players tend to get much lower votes in comparison and would have a lower mvp_share. The graph appears to have peaks around 0.15 and 0.30.

Let us check our predictions to see if points, rebounds, assists are the main predictors of mvp_share

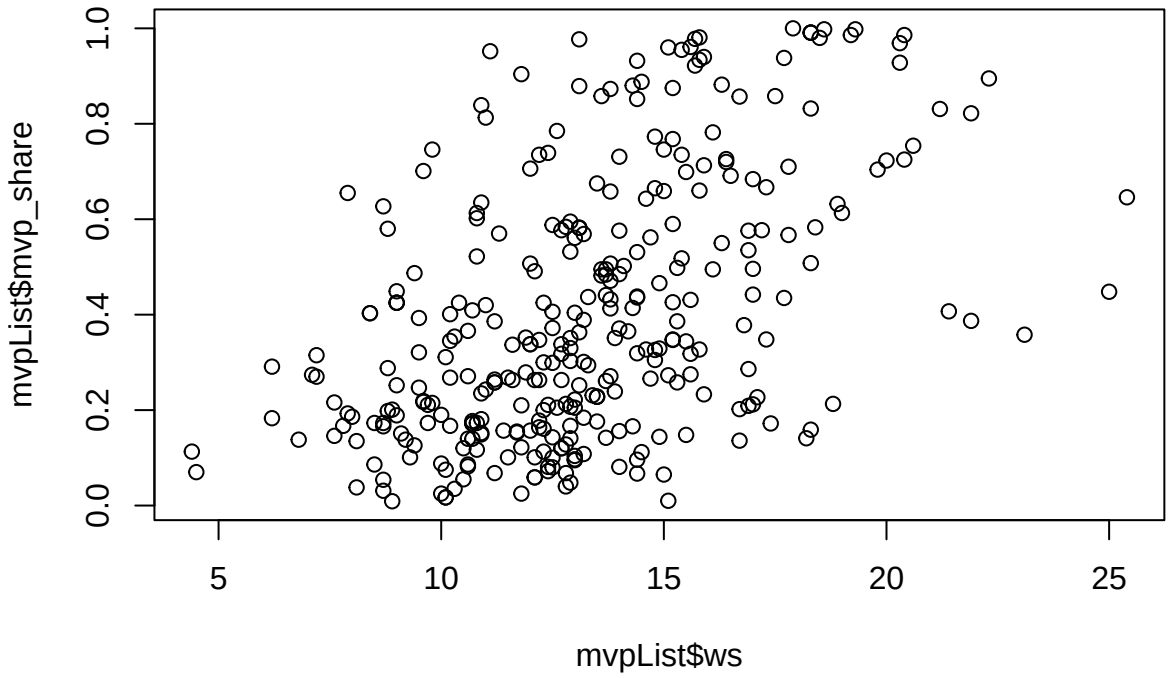
```
plot(mvpList2$pra, mvpList2$mvp_share)
```



My predictions were wrong, it appears that the average total of Points, Rebounds, Assists has little impact on the mvp_share.

Let us see if my prediction of Win Shares has an impact on mvp_share.

```
plot(mvpList$ws, mvpList$mvp_share)
```

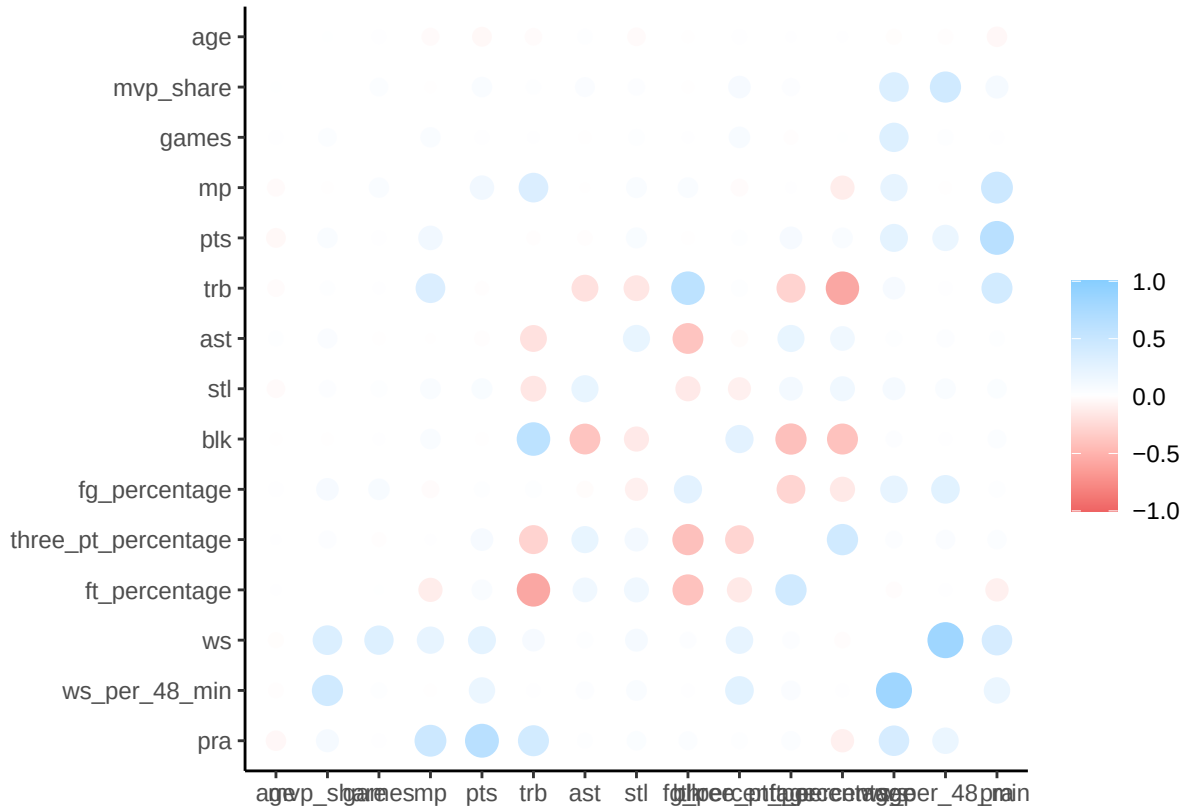


This prediction was right! It appears that win shares has a strong positive correlation with mvp_share.

We can now check the relationship between all variables(deselecting the non important ones) using a correlation matrix

Correlation Plot Between Variables

```
#deselecting all variables that would not be needed
cor_mvp <- mvpList2%>%
  select(-player, -team, -year, -first, -rank, -pts_won, -pts_max) %>%
  correlate()
rplot(cor_mvp)
```



It appears that mvp_share has a strong positive correlation with only win shares and win shares per 48 min. Freethrow percentage also has a strong negative correlation with total rebounds which makes sense because taller players tend to grab more rebounds and be worse shooters from further distances. Blocks and total rebounds have a positive correlation because taller players can easily get more rebounds and can block more shots. The number of games has a slight positive correlation with win shares because the more games you play, the more wins you can contribute for(definition of win shares).

Exploring Missing Data

```
summary(mvpList)
```

```
##       year           rank      player              age
##  Min.   :1956   Min.   :1   Length:335         Min.   :20.00
##  1st Qu.:1972   1st Qu.:2   Class :character   1st Qu.:25.00
##  Median :1989   Median :3   Mode  :character   Median :27.00
##  Mean   :1989   Mean   :3                      Mean   :27.23
##  3rd Qu.:2006   3rd Qu.:4                      3rd Qu.:29.00
##  Max.   :2022   Max.   :5                      Max.   :38.00
##
##      team               first           pts_won          pts_max
##  Length:335         Min.   :  0.00   Min.   :   1.0   Min.   :  80.0
##  Class :character   1st Qu.:  1.00   1st Qu.: 119.0   1st Qu.: 540.0
##  Mode  :character   Median :  7.00   Median : 286.0   Median : 925.0
##                     Mean   : 22.96   Mean   : 373.9   Mean   : 851.7
##                     3rd Qu.: 30.75   3rd Qu.: 549.5   3rd Qu.:1210.0
```

```
##                     Max.   :147.00   Max.   :1310.0   Max.   :1310.0
##
##    mvp_share          games             mp             pts
##  Min.   :0.0090   Min.   :46.00   Min.   :30.40   Min.   : 9.90
##  1st Qu.:0.1730   1st Qu.:72.00   1st Qu.:36.20   1st Qu.:22.00
##  Median :0.3380   Median :78.00   Median :38.10   Median :25.70
##  Mean   :0.4016   Mean   :75.62   Mean   :38.34   Mean   :25.39
##  3rd Qu.:0.5820   3rd Qu.:81.00   3rd Qu.:40.30   3rd Qu.:28.55
##  Max.   :1.0000   Max.   :82.00   Max.   :48.50   Max.   :50.40
##
##       trb              ast              stl             blk
##  Min.   : 2.50   Min.   : 1.300   Min.   :0.500   Min.   :0.100
##  1st Qu.: 6.20   1st Qu.: 3.100   1st Qu.:1.100   1st Qu.:0.500
##  Median : 9.50   Median : 4.600   Median :1.400   Median :0.900
##  Mean   :10.18   Mean   : 5.147   Mean   :1.489   Mean   :1.309
##  3rd Qu.:12.70   3rd Qu.: 6.600   3rd Qu.:1.800   3rd Qu.:2.100
##  Max.   :27.20   Max.   :13.100   Max.   :3.200   Max.   :4.500
##                                   NA's   :90      NA's   :90
##  fg_percentage    three_pt_percentage ft_percentage          ws
##  Min.   :0.3510   Min.   :0.0000      Min.   :0.3800   Min.   : 4.40
##  1st Qu.:0.4575   1st Qu.:0.1980      1st Qu.:0.7320   1st Qu.:10.90
##  Median :0.4940   Median :0.3120      Median :0.7830   Median :13.00
##  Mean   :0.4941   Mean   :0.2664      Mean   :0.7706   Mean   :13.27
##  3rd Qu.:0.5260   3rd Qu.:0.3660      3rd Qu.:0.8415   3rd Qu.:15.20
##  Max.   :0.7270   Max.   :0.4700      Max.   :0.9340   Max.   :25.40
##                   NA's   :124
##  ws_per_48_min
##  Min.   :0.088
##  1st Qu.:0.190
##  Median :0.218
##  Mean   :0.219
##  3rd Qu.:0.245
##  Max.   :0.340
##
```

From the summary() we can see that three columns have missing values: three_pt_percentage, blk, stl.

Setting up Models

Cross-validation is essentially a resampling tool that uses different portions of the data to train/test a model on different iterations(folds). I decided to run cross fold validation on the 4 different models (linear regression, ridge regression, random forest, boosted model).

Creating the Folds

Folding the training data set into 5 folds.

```
mvp_folds <- vfold_cv(mvp_train, v = 5)
```

Building the Recipe

I had originally thought that there was no missing data when taking the data from basketball-reference, but I soon found out that blocks and steal were not recorded until 1973 and 3 point percentage was not recorded until 1979. I needed to adjust my recipe to account for this missing data using step_impute_linear().

I did not want to use first, pts_won, and pts_max as predictors because they are used to calculate mvp_share and I am trying to see how the other predictors can predict mvp_share.

```
#Creating our recipe and using all predictors except player, team, rank, pts_won, and pts_max
mvp_recipe <- recipe(mvp_share ~ age + ast + fg_percentage + ft_percentage + games + mp + pts
                       + trb + ws + ws_per_48_min + blk + stl + three_pt_percentage , data = mvp_train)
                     step_impute_linear(stl, impute_with = imp_vars(age, fg_percentage, ft_percentage, g
                     step_impute_linear(blk, impute_with = imp_vars(age, fg_percentage, ft_percentage, g
                     step_impute_linear(three_pt_percentage, impute_with = imp_vars(age, fg_percentage,
                     step_center(all_predictors()) %>% #center and scale all predictors
                     step_scale(all_predictors())
```

Building the Models

Linear Regression

```
#Using lm model engine
lm_model <- linear_reg() %>%
set_engine("lm")

#setting up workflow
lm_wflow <- workflow() %>%
  add_model(lm_model) %>%
  add_recipe(mvp_recipe)

#fitting to folds
mvp_lm_fit <-fit_resamples(lm_wflow, resamples = mvp_folds)
linreg_rmse <- collect_metrics(mvp_lm_fit) %>%slice(1) #getting rmse using collect_metrics()

linreg_rmse
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.236     5  0.0129 Preprocessor1_Model1
```

```
#RMSE: 0.2364404
```

Ridge Regression

```
#using glmnet model engine and setting mode to regression
ridge_spec <-
  linear_reg(penalty = tune(), mixture = 0) %>% #tells tune_grid() that the penalty parameter should be
```

```
  set_mode("regression") %>%
  set_engine("glmnet")

#setting up workflow
ridge_wflow <- workflow() %>%
  add_recipe(mvp_recipe) %>%
  add_model(ridge_spec)

#setting up penalty grid
# We use the penalty() function from the dials package to denote the parameter and set the range of the
penalty_grid <- grid_regular(penalty(range = c(-5, 5)), levels = 50)

#hyper parameter tuning
tune_res1 <- tune_grid(
  ridge_wflow,
  resamples = mvp_folds,
  grid = penalty_grid
)


best_penalty <- select_best(tune_res1, metric = "rmse")

# This value of penalty can now be used with finalize_workflow() to update/finalize the recipe by repla

#Now, this best model should be fit again, this time using the whole training data set.

ridge_final <- finalize_workflow(ridge_wflow, best_penalty)
mvp_ridge_fit <- fit_resamples(ridge_final, resamples = mvp_folds)

ridge_rmse<-collect_metrics(mvp_ridge_fit) %>% #finding out rmse of our model
slice(1)
ridge_rmse
```

```
## # A tibble: 1 x 6
##   .metric .estimator  mean     n std_err .config
##   <chr>   <chr>      <dbl> <int>   <dbl> <chr>
## 1 rmse    standard   0.233     5  0.0122 Preprocessor1_Model1
```

```
#RMSE: 0.231441
```

Random Forest

```
#using ranger engine and setting mode to regression
rf_model <-
  rand_forest(min_n = tune(), mtry = tune(), trees = tune()) %>%
  set_mode("regression")%>%
  set_engine("ranger")

#setting up workflow
rf_workflow <- workflow() %>%
  add_model(rf_model) %>%
```

```
  add_recipe(mvp_recipe)


param_grid <- grid_regular(mtry(range = c(1, 13)), trees(range = c(200, 1000)), min_n(range = c(1, 20))
```

We want to perform hyperparameter tuning for the random forest model

```
tune_res <- rf_workflow %>%
  tune_grid(resamples = mvp_folds, grid = param_grid)

write_rds(tune_res, file = "nba_mvp_random_forest.rds") #writing results to file so it doesn't take an
```

```
tune_res2 <- read_rds("nba_mvp_random_forest.rds") #easily read from file
#autoplot(tune_res2, metric = "rmse")
```

```
#using show_best()
rf_rmse<-show_best(tune_res2, metric = "rmse") %>% select(-.estimator, -.config) %>%
  slice(1) #mean is 0.2348788

rf_rmse_mean<- 0.2348788

#RMSE: 0.2348778
```

Using the show_best() function, the smallest mean is 0.2348788, with mtry = 4 and min_n = 3.

Boosted Model

I will now fit a boosted tree model to my data set. The xgboost package helps gives a good implementation of boosted trees. It has many parameters to tune and we know that setting trees too high can lead to overfitting. We will try to set a range between 10 and 2000.

```
#using xgboost engine and setting mode to regression
boost_spec <- boost_tree() %>%
  set_engine("xgboost") %>%
  set_mode("regression")

#setting up workflow
boost_wf <- workflow() %>%
  add_recipe(mvp_recipe) %>%
  add_model(boost_spec %>% set_args(trees = tune())) #tuning trees

param_grid2 <- grid_regular(trees(range = c(10, 2000)), levels = 10) #using 10 levels
```

```
tune_res3 <- tune_grid(boost_wf, resamples = mvp_folds, grid = param_grid2)

write_rds(tune_res3, file = "nba_mvp_boost_tree.rds") #writing results to file so it doesn't take long
```

```
tune_res3 <- read_rds("nba_mvp_boost_tree.rds") #reading from file

#autoplot(tune_res3, metric = "rmse")
```

```
boosted_rmse<-show_best(tune_res3, metric = "rmse") %>% select(-.estimator, -.config) %>%
  slice(1)

boosted_rmse$mean #selecting lowest RMSE
```

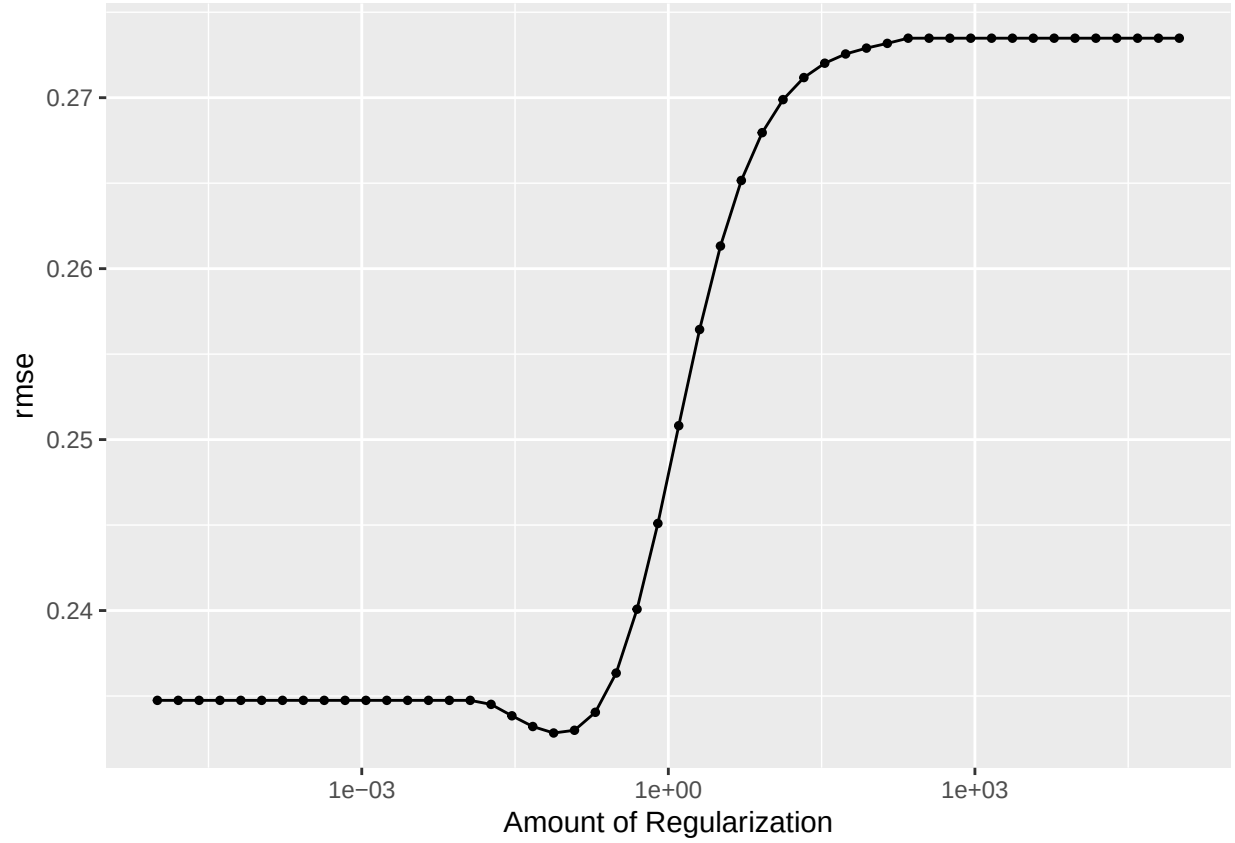
```
## [1] 0.2443362
```

```
#RMSE:0.2443362
```

Autoplots of Models

Autoplot of RMSE of Ridge Regression Model

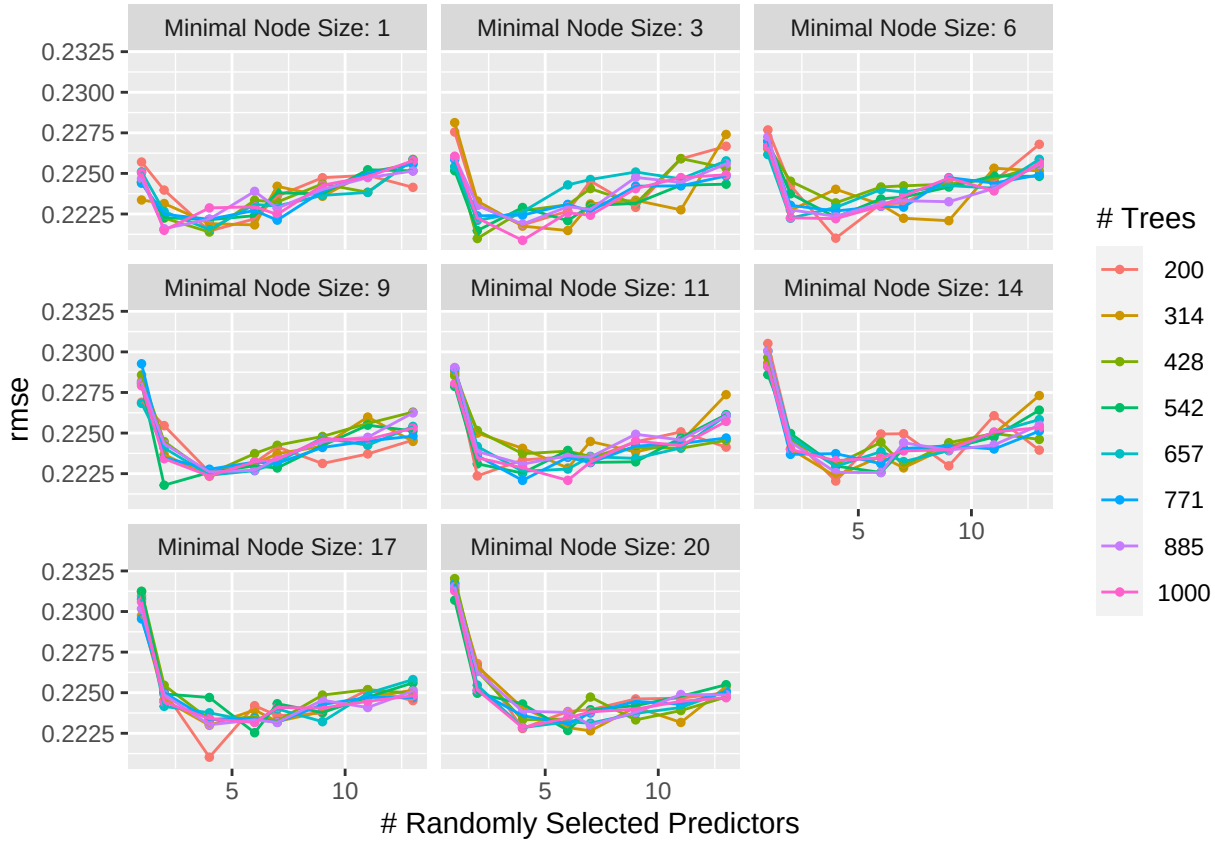
```
autoplot(tune_res1, metric = "rmse")
```



From the autoplot we can see that the amount of regularization affects the performance metrics. Note how there are areas where the amount of regularization doesn't have any meaningful influence on the coefficient estimates. We can see that that after a certain amount of regularization the rmse dips slightly and quickly rises.

Autoplot of RMSE of Random Forest Model

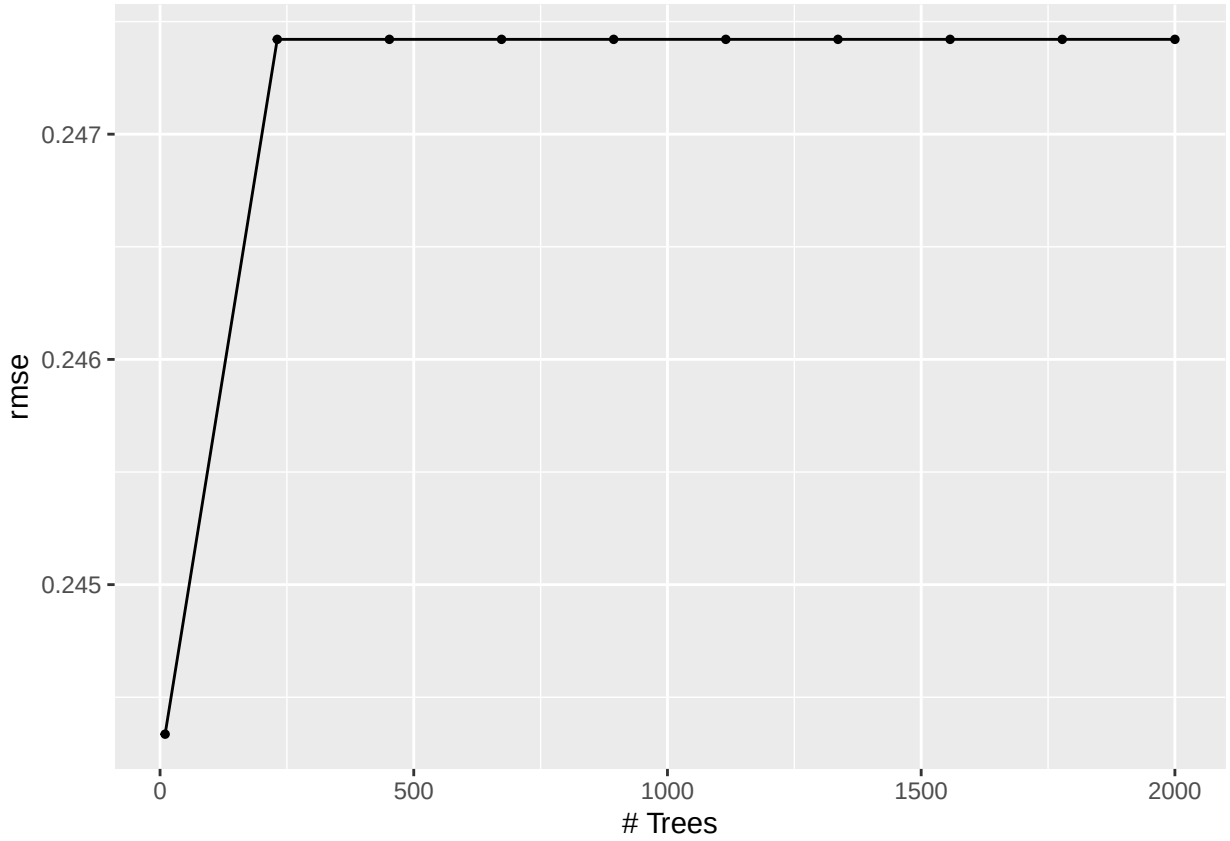
```
autoplot(tune_res2, metric = "rmse")
```



From the autoplot, the model varies slightly by number of trees so number of trees doesn't affect the model. Very small values of min_n seem to start at lower rmse values. Mtry looks to be the best around like 4-6 predictors. The model appears to do worse at lower and higher values of mtry.

Autoplot of RMSE of Boosted Model

```
autoplot(tune_res3, metric = "rmse")
```



The RMSE peaks around ~240 trees.

Results from the Best Model

Comparing RMSEs of All Models

```
rmse_values <- c(linreg_rmse$mean,   ridge_rmse$mean , rf_rmse_mean, boosted_rmse$mean) #all of the RMS

models <- c("Linear Regression", "Ridge Regression", "Random Forest", "Boosted Trees") #creating matchi

results <- tibble(rmse_values = rmse_values, models = models) #creating table

results %>%
  arrange(rmse_values) #arrange by ascending
```

```
## # A tibble: 4 x 2
##   rmse_values models
##         <dbl> <chr>
## 1       0.233 Ridge Regression
## 2       0.235 Random Forest
## 3       0.236 Linear Regression
## 4       0.244 Boosted Trees
```

With an RMSE value of 0.23, it appears that the Ridge Regression model performed the best. All of the models appeared to have similar RMSEs so it is not too big of a difference. It is important to note that

RMSE of 0.23 might be slightly large since our outcome variable only ranges from 0 to 1 a and that could be considered a big difference. Essentially the meaning of a good RMSE depends on the values that the data takes!

Using Model on Testing Data

```
mvp_ridge_fit2 <- fit(ridge_final, data = mvp_train)
final_tibble <- augment(mvp_ridge_fit2, new_data = mvp_test) #fitting model to testing data
comparecol <- bind_cols(final_tibble$player, final_tibble$mvp_share, final_tibble$.pred) #binding colum
colnames(comparecol) <- c("Player", "Actual MVP_Share", ".pred")
comparecol
```

```
## # A tibble: 102 x 3
##    Player            `Actual MVP_Share` .pred
##    <chr>                          <dbl> <dbl>
##  1 Joel Embiid                    0.706 0.483
##  2 Devin Booker                   0.216 0.219
##  3 Joel Embiid                    0.58  0.447
##  4 Luka Doncic                    0.198 0.415
##  5 Paul George                    0.352 0.313
##  6 Nikola Jokic                   0.21  0.491
##  7 Stephen Curry                  0.173 0.322
##  8 LeBron James                   0.731 0.491
##  9 Russell Westbrook              0.879 0.500
## 10 James Harden                   0.746 0.556
## # ... with 92 more rows
```

Finding out well how our model did on our testing set

```
final_tibble %>% rmse(truth = mvp_share, estimate = .pred)
```

```
## # A tibble: 1 x 3
##   .metric .estimator .estimate
##   <chr>   <chr>          <dbl>
## 1 rmse    standard       0.225
```

My model returned an RMSE value of 0.225 on my testing data, which is lower than the training data. Although it is very rare to have a lower test RMSE than the training RMSE, it does mean that my model did a good job not overfitting to the training data.

Remember! The meaning of a good RMSE depends on the values that the data takes so it might not done too great a job of predicting the mvp_share accurately as the data values only lie from 0 to 1. This could indicate that there are other factors that I might not have included in my model that could affect the mvp_share.

Further Exploration for Fun

Predicted MVPs vs Real MVPs

I was curious to see how my model does in terms of accurately predicting the MVP of each year and wanted to compare it to the list of players who have actually won in previous years. I will apply my best model(Ridge Regression) to the entire data set just to see how well it does out of curiosity.

NOTE: This might not be good practice because the model was trained on some values and not trained on others !!

Reading in Data file with list of Actual NBA MVPs by Year

```
actual_MVPs <-read.csv('/Users/praveenmanimaran/Desktop/PSTAT131_FinalProject/data/NBA_MVP_WINNERS.csv')
```

Creating a table with labels(correct, wrong) to compare Predicted MVPs and Actual MVP winners

```
mvp_ridge_fit2 <- fit(ridge_final, data = mvp_train)
final_tibble2 <- augment(mvp_ridge_fit2, new_data = mvpList) #fit model to entire data set to check acc

mvp_share_comparison <- bind_cols(final_tibble2$year, final_tibble2$player, final_tibble2$.pred) #adds

colnames(mvp_share_comparison) <- c("Year", "Player", "Predicted_MVP_Share") #sets column names

final_data = data.frame(mvp_share_comparison) #convert tibble to data frame to make it easy to work wit


#sort data frame by year and mvp_share in descending order to help sort the player with the highest mvp

prediction_Name <- final_data[order(final_data$Year, final_data$Predicted_MVP_Share, decreasing = TRUE)

df1 <- data.frame()
index <- 1

while(index < 333)
{
  added_row <- prediction_Name[index,]
  df1 = rbind(added_row,df1)
  index<-index+5 #add every 5th player to data frame because they have the highest mvp_share for that y

}

df1<-df1[order(df1$Year, decreasing = TRUE), ] #sort data frame by year

predicted_mvp_winners <- df1[1:2]

actual_MVPS_df <- data.frame(actual_MVPs)

mvp_comparison <-bind_cols(actual_MVPS_df$Year, actual_MVPS_df$Player, predicted_mvp_winners$Player)

colnames(mvp_comparison) <- c("Year", "Actual_NBA_MVP", "Predicted_NBA_MVP")

#Code below is checking if predicted MVPs and actual MVPs are same and adding label: Correct or Wrong
label <- c()
index <- 1
num_Correct <- 0
while(index < 68)
{
  if(mvp_comparison$Actual_NBA_MVP[index]==mvp_comparison $Predicted_NBA_MVP[index])
  {
    label <- append(label, "CORRECT")
    index <- index +1
```

```
    num_Correct <- num_Correct + 1 #increment number of correctly predicted mvps
  }
  else
  {
    label <- append(label, "WRONG")
    index <- index +1
  }
}
mvp_comparison <-bind_cols(mvp_comparison, label) #adds label to table
colnames(mvp_comparison) <- c("Year", "Actual_NBA_MVP", "Predicted_NBA_MVP", "Label") #sets column name
mvp_comparison
```

```
## # A tibble: 67 x 4
##     Year Actual_NBA_MVP        Predicted_NBA_MVP     Label
##    <int> <chr>                 <chr>                 <chr>
##  1  2022 Nikola Jokic          Nikola Jokic          CORRECT
##  2  2021 Nikola Jokic          Nikola Jokic          CORRECT
##  3  2020 Giannis Antetokounmpo Giannis Antetokounmpo CORRECT
##  4  2019 Giannis Antetokounmpo Giannis Antetokounmpo CORRECT
##  5  2018 James Harden          James Harden          CORRECT
##  6  2017 Russell Westbrook     James Harden          WRONG
##  7  2016 Stephen Curry         Stephen Curry         CORRECT
##  8  2015 Stephen Curry         Stephen Curry         CORRECT
##  9  2014 Kevin Durant          Kevin Durant          CORRECT
## 10  2013 LeBron James          LeBron James          CORRECT
## # ... with 57 more rows
```

```
overall_accuracy <- (num_Correct / 67.0) * 100 #overall percentage of correctly predicted mvps over tot

overall_accuracy
```

```
## [1] 58.20896
```

It correctly predicted the NBA MVP with an accuracy of 58.2%, which might appear to be low but this is reasonable since the model was only trained on some values and not trained on others.

Predicting 2023 NBA MVP

Using the stats from the Top 10 NBA MVP candidates for the first half of the regular season to predict the mvp for the 2023 season. Since the stats only cover half of the season, this prediction might not be accurate as there are several more games left to go! I will be applying the ridge regression model to this data set.

```
currentMVPCandidates <-read.csv('/Users/praveenmanimaran/Desktop/PSTAT131_FinalProject/data/NBA_MVP_202

currentMVPCandidates <- currentMVPCandidates %>%
  clean_names() #cleaning names

final_tibble2 <- augment(mvp_ridge_fit2, new_data = currentMVPCandidates) #applying ridge regression to

#creates table of players with predicted mvp_share
final_comparison <- bind_cols(final_tibble2$player, final_tibble2$.pred)
```

```r
colnames(final_comparison) <- c("Player Name", "Predicted mvp_share") #set column names
final_comparison %>% arrange(-`Predicted mvp_share`)
```

```
## # A tibble: 10 x 2
##    `Player Name`          `Predicted mvp_share`
##    <chr>                                  <dbl>
##  1 Nikola Jokic                           0.688
##  2 Luka Doncic                            0.556
##  3 Anthony Davis                          0.553
##  4 Giannis Antetokounmpo                  0.484
##  5 Zion Williamson                        0.434
##  6 Stephen Curry                          0.431
##  7 Ja Morant                              0.357
##  8 Kevin Durant                           0.354
##  9 Jayson Tatum                           0.350
## 10 Devin Booker                           0.326
```

From this table, my model predicted Nikola Jokic to win the 2023 NBA MVP award!

However, since I was only able to use stats from half of the regular season and there is still 40+ more games to go this might not be the actual end result. Players could potentially do better in the second half of the season and could beat out Jokic. Jokic has also won the award in back to back seasons (2021 and 2022), so voters may end up getting tired of voting for him again.

Conclusion

After testing different models, the model that yielded the best results in terms of predicting the mvp_share was the Ridge Regression Model. I found it interesting that all of my models had a very similar RMSE and that there wasn't too big of a difference. I was also surprised to see that ridge regression performed the best as I had originally thought that the random forest model would perform better since it can handle large amounts of variables and see which of them are related and not related to each other. Even though it is unlikely to have a lower test RMSE than a training RMSE, it can be seen that my model did not overfit the data.

I was able to understand which features were the most important in picking out the MVP (Player with the highest mvp_share) and found out that win shares were primarily the driving factor. This makes sense because win shares are the estimated number of wins contributed by that player for that regular season and a player who contributed more wins is deemed more valuable.

I do believe that since my rmse of 0.22 is relatively high as my data ranges from 0 to 1, that there could be other driving factors that could result in one player being picked MVP over another. In order to improve upon my project, I would definitely include the ranking of the player's team record for that regular season and a player's efficiency rating. These two variables are also very important in determining the impact that a player has on their team and how valuable they are.

It is almost impossible to make a model that can accurately predict the MVP 100% of the time as there could be different reasons as to how reporters make their votes which a model might not be able to predict. Reporters can sometimes have bias towards players they like and dislike and over the years, it has been hard to tell why certain players won the award over others. Reporters also get tired of voting for the same player again and again, so they end up voting for a different candidate instead of the player who might actually be more "valuable".

Over the past 10 years, the criteria needed to win the award has skyrocketed as players need to have stats that go above and beyond and also have a good team record. Player A with the best stats and have a low team record might not considered as valuable as Player B with above average stats and a good team record. Does this mean that Player B is a better player than Player A? Or is Player B more valuable than Player A simply because Player B has the better team? These are the kinds of questions I want to answer to improve upon my project in the future because these are the questions most NBA fans have been asking and debating about with each other for years and years.

Overall, the NBA MVP data set has allowed me to get a better understanding of how the voting process works and how it has changed over the course of NBA History. I was glad to be able to apply the models that I had learned in class to something I am passionate about as a both a fan and someone who enjoys playing basketball.