DA5030 Final Project Aleksandr Prystupa

Introduction

The NBA has launched into a new era where player performances are now judged by much more than the amount of points they have scored, or the amount of rebounds they collected. Advanced metrics are attempting to find diamonds in the rough, showing how some seemingly undervalued players are contributing to their teams. Some of the most interesting of these metrics are the RAPTOR, PREDATOR and WAR metrics. It would be extremely beneficial for an NBA analyst to know if players could be predicted as All Stars via these advanced metrics as it could help analysts sign players to contracts before they become too expensive, or find value in a player who is not drawing a lot of attention. The aim of this project is to test just that and see if Machine Learning Models can predict all stars. In doing so, Models may also identify potential players of interest who seem to be overlooked, or hint that some players who are all stars are undeserving.

Required Packages

```
pacman::p_load("tidyr", "dplyr", "psych", "gmodels", "smotefamily", "ggplot2",
               "randomForest", "rpart", "ipred", "caret", "Amelia")
```

Loading Data

```
# Data comes in test & train files, but its not clear how they are split
# I decided its best to combine them and then do the splitting myself later
raptor_train <- read.csv("nba_raptor_train.csv")
raptor_test <- read.csv("nba_raptor_test.csv")
raptor_all <- rbind(raptor_test, raptor_train)
```

Data Preparation

Subsetting Players above Year 2010 & 3000 possessions

```
# Data set has multiple duplicate columns and many duplicate rows
# Taking only relevant columns
important.cols <- c("player_name_x", "mp_x","season_x", "poss_x",
                    "raptor_offense_x", "raptor_defense_x",
          "raptor_total_x", "war_reg_season_x", "predator_offense_x",
          "predator_defense_x",
```

```
          "predator_total_x", "pace_impact_x")

# Keeping important columns
sub.raptor_all <- raptor_all[important.cols]

# Subsetting above 2010 up & > 3000 possessions (Explain why I did 3000+ possessions)
above.2010.3000 <- subset(sub.raptor_all, sub.raptor_all$season_x > 2010 &
                            sub.raptor_all$poss_x > 3000)

# Finding Unique Rows
above.2010.3000 <- above.2010.3000 %>%
  distinct()
```

I have used data above the year 2010 because around this year was when the 3 point revolution in the NBA began. At this point players became much more efficient and as such a players advanced stats will likely do a better job reflecting how well they play than before 2010. Also, data from this data set was much more sparse in the years before 2010. Since we want to predict all stars I made a cutoff at 3000+ possessions in a season. I find that 3000+ posessions indicated that you played enough possessions in a season to potentially vote for you as an all star. I would not want someone who only played 1000 possession in the whole season, but has very good advanced stats to be potentially predicted to be an all star, because this is never the case in real life.

List of All Stars for every year

```
# I have to manually add who was all stars in each specific year since
# that does not come with the data set
# I found who was an all star by just going on NBA Wikipedia pages
ALL.STAR.2011 <- c("Derrick Rose", "Dwyane Wade", "LeBron James",
                   "Amar'e Stoudemire", "Dwight Howard",
                   "Ray Allen", "Chris Bosh", "Kevin Garnett",
                   "Al Horford", "Joe Johnson", "Paul Pierce",
                   "Rajon Rondo", "Chris Paul", "Kobe Bryant",
                   "Kevin Durant", "Carmelo Anthony", "Yao Ming",
                   "Tim Duncan", "Pau Gasol", "Manu Ginobli",
                   "Blake Griffin", "Kevin Love", "Dirk Nowitzki",
                   "Russell Westbrook", "Deron Williams")

ALL.STAR.2012 <- c("Chris Paul", "Kobe Bryant", "Kevin Durant",
                   "Blake Griffin", "Andrew Bynum", "LaMarcus Aldridge",
                   "Marc Gasol", "Kevin Love", "Steve Nash", "Dirk Nowitzki",
                   "Tony Parker", "Russell Westbrook",
                   "Derrick Rose", "Dwyane Wade", "LeBron James", "Carmelo Anthony",
                   "Dwight Howard", "Chris Bosh",
                   "Luol Deng", "Roy Hibbert", "Andre Iguodala", "Joe Johnson",
                   "Paul Pierce", "Rajon Rondo", "Deron Williams")

ALL.STAR.2013 <- c("Rajon Rondo", "Dwyane Wade", "LeBron James",
                   "Carmelo Anthony", "Kevin Garnett",
                   "Chris Bosh", "Tyson Chandler", "Luol Deng",
                   "Paul George", "Jrue Holiday", "Kyrie Irving",
                   "Joakim Noah", "Brook Lopez", "Chris Paul",
```

```
                  "Kobe Bryant", "Kevin Durant", "Blake Griffin",
                  "Dwight Howard", "LaMarcus Aldridge", "Tim Duncan",
                  "James Harden", "David Lee", "Tony Parker",
                  "Zach Randolph", "Russell Westbrook")

ALL.STAR.2014 <- c("Dwyane Wade", "Kyrie Irving", "LeBron James",
                  "Paul George", "Carmelo Anthony",
                  "Joakim Noah", "Roy Hibbert", "Chris Bosh",
                  "Paul Millsap", "John Wall", "Joe Johnson",
                  "DeMar DeRozan", "Stephen Curry", "Kobe Bryant",
                  "Kevin Durant", "Blake Griffin",
                  "Kevin Love", "Dwight Howard", "LaMarcus Aldridge",
                  "Dirk Nowitzki", "Chris Paul",
                  "James Harden", "Tony Parker", "Damian Lillard",
                  "Anthony Davis")

ALL.STAR.2015 <- c("John Wall", "Kyle Lowry", "LeBron James",
                  "Pau Gasol", "Carmelo Anthony",
                  "Al Horford", "Chris Bosh", "Paul Millsap", "Jimmy Butler",
                  "Dwyane Wade", "Jeff Teague",
                  "Kyrie Irving", "Kyle Korver", "Stephen Curry", "Kobe Bryant",
                  "Anthony Davis", "Marc Gasol",
                  "Blake Griffin", "LaMarcus Aldridge", "Tim Duncan", "Kevin Durant",
                  "Klay Thompson", "Russell Westbrook",
                  "James Harden", "Chris Paul", "DeMarcus Cousins",
                  "Damian Lillard", "Dirk Nowitzki")

ALL.STAR.2016 <- c("Dwyane Wade", "Kyle Lowry", "LeBron James", "Paul George",
                  "Carmelo Anthony",
                  "Jimmy Butler", "Chris Bosh", "John Wall", "Paul Millsap",
                  "DeMar DeRozan",
                  "Andre Drummond", "Isaiah Thomas", "Pau Gasol",
                  "Al Horford", "Stephen Curry",
                  "Russell Westbrook", "Kobe Bryant", "Kevin Durant",
                  "Kawhi Leonard", "Chris Paul",
                  "LaMarcus Aldridge", "James Harden", "Anthony Davis",
                  "DeMarcus Cousins", "Klay Thompson", "Draymond Green")


ALL.STAR.2017 <- c("Kyrie Irving", "DeMar DeRozan", "LeBron James",
                  "Jimmy Butler", "Giannis Antetokounmpo",
                  "Isaiah Thomas", "John Wall", "Kevin Love",
                  "Carmelo Anthony", "Kyle Lowry", "Paul George",
                  "Kemba Walker", "Paul Millsap", "Stephen Curry",
                  "James Harden", "Kevin Durant",
                  "Kawhi Leonard", "Anthony Davis", "Kemba Walker",
                  "Russell Westbrook", "Klay Thompson",
                  "Draymond Green", "DeMarcus Cousins", "Marc Gasol",
                  "DeAndre Jordan", "Gordon Hayward")


ALL.STAR.2018 <- c("Kyrie Irving", "DeMar DeRozan", "LeBron James",
                  "Giannis Antetokounmpo", "Joel Embiid",
```

```
                    "Bradley Beal", "Goran Dragic", "Al Horford",
                    "Kevin Love", "Kyle Lowry", "Victor Oladipo",
                    "Kristaps Porzingis", "John Wall", "Andre Drummond",
                    "Kemba Walker", "Stephen Curry", "James Harden",
                    "Kevin Durant", "Anthony Davis", "DeMarcus Cousins",
                    "Russell Westbrook", "Damian Lillard", "Draymond Green",
                    "Karl-Anthony Towns", "LaMarcus Aldridge", "Klay Thompson",
                    "Jimmy Butler", "Paul George")

ALL.STAR.2019 <- c("Kemba Walker", "Kyrie Irving", "Kawhi Leonard",
                    "Giannis Antetokounmpo", "Joel Embiid",
                    "Kyle Lowry", "Victor Oladipo", "Khris Middleton",
                    "Bradley Beal", "Ben Simmons", "Blake Griffin",
                    "Nikola Vucevic", "Dwyane Wade", "D'Angelo Russell",
                    "Stephen Curry", "James Harden", "Kevin Durant", "Paul George",
                    "LeBron James", "Russell Westbrook", "Damian Lillard",
                    "Klay Thompson", "Anthony Davis", "LaMarcus Aldridge",
                    "Nikola Jokic", "Karl-Anthony Towns")

ALL.STAR.2020 <- c("Kemba Walker", "Trae Young", "Giannis Antetokounmpo",
                    "Pascal Siakam", "Joel Embiid",
                    "Kyle Lowry", "Ben Simmons", "Jimmy Butler",
                    "Khris Middleton", "Bam Adebayo", "Jayson Tatum",
                    "Domantis Sabonis", "James Harden", "Luka Doncic",
                    "LeBron James", "Kawhi Leonard", "Anthony Davis",
                    "Chris Paul", "Russell Westbrook", "Damian Lillard",
                    "Donovan Mitchell", "Brandom Ingram",
                    "Nikola Jokic", "Rudy Gobert", "Devin Booker")

ALL.STAR.2021 <- c("Bradley Beal", "Kyrie Irving", "Giannis Antetokounmpo",
                    "Kevin Durant", "Joel Embiid",
                    "Jaylen Brown", "James Harden", "Zach Lavine",
                    "Ben Simmons", "Julius Randle", "Domantas Sabonis",
                    "Jayson Tatum", "Nikola Vucevic", "Stephen Curry",
                    "Luka Doncic", "LeBron James", "Kawhi Leonard",
                    "Nikola Jokic", "Devin Booker", "Mike Conley",
                    "Damian Lillard", "Donovan Mitchell", "Chris Paul",
                    "Anthony Davis", "Paul George", "Zion Williamson",
                    "Rudy Gobert")
```

I had to go on nba.com and find the list of all the players that were All Stars in any given year. If that player was not found in our data set they were simply not added.

Separate each year into its own data frame

```
# Subset by year so then I can add the extra all stars column
data.2011 <- subset(above.2010.3000, season_x == 2011)
data.2012 <- subset(above.2010.3000, season_x == 2012)
data.2013 <- subset(above.2010.3000, season_x == 2013)
data.2014 <- subset(above.2010.3000, season_x == 2014)
data.2015 <- subset(above.2010.3000, season_x == 2015)
```

```
data.2016 <- subset(above.2010.3000, season_x == 2016)
data.2017 <- subset(above.2010.3000, season_x == 2017)
data.2018 <- subset(above.2010.3000, season_x == 2018)
data.2019 <- subset(above.2010.3000, season_x == 2019)
data.2020 <- subset(above.2010.3000, season_x == 2020)
data.2021 <- subset(above.2010.3000, season_x == 2021)
```

These were separated in order to impute the ALL STARS by year.

Creating Function to add All Stars to column

```
# Function to add All Stars to a new column
# 1 --> All Star 0 --> Not an All Star
add.all.star <- function(nba.df, all.star.list) {
  length.nba.df <- rep(NA, nrow(nba.df))
  nba.df$ALL.STAR <- ifelse(nba.df$player_name_x %in% all.star.list, 1, 0)
  return(nba.df)
}
```

Adding All Stars and rejoining data frames

```
# Add the all start column
data.2011 <- add.all.star(data.2011, ALL.STAR.2011)
data.2012 <- add.all.star(data.2012, ALL.STAR.2012)
data.2013 <- add.all.star(data.2013, ALL.STAR.2013)
data.2014 <- add.all.star(data.2014, ALL.STAR.2014)
data.2015 <- add.all.star(data.2015, ALL.STAR.2015)
data.2016 <- add.all.star(data.2016, ALL.STAR.2016)
data.2017 <- add.all.star(data.2017, ALL.STAR.2017)
data.2018 <- add.all.star(data.2018, ALL.STAR.2018)
data.2019 <- add.all.star(data.2019, ALL.STAR.2019)
data.2020 <- add.all.star(data.2020, ALL.STAR.2020)
data.2021 <- add.all.star(data.2021, ALL.STAR.2021)

# Rejoin all the years together
all.star.data <- rbind(data.2011, data.2012, data.2013, data.2014, data.2015,
                       data.2016, data.2017, data.2018, data.2019,
                       data.2020, data.2021)
```

View All Star Data

```
head(all.star.data)
```

```
##       player_name_x mp_x season_x poss_x raptor_offense_x raptor_defense_x
## 13 DeMarcus Cousins 2309     2011   4632      -3.99018881       -0.5163295
## 41  Wesley Matthews 2960     2011   5462       1.29530693       -0.4890027
## 46     James Harden 2726     2011   5303       3.42163424       -0.3234184
```

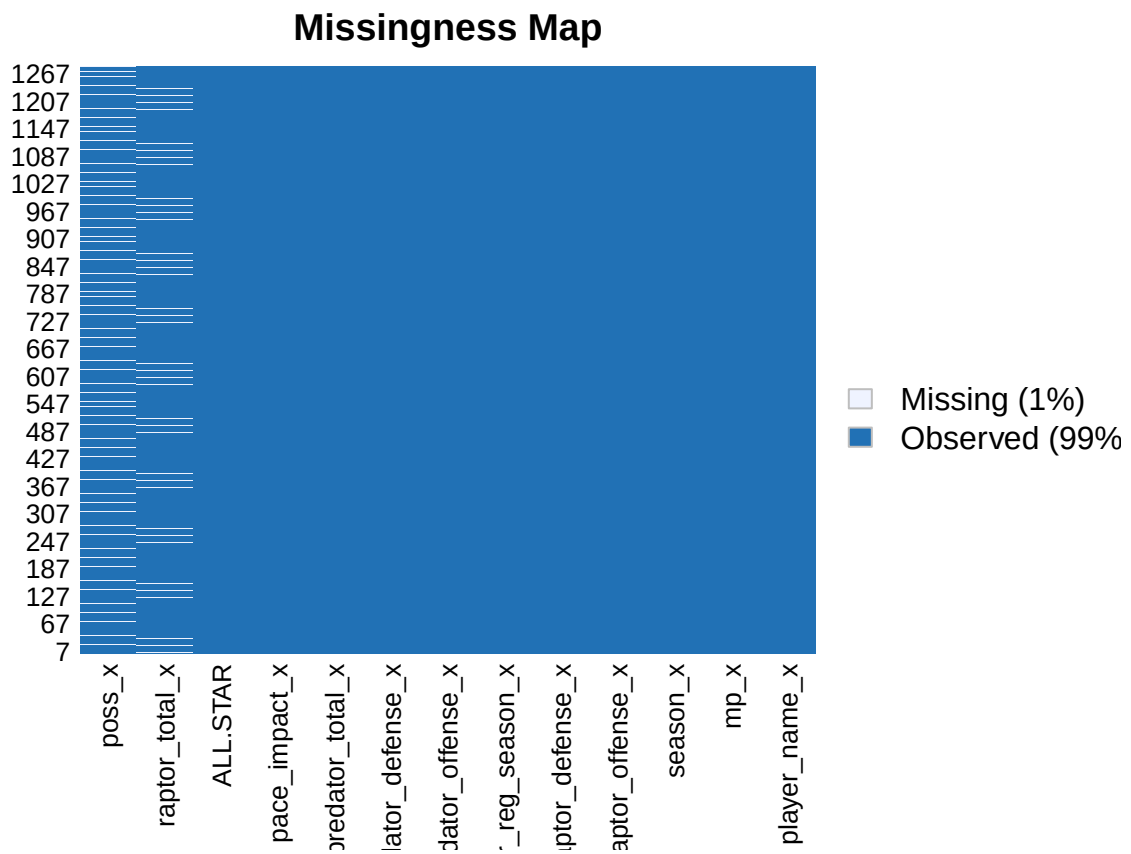
```
## 54      Greg Monroe 2222     2011   4177       0.34107913       -0.3665932
## 71    Stephen Curry 2489     2011   4972       3.25524437       -0.3405174
## 94  Darren Collison 2506     2011   4987       0.03574525       -1.4342527
##    raptor_total_x war_reg_season_x predator_offense_x predator_defense_x
## 13    -4.50651827        -2.081331        -3.27942558       -0.007717626
## 41     0.80630428         5.184184         1.36282746       -0.183140424
## 46     3.09821579         5.791326         3.33961485        0.008035568
## 54    -0.02551408         3.073817         0.48205809        0.375483953
## 71     2.91472700         7.197756         3.23181858       -0.013222066
## 94    -1.39850742         1.657194        -0.08618748       -1.286254782
##    predator_total_x pace_impact_x ALL.STAR
## 13        -3.287143    0.54319928        0
## 41         1.179687    0.09208183        0
## 46         3.347650    0.43647066        0
## 54         0.857542   -0.44764960        0
## 71         3.218597    0.05426188        0
## 94        -1.372442   -0.58686407        0
```

Create Fake Data to Demonstrate Imputation and Dealing with NA's

```
# Fake data
fake.all.star <- all.star.data

# Indices to make NA's with
NA.indices.1 <- seq(from = 1, to = nrow(fake.all.star), by = 10)
NA.indices.2 <- seq(from = 3, to = nrow(fake.all.star), by = 15)

# Adding NA's to poss_x & adding NA's to Raptor_total_x
fake.all.star$poss_x[NA.indices.1] <- NA
fake.all.star$raptor_total_x[NA.indices.2] <- NA
```

View Missing Data with Amelia package

```
# View Missing Values with Amelia missmap function
Amelia::missmap(fake.all.star)
```



Impute total possesstion by looking at total minutes played

```
# Find mean of minutes played
(mean.minutes <- mean(fake.all.star$mp_x))
```

```
## [1] 2240.131
```

```
# Find mean of possessions played (Remove NA's)
(mean.possessions <- mean(fake.all.star$poss_x, na.rm = T))
```

```
## [1] 4576.02
```

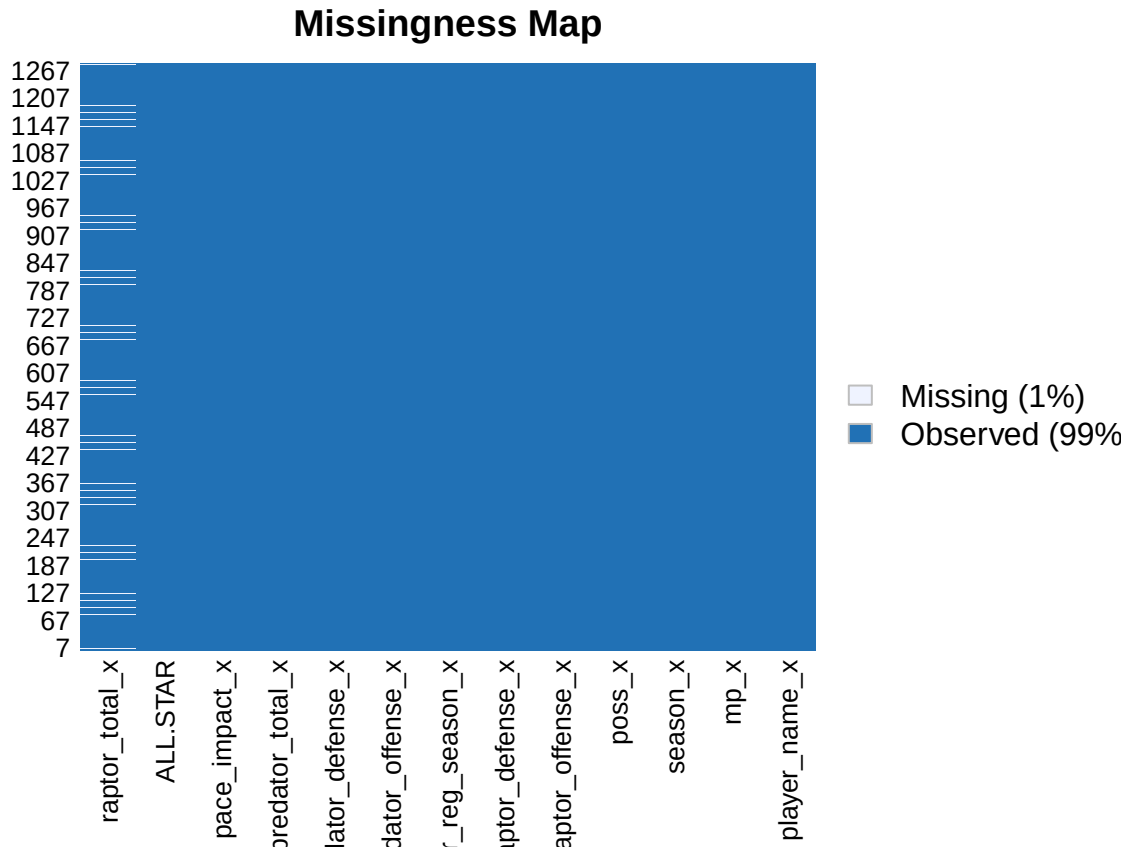
```
# Find ratio to use for imputation
(ratio.poss.minutes <- mean.possessions/mean.minutes)
```

```
## [1] 2.042747
```

```
# Impute possessions by multiplying minutes by ratio where there are NA values
for (i in 1:nrow(fake.all.star)) {
  if (is.na(fake.all.star$poss_x[i])) {
    fake.all.star$poss_x[i] <- round(fake.all.star$mp_x[i] * ratio.poss.minutes, 0)
  }
}
```

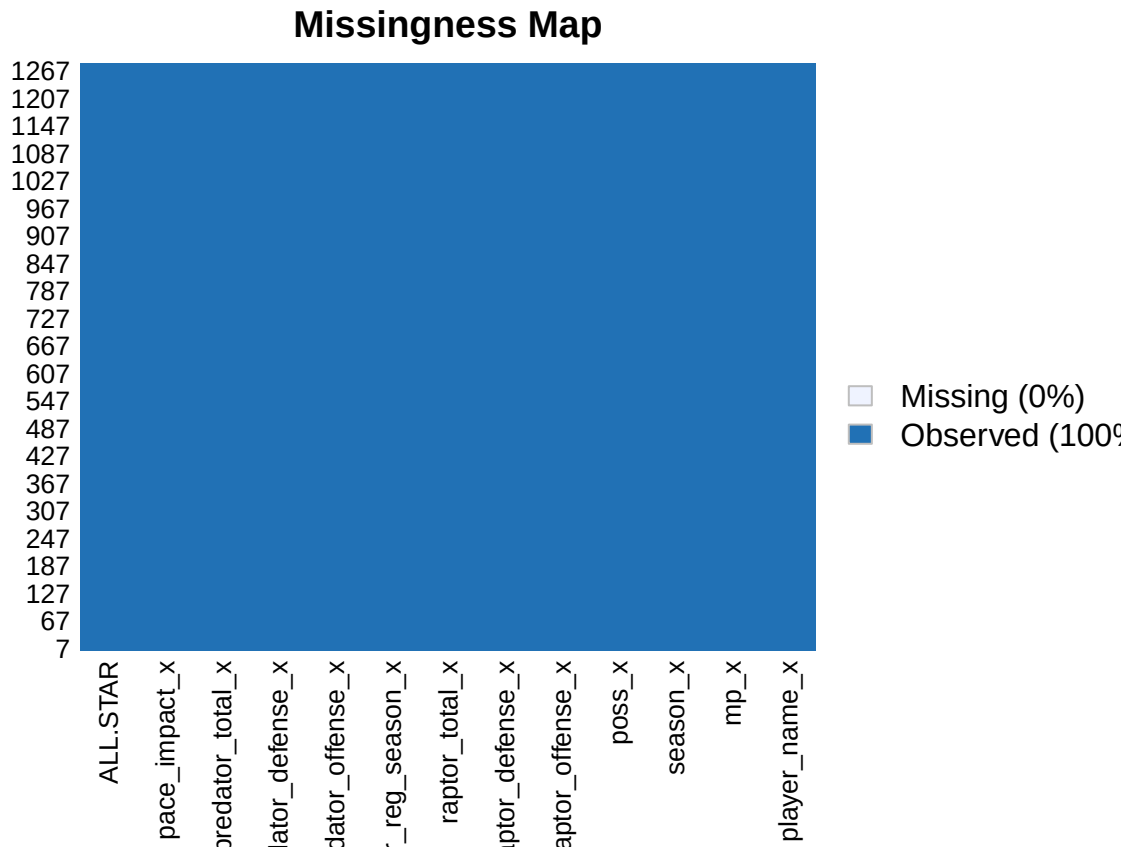
View Amelia again to see if values were all imputd

```
# View Missing Values with Amelia missmap function
Amelia::missmap(fake.all.star) # Now we only have raptor_total left to impute
```



Impute Raptor Total

```
# Impute Raptor Total with Raptore Offense & Defense Values
for (i in 1:nrow(fake.all.star)) {
  if (is.na(fake.all.star$raptor_total_x[i])) {
    fake.all.star$raptor_total_x[i] <- fake.all.star$raptor_offense_x[i] +
      fake.all.star$raptor_defense_x[i]
  }
}

# View Amelia to see if there are any missing values left
Amelia::missmap(fake.all.star) # No missing values!
```



This section was done to show that I know how to deal with NA values, as my data already came with no NA values. I randomly inserted NA values into both possession and raptor total columns. I then showed that I can impute possessions by finding the ratio between the possessions and minutes played. Then I can replace NA values with minutes played times that ratio. For raptor total I showed that I can impute the NA values simply by adding the raptor offense and raptor defense

Data Exploration

Checking Frequency of All Star vs Not All Star

```
# View Frequency of All Star vs not all-star
all.star.data$ALL.STAR <- as.factor(all.star.data$ALL.STAR) # make into factor
all.stars <- all.star.data$ALL.STAR

# Looking at number of all stars versus non-all stars
table(all.stars)
```

```
## all.stars
##    0    1
## 1078  204
```

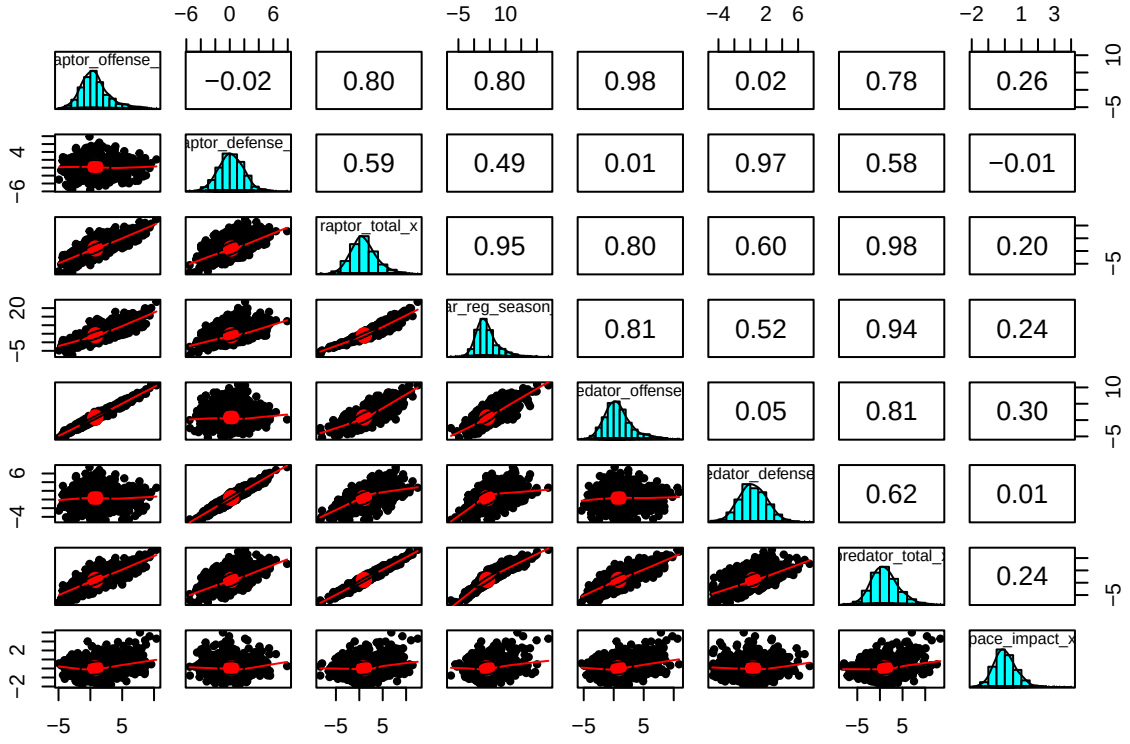
```
# Looking at proportion of all stars versus non-all stars
prop.table(table(all.stars))
```

```
## all.stars
##         0         1
## 0.8408736 0.1591264
```

The distribution of Non-All Star to All-Star is around 85:15. This means that their is some class imbalance and it may be worth looking at using a package to deal with it. I ended up using SMOTE in 2 of my 3 models to deal with class imbalance.

View Distribution & Correlations of Predictor Variables

```
# Subset only predictors
all.star.predictors <- all.star.data[-c(1, 2, 3, 4)]

# Make ALL.STAR a factor
all.star.predictors$ALL.STAR <- as.factor(all.star.predictors$ALL.STAR)

# View pairs.panels
psych::pairs.panels(all.star.predictors[-ncol(all.star.predictors)])
```



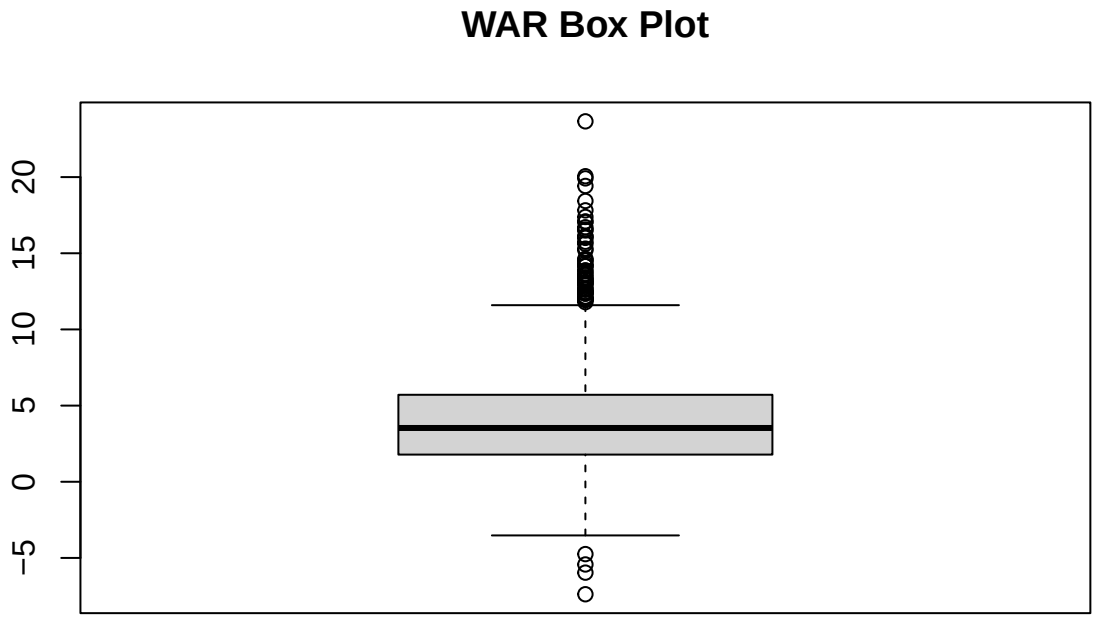
All of our variables seem to be normally distributed which means that we do not need to transform them. You can see that raptor_offense & predator offense have a very high correlation. Additionally, you can see that raptor_defense & predator defense have a very high correlation. Although they have an extremely high correlation, part of this project is figuring out if one of these advanced metrics has a better chance at all star prediction and therefore I am leaving both in for all the models I will do to see what kind of predictive power they have and how they compare against each other.

Looking for Outliers

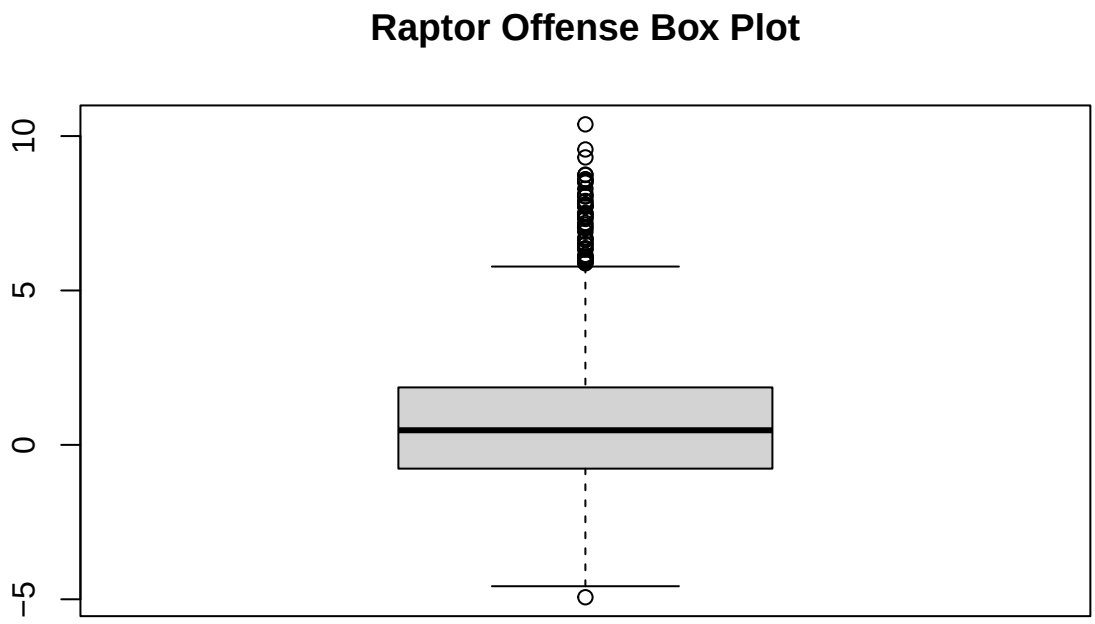
```
summary(all.star.predictors)
```

```
##  raptor_offense_x  raptor_defense_x  raptor_total_x    war_reg_season_x
##  Min.   :-4.9340   Min.   :-5.5480   Min.   :-8.3479   Min.   :-7.383
##  1st Qu.:-0.7690   1st Qu.:-0.9405   1st Qu.:-0.8902   1st Qu.: 1.785
##  Median : 0.4731   Median : 0.1377   Median : 0.7199   Median : 3.531
##  Mean   : 0.7954   Mean   : 0.1847   Mean   : 0.9801   Mean   : 4.140
##  3rd Qu.: 1.8612   3rd Qu.: 1.3084   3rd Qu.: 2.5551   3rd Qu.: 5.709
##  Max.   :10.3794   Max.   : 7.9074   Max.   :12.4879   Max.   :23.659
##  predator_offense_x predator_defense_x predator_total_x  pace_impact_x
##  Min.   :-5.3596    Min.   :-4.7815    Min.   :-8.1800   Min.   :-1.89129
##  1st Qu.:-0.7866    1st Qu.:-0.8516    1st Qu.:-0.9501   1st Qu.:-0.42465
##  Median : 0.4569    Median : 0.2960    Median : 0.9277   Median :-0.02755
##  Mean   : 0.8005    Mean   : 0.3386    Mean   : 1.1392   Mean   : 0.04743
##  3rd Qu.: 1.9241    3rd Qu.: 1.5204    3rd Qu.: 2.8580   3rd Qu.: 0.46175
##  Max.   :10.8931    Max.   : 7.4592    Max.   :13.3479   Max.   : 3.99692
##  ALL.STAR
##  0:1078
##  1: 204
##
##
##
##
```

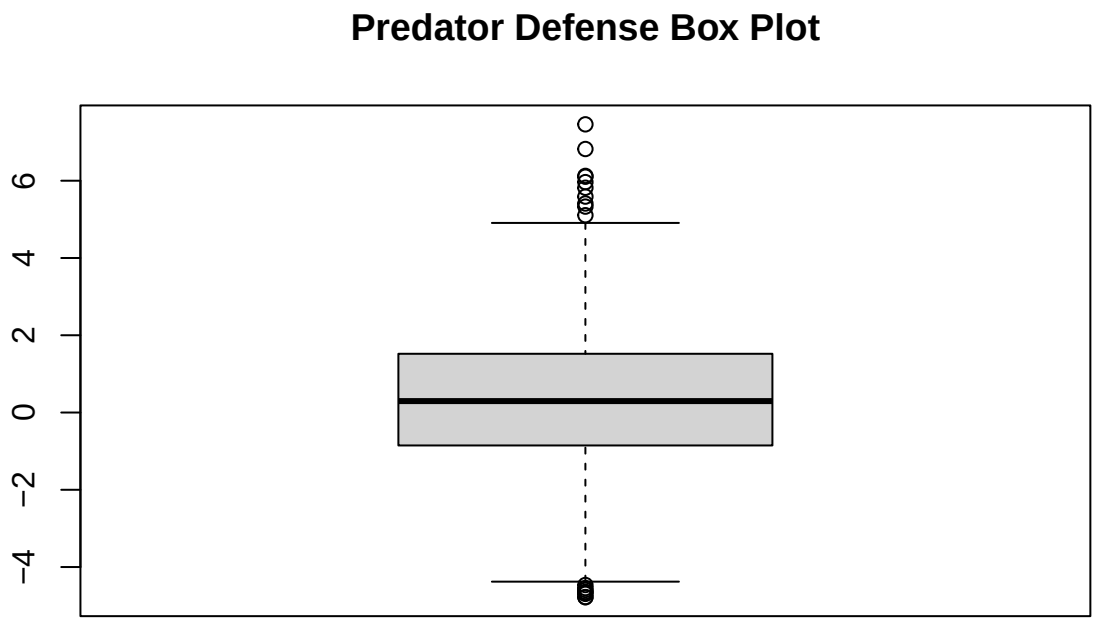
```
# Box plot of WAR (Wins against replacement)
boxplot(all.star.predictors$war_reg_season_x, main = "WAR Box Plot")
```




```
# Box plot of Raptor Offense
boxplot(all.star.predictors$raptor_offense_x, main = "Raptor Offense Box Plot")
```




```
# Box plot of Predator Defense
boxplot(all.star.predictors$predator_defense_x, main = "Predator Defense Box Plot")
```



One thing we can immediately notice with these box plots is that there are a lot of outliers as defined by points being outside the Interquartile Range. However, in our case we must keep these outliers, specifically because it is likely that the outliers in our data set are the ones that are going to be predicted as all stars. After all, all stars in the NBA are generally outliers as to how good they are compared to the rest of the league. What these box plots do a great job of instead is illustrating a group of players that may be more likely to be predicted all stars.

Normalizing Data for SVM model

Normalize Function

```
normalize <- function(x) {
  return((x - min(x)) / (max(x) - min(x)))
}
```

Normalizing Data

```
# Normalize all columns except predictor column
all.star.norm <- as.data.frame(lapply(all.star.predictors[-9], normalize))

# Add back the predictor column
all.star.norm <- cbind(all.star.norm, all.star.predictors[9])
```

We need normalized data for the SVM model that I will run.

Train/Test/Validation

```
# Set seed for reproducibility
set.seed(300)

# Train/Test/Validation Splits
splits <- c(train = 0.6, test = 0.2, validate = 0.2)

# Cutting the sample based on splits
split.label <- sample(cut(
  seq(nrow(all.star.predictors)),
  nrow(all.star.predictors) * cumsum(c(0, splits)),
  labels = names(splits)
))

# Get 3 data frames for normalized and un-normalized data
all.stars <- split(all.star.predictors, split.label)
normalized.all.stars <- split(all.star.norm, split.label)
```

Get data frames for test/train/vdalidate for normalized and non-normalized data

```
# Train/Test/Validation for non normalized data
all.star.train <- all.stars$train
all.star.test <- all.stars$test
all.star.validate <- all.stars$validate

# Train/Test/Validation for Normalized data
all.star.norm.train <- normalized.all.stars$train
all.star.norm.test <- normalized.all.stars$test
all.star.norm.validate <- normalized.all.stars$validate
```

Add back All Star Train & Test so we can use later

```
all.stars <- rbind(all.star.train, all.star.test)
all.stars.norm <- rbind(all.star.norm.train, all.star.norm.test)
```

What we have now done is create a validation and normalized validation set that will not be seen by any of the models through training or testing. The Training and Testing data sets were combined again because for models where I use k-fold cross validation, the cross validation will be done on all of that data. For models were I do not use cross validation, the model will be trained on the train data and tested on the test data. At the very end when I do model comparison all of the models will be compared on the validationd data.

Viewing proportions to see if they are similar

```
## Proportion for non normalized data
# Train
prop.table(table(all.star.train$ALL.STAR))
```

```
##
##        0        1
## 0.840052 0.159948
```

```
# Test
prop.table(table(all.star.test$ALL.STAR))
```

```
##
##        0        1
## 0.828125 0.171875
```

```
# Validate
prop.table(table(all.star.validate$ALL.STAR))
```

```
##
##         0         1
## 0.8560311 0.1439689
```

```
## Proportion for normalized data
# Train
prop.table(table(all.star.norm.train$ALL.STAR))
```

```
##
##        0        1
## 0.840052 0.159948
```

```
# Test
prop.table(table(all.star.norm.test$ALL.STAR))
```

```
##
##        0        1
## 0.828125 0.171875
```

```
# Validate
prop.table(table(all.star.norm.validate$ALL.STAR))
```

```
##
##         0         1
## 0.8560311 0.1439689
```

They are both similar proportions across test, train and validate. Additionally, they are identical between normalized and not normalized which means the random split went as planned.

Prinicipal Component Analysis

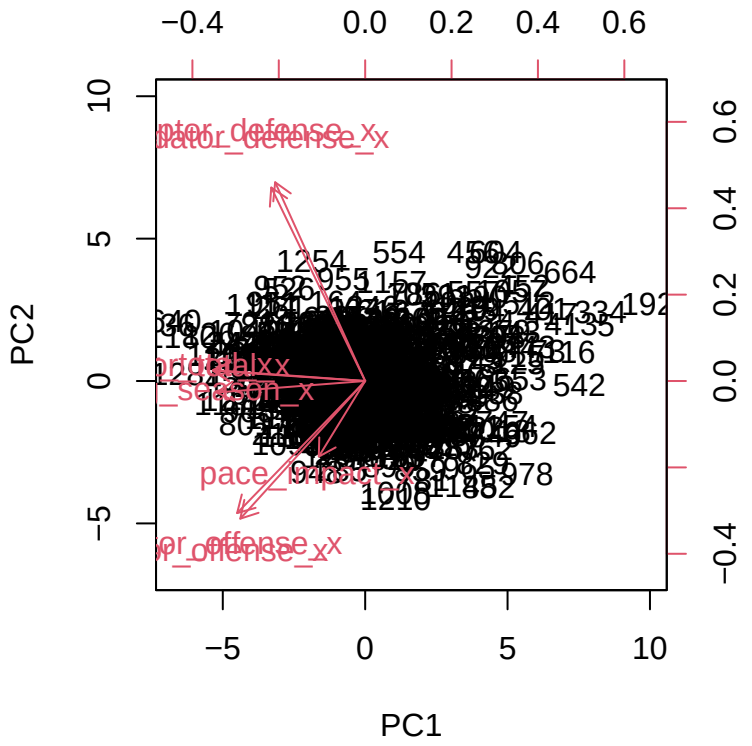
Get PCA values

```
# Creating a data frame without the response variable (ALL.STAR) for PCA
pca.df <- all.star.train[-9]

# Get PCA results and scale dataframe
pca.results <- prcomp(pca.df, scale = TRUE)

# Reverse the signs b/c eigenvectors point in negative direction by default
pca.results$rotation <- -1 * pca.results$rotation

# View Principal Components
pca.results$rotation
```

```
##                          PC1         PC2         PC3          PC4         PC5
## raptor_offense_x   -0.3614273 -0.39836079 -0.16654690  0.121017556 -0.25267754
## raptor_defense_x   -0.2604886  0.57520890  0.08842756 -0.053675184 -0.50729396
## raptor_total_x     -0.4401471  0.03198656 -0.07781161  0.062883866 -0.50271661
## war_reg_season_x   -0.4296280 -0.03317569 -0.05136597 -0.859057570  0.27134985
## predator_offense_x -0.3705615 -0.38222113 -0.11820749  0.256084080  0.13767356
## predator_defense_x -0.2718362  0.55994147  0.09042913  0.246290137  0.44763593
## predator_total_x   -0.4404160  0.02974544 -0.03845845  0.338014589  0.36328709
## pace_impact_x      -0.1342733 -0.21864904  0.96546367  0.006905502 -0.03380423
##                           PC6           PC7           PC8
## raptor_offense_x    0.540592613 -4.958991e-01  2.574755e-01
## raptor_defense_x   -0.388991890 -3.785369e-01  1.965399e-01
## raptor_total_x      0.191421149  6.312344e-01 -3.277429e-01
## war_reg_season_x   -0.008370361  2.745587e-11  2.072965e-11
## predator_offense_x -0.558801036 -2.548299e-01 -4.908036e-01
## predator_defense_x  0.412829396 -1.916223e-01 -3.690654e-01
## predator_total_x   -0.190245476  3.326855e-01  6.407539e-01
## pace_impact_x       0.029202181  4.280439e-12 -7.674948e-12
```

Plot Biplot

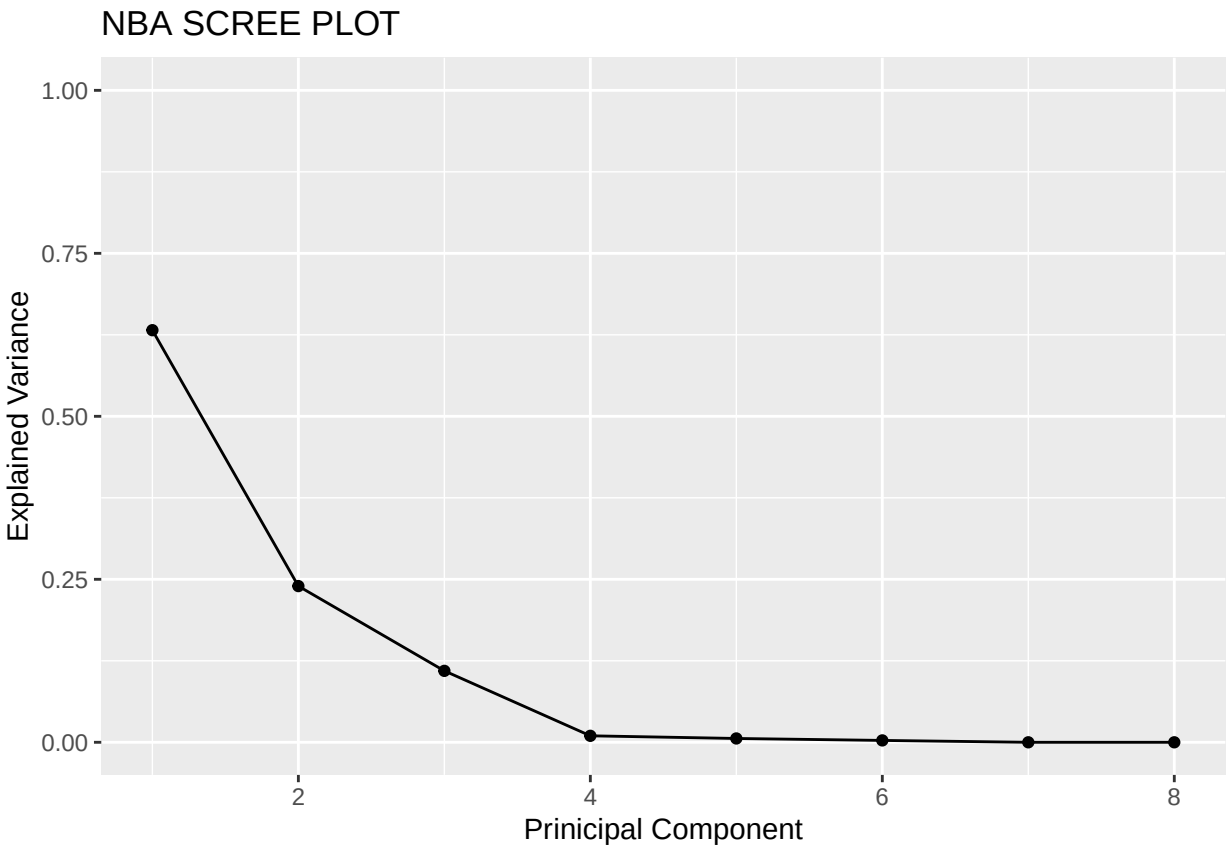
```
# Plotting biplot
biplot(pca.results, scale = 0)
```



As we can see the arrows for both offense and defense and total metrics are pointing the same way, indicating that they help explain the variance in the same way. Because of this I am going to get rid of the total metrics for the logistic regression model I am planning to do. This makes sense because total is just the sum of offense and defense and so does not help the model learn more from having it.

PCA Scree Plot to see how much variance is explained by first component

```
# Calculate total variance explained by each PC
var.explained <- pca.results$sdev^2 / sum(pca.results$sdev^2)

# Create Scree Plot
qplot(c(1:8), var.explained) +
        geom_line() +
        xlab("Prinicipal Component") +
        ylab("Explained Variance") +
        ggtitle("NBA SCREE PLOT") +
        ylim(0,1)
```



As you can see in the Scree Plot all the variance is explained by the first 4 components, with over 80% of it being explained by the first 2 components alone.

Training Models

Logistic Regression

Logistic Regression is an excellent choice to use as a model because logistic regression allows me to use regression to do classification. Additionally, Logistic Regression will allow me to easily find out what parameters are helpful for my model by backwards p-value elimination which will help my model be accurate on the validation data set. The model will also be checked with cross validation.

Setting up Cross Validation Parameters from Caret Package

```
# defining training control as CV
train_control <- trainControl(method = "cv", number = 10)
```

I am doing cross validation with a k = 10.

First Logistic Regression Model

```r
set.seed(125)

# Duplicating all.star.predictors to be able to easily delete variables
dummy.all.star <- all.stars

# Removing total predictors as suggested by PCA Analysis
dummy.all.star$predator_total_x <- NULL
dummy.all.star$raptor_total_x <- NULL

# Fitting log regression model 1
fit1.glm <- train(ALL.STAR ~., data = dummy.all.star,
                  method = "glm",
                  trControl = train_control)

# Summary model 1
summary(fit1.glm)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.34945  -0.33056  -0.17231  -0.08595   2.86165
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.22366    0.39118 -13.354  < 2e-16 ***
## raptor_offense_x   -1.04992    0.27945  -3.757 0.000172 ***
## raptor_defense_x    0.79308    0.29245   2.712 0.006690 **
## war_reg_season_x    0.57746    0.09816   5.883 4.03e-09 ***
## predator_offense_x  1.28155    0.27724   4.623 3.79e-06 ***
## predator_defense_x -1.05683    0.29636  -3.566 0.000362 ***
## pace_impact_x       0.28468    0.17470   1.630 0.103201
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 911.21  on 1024  degrees of freedom
## Residual deviance: 433.89  on 1018  degrees of freedom
## AIC: 447.89
##
## Number of Fisher Scoring iterations: 6
```

Second Logistic Regression Model

```r
# Removing pace_impact_x because it has highest p-value
dummy.all.star$raptor_defense_x <- NULL

# Running model with new training set
```

```r
set.seed(125)
fit.glm2 <- train(ALL.STAR ~., data = dummy.all.star,
                  method = "glm",
                  trControl = train_control)

# Summary model 2
summary(fit.glm2)
```

```
##
## Call:
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.28087  -0.33558  -0.17907  -0.09459   2.65993
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -5.34421    0.38402 -13.917  < 2e-16 ***
## raptor_offense_x   -1.09915    0.27925  -3.936 8.28e-05 ***
## war_reg_season_x    0.60743    0.09677   6.277 3.45e-10 ***
## predator_offense_x  1.26027    0.27658   4.557 5.20e-06 ***
## predator_defense_x -0.33521    0.12394  -2.705  0.00684 **
## pace_impact_x       0.29092    0.17220   1.689  0.09114 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 911.21  on 1024  degrees of freedom
## Residual deviance: 441.44  on 1019  degrees of freedom
## AIC: 453.44
##
## Number of Fisher Scoring iterations: 6
```

Third Logistic Regression Model

```r
# Removing pace_impact_x because it has highest p-value
dummy.all.star$predator_defense_x <- NULL

# Running model with new training set
set.seed(125)
fit.glm3 <- train(ALL.STAR ~., data = dummy.all.star,
                  method = "glm",
                  trControl = train_control)

# Summary model 2
summary(fit.glm3)
```

```
##
## Call:
```

```
## NULL
##
## Deviance Residuals:
##      Min        1Q    Median        3Q       Max
## -2.23516  -0.34145  -0.17822  -0.08404   2.64232
##
## Coefficients:
##                    Estimate Std. Error z value Pr(>|z|)
## (Intercept)        -4.85654    0.32639 -14.880  < 2e-16 ***
## raptor_offense_x   -0.85224    0.25793  -3.304 0.000953 ***
## war_reg_season_x    0.39964    0.05526   7.233 4.74e-13 ***
## predator_offense_x  1.26352    0.27207   4.644 3.41e-06 ***
## pace_impact_x       0.29921    0.17030   1.757 0.078921 .
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for binomial family taken to be 1)
##
##     Null deviance: 911.21  on 1024  degrees of freedom
## Residual deviance: 449.13  on 1020  degrees of freedom
## AIC: 459.13
##
## Number of Fisher Scoring iterations: 6
```

```
# Renaming statistically significant model
glm.best.fit <- fit.glm3
```

All of my predictors have a p-value of under 0.15, which is what I want to set as my cutoff. Now we can test the logistic regression model.

Testing Logistic Regression Model

```
# Vector of Columns that were significant based off my 0.15 p-value cutoff
significant.cols <- c("raptor_offense_x", "war_reg_season_x", "predator_offense_x",
                      "pace_impact_x")

# Get predictions with type response giving a probability
glm.predictions <- predict(glm.best.fit, all.star.test[ ,significant.cols],
                           type = "raw")

# View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.test$ALL.STAR, glm.predictions,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
```

```
## |-------------------------|
##
##
## Total Observations in Table:  256
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       203 |         9 |       212 |
##              |     0.793 |     0.035 |           |
## -------------|-----------|-----------|-----------|
##            1 |        15 |        29 |        44 |
##              |     0.059 |     0.113 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       218 |        38 |       256 |
## -------------|-----------|-----------|-----------|
##
##
```

Gather GLM Stats for Comparison against all Models

```
# (True positive + True negative)/Total
(accuracy.glm <- (25 + 213)/256)
```

```
## [1] 0.9296875
```

```
# True positive / (True positive + False positive)
(precision.glm <- 25/(25 + 8))
```

```
## [1] 0.7575758
```

```
# True positive / (True positive + False negative)
(recall.glm <- 25/(25 + 10))
```

```
## [1] 0.7142857
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.glm <-(2 *precision.glm * recall.glm) / (precision.glm + recall.glm))
```

```
## [1] 0.7352941
```

Support Vector Machine

An SVM is a great choice for a model for my data set because SVM specializes in binary classification. After testing the SVM I have found that it works best when it works with more balanced data and as you will see below I used the smotefamily package to balance data. Additionally, SVMs have a great number of parameters that can be tuned and I will be using the built in caret tune control to tune my SVM. The model will also be checked with cross validation.

Using Smote to deal with class imbalance

```
# Get data frame with balanced classes
balanced <- smotefamily::SMOTE(all.stars.norm[-9], all.stars.norm$ALL.STAR, K = 5)
balanced <- balanced$data

# Smote creates class variable, rename to ALL.STAR
colnames(balanced )[9] <- "ALL.STAR"

# Make ALL.STAR a numeric factor
balanced$ALL.STAR <- as.factor(as.numeric(balanced$ALL.STAR))
```

I am using smotefamily SMOTE function to deal with class imbalance.

View new proportions after Smote

```
prop.table(table(balanced$ALL.STAR))
```

```
##
##         0         1
## 0.5067927 0.4932073
```

The proportion is much closer to 50-50 now which should help use identify ALL.STAR cases better.

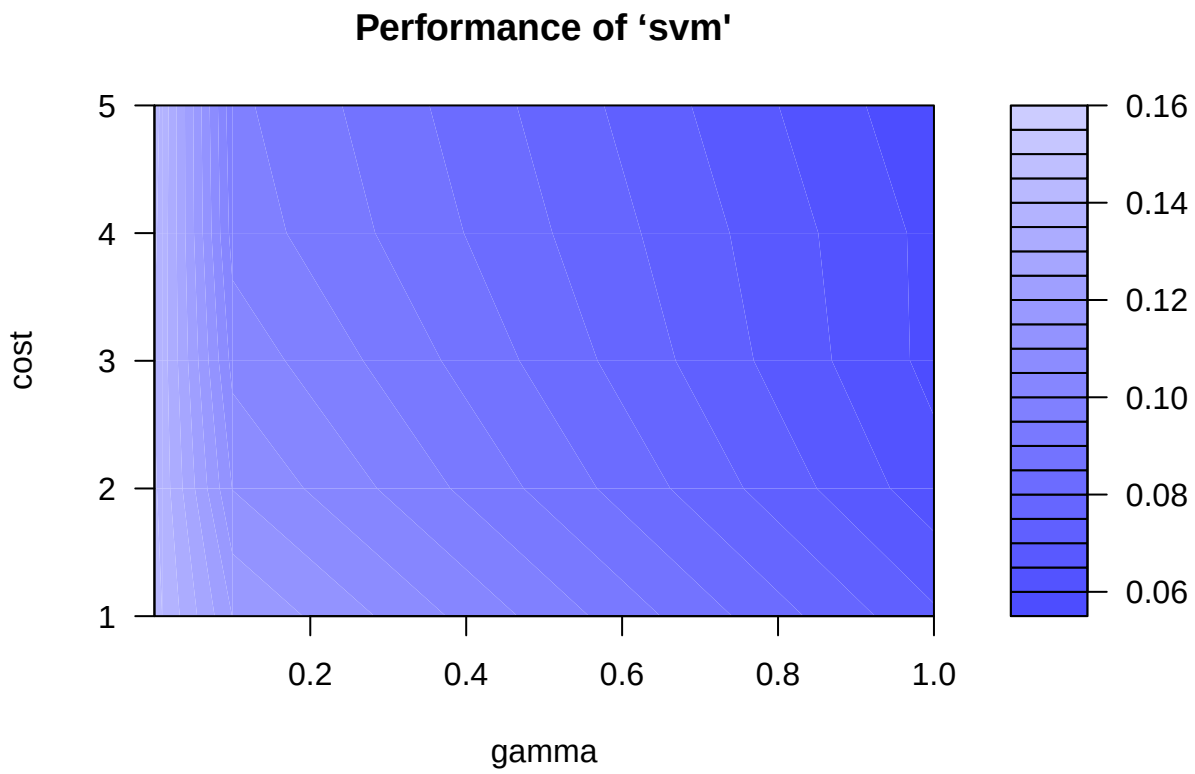
Finding Optimal SVM parameters with tuning and cross validation

```
# Testing various gamma & cost parameter combinations
gamma.params <- c(0.0001, 0.001, 0.01, 0.1, 1)
cost.params <- c(1:5)

# Setting up 10 fold cross validation
tc <- e1071::tune.control(cross = 10)

# Using tune.svm function to test gamma & cost with cross validation
set.seed(125)
tune.svm <- e1071::tune.svm(ALL.STAR ~., data = balanced, gamma = gamma.params,
                            cost = cost.params, tunecontrol = tc)

# View tune.svm summary
summary(tune.svm)
```

```
##
## Parameter tuning of 'svm':
##
## - sampling method: 10-fold cross validation
##
## - best parameters:
##  gamma cost
```

```
##      1    5
##
## - best performance: 0.0561086
##
## - Detailed performance results:
##    gamma cost      error dispersion
## 1  1e-04    1 0.15655761 0.03357813
## 2  1e-03    1 0.14179255 0.03139675
## 3  1e-02    1 0.14001740 0.03186388
## 4  1e-01    1 0.11993387 0.02607168
## 5  1e+00    1 0.07088061 0.01819472
## 6  1e-04    2 0.14769927 0.03447163
## 7  1e-03    2 0.14060216 0.02707875
## 8  1e-02    2 0.13824922 0.02703481
## 9  1e-01    2 0.10987818 0.02744837
## 10 1e+00    2 0.06201183 0.01597574
## 11 1e-04    3 0.15006613 0.03463479
## 12 1e-03    3 0.13941873 0.02669147
## 13 1e-02    3 0.13765750 0.02799820
## 14 1e-01    3 0.10337626 0.02771072
## 15 1e+00    3 0.05846850 0.01188081
## 16 1e-04    4 0.14592412 0.02935507
## 17 1e-03    4 0.14059868 0.02705585
## 18 1e-02    4 0.13824922 0.02857390
## 19 1e-01    4 0.09806126 0.03101777
## 20 1e+00    4 0.05847546 0.01225270
## 21 1e-04    5 0.14533589 0.02912591
## 22 1e-03    5 0.14178211 0.02680705
## 23 1e-02    5 0.13884441 0.02887189
## 24 1e-01    5 0.09628959 0.03209900
## 25 1e+00    5 0.05610860 0.01248215
```

```r
# View tune.svm plot
plot(tune.svm)
```



The best parameters that were found by tuning are gamma = 1 and cost = 4. I can just saved the best model from the tune.svm object by using $best.model, and I perform predictions with it below.s

Running SVM classifier with optimal parameters

```
# Extracting best model from tune.svm
svm.best.fit <- tune.svm$best.model

# SVM predictions
svm.predictions <- predict(svm.best.fit, all.star.norm.test[-ncol(all.star.norm.test)],
                           type = "response")

# View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.norm.test$ALL.STAR, svm.predictions,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
```

```
##
## Total Observations in Table:  256
##
##
##               | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       203 |         9 |       212 |
##              |     0.793 |     0.035 |           |
## -------------|-----------|-----------|-----------|
##            1 |         0 |        44 |        44 |
##              |     0.000 |     0.172 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       203 |        53 |       256 |
## -------------|-----------|-----------|-----------|
##
##
```

When using SMOTE the classifier does an incredible job making sure that there are no false negatives greatly increasing our recall. This however, may be a little suspect for our validation data set.

Gather SVM Stats for Comparison against all Models

```
# (True positive + True negative)/Total
(accuracy.svm <- (35 + 211)/256)
```

```
## [1] 0.9609375
```

```
# True positive / (True positive + False positive)
(precision.svm <- 35/(35 + 10))
```

```
## [1] 0.7777778
```

```
# True positive / (True positive + False negative)
(recall.svm <- 35/(35 + 0))
```

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

```
# 2 * (precision/recall) / (precision + recall)
(f1.svm <- (2 *precision.svm * recall.svm) / (precision.svm + recall.svm))
```

```
## [1] 0.875
```

Random Forest (Bagged Decision Trees)

Random Forest is an ideal model for this data set because decision trees work well with binary classification. Additionally, they lend themselves very well to bagging, which was done below to make a Random Forest of decision trees from the rpart tree algorith. The ipred library was used to set a bag of 100 trees from which a Random Forest classifier was made.

Using Smote on Train Set

```
smote.train <- smotefamily::SMOTE(all.star.train[-9], all.star.train$ALL.STAR,
                                  K = 5)
smote.train <- smote.train$data
colnames(smote.train)[9] <- "ALL.STAR"
smote.train$ALL.STAR <- as.factor(as.numeric(smote.train$ALL.STAR))
```

Here I am using smote again except I am doing it only on the training data set. This is because when I did it on the full data set (training + test) it would just predict everything 100% correct since it is a decision tree and has seen all of the predictions. This is different from before where I used the full (without only validation) set since cross validation uses that full data set.

Random Forest using Rpart & Ipred

```
# Setting seed for Reproducibility
set.seed(200)

# Using ipred bagging function for bagging
# number of bags set to 100
# control is done with rpart decision trees
rf.bag.fit <- ipred::bagging(
  formula = ALL.STAR ~ .,
  data = smote.train,
  nbagg = 100,
  coob = TRUE,
  control = rpart.control(minsplit = 2, cp = 0)
)
```

Testing Random Forest Model

```
# Storing Random Forest predictions
rf.bag.pred <- predict(rf.bag.fit, all.star.test[-ncol(all.star.test)],
                       type = "class")

# View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.test$ALL.STAR, rf.bag.pred,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
```

```
## Total Observations in Table:  256
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       188 |        24 |       212 |
##              |     0.734 |     0.094 |           |
## -------------|-----------|-----------|-----------|
##            1 |        10 |        34 |        44 |
##              |     0.039 |     0.133 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       198 |        58 |       256 |
## -------------|-----------|-----------|-----------|
##
##
```

Gather Random Forest Stats for Comparison against all Models

```
# (True positive + True negative)/Total
(accuracy.rf <- (29 + 198)/256)
```

```
## [1] 0.8867188
```

```
# True positive / (True positive + False positive)
(precision.rf <- 29/(29 + 23))
```

```
## [1] 0.5576923
```

```
# True positive / (True positive + False negative)
(recall.rf <- 29/(29 + 6))
```

```
## [1] 0.8285714
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.rf <-(2 *precision.rf * recall.rf) / (precision.rf + recall.rf))
```

```
## [1] 0.6666667
```

Ensemble Model

My ensemble model simply finds the mode for a certain prediction and returns that. It is a functon that takes in data and the normalized version of that data that we can run on the models.

```
ensemble <- function(data, norm_data) {
  # Takes in a testing data set and normalized testing data set (all.star.test, all.star.norm.test)
```

```
  ### Run Model Predictions
  # GLM
  significant.cols <- c("raptor_offense_x", "war_reg_season_x",
                        "predator_offense_x", "pace_impact_x")

  glm.predictions <- predict(glm.best.fit, data[ ,significant.cols],
                             type = "raw")

  ## SVM
  svm.predictions <- predict(svm.best.fit, norm_data[-ncol(norm_data)],
                             type = "response")

  ## Random Forest
  rf.bag.predictions <- predict(rf.bag.fit, data[-ncol(data)], type = "class")

  ### Store All Predictions & Find mode
  all.predictions <- cbind.data.frame(glm.predictions, svm.predictions,
                                      rf.bag.predictions)

  ensemble.pred <- apply(all.predictions, 1, modeest::mfv)

  return(as.numeric(ensemble.pred))
}
```

Testing Ensemble Model

```
# Getting ensemble model predictions
ensemble.predictions <- ensemble(all.star.test, all.star.norm.test)
```

```
## Registered S3 method overwritten by 'rmutil':
##   method         from
##   plot.residuals psych
```

```
#View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.test$ALL.STAR, ensemble.predictions,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  256
##
##
##              | Predicted All Star
```

```
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       199 |        13 |       212 |
##              |     0.777 |     0.051 |           |
## -------------|-----------|-----------|-----------|
##            1 |        10 |        34 |        44 |
##              |     0.039 |     0.133 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       209 |        47 |       256 |
## -------------|-----------|-----------|-----------|
##
##
```

Gather Ensemble Stats for Comparison against all Models

```
# (True positive + True negative)/Total
(accuracy.ensemble <- (30 + 209)/256)
```

```
## [1] 0.9335938
```

```
# True positive / (True positive + False positive)
(precision.ensemble <- 30/(30 + 12))
```

```
## [1] 0.7142857
```

```
# True positive / (True positive + False negative)
(recall.ensemble <- 30/(30 + 5))
```

```
## [1] 0.8571429
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.ensemble <- (2 *precision.ensemble * recall.ensemble) /
    (precision.ensemble + recall.ensemble))
```

```
## [1] 0.7792208
```

Comparing All Models with Test Set

```
# Gathering all stats into individual vectors
accuracy <- c(accuracy.glm, accuracy.rf, accuracy.svm, accuracy.ensemble)
precision <- c(precision.glm, precision.rf, precision.svm, precision.ensemble)
recall <- c(recall.glm, recall.rf, recall.svm, recall.ensemble)
f1.score <- c(f1.glm, f1.rf, f1.svm, f1.ensemble)

# Binding stats into data frame for comparison
model.comparison.df <- rbind.data.frame(accuracy, precision, recall, f1.score)
```

```
# Naming columns & rows for data frame
colnames(model.comparison.df) <- c("GLM", "RF", "SVM", "Ensemble")
rownames(model.comparison.df) <- c("Accuracy", "Precision", "Recall", "F1 Score")

model.comparison.df
```

```
##                 GLM        RF       SVM  Ensemble
## Accuracy  0.9296875 0.8867188 0.9609375 0.9335938
## Precision 0.7575758 0.5576923 0.7777778 0.7142857
## Recall    0.7142857 0.8285714 1.0000000 0.8571429
## F1 Score  0.7352941 0.6666667 0.8750000 0.7792208
```

All of the models perform quite well on the test data set, particularly SVM has perfect recall causing it to have a very high F1 score. It is good to see that the F1 score for the Ensemble is high, but it will be much more important to see how the models perform on the validation data.

Model Selection with Validation set

Validation Logistic Regression

```
significant.cols <- c("raptor_offense_x", "war_reg_season_x",
                      "predator_offense_x", "pace_impact_x")

log.pred.val <- predict(glm.best.fit, all.star.validate[, significant.cols],
                        type = "raw")

#View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.validate$ALL.STAR, log.pred.val,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  257
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       213 |         7 |       220 |
##              |     0.829 |     0.027 |           |
## -------------|-----------|-----------|-----------|
##            1 |        15 |        22 |        37 |
```

```
##              |     0.058 |     0.086 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       228 |        29 |       257 |
## -------------|-----------|-----------|-----------|
##
##
```

Validation Stats GLM

```
# (True positive + True negative)/Total
(accuracy.val.glm <- (22 + 209)/257)
```

```
## [1] 0.8988327
```

```
# True positive / (True positive + False positive)
(precision.val.glm <- 22/(22 + 3))
```

```
## [1] 0.88
```

```
# True positive / (True positive + False negative)
(recall.val.glm <- 22/(22 + 23))
```

```
## [1] 0.4888889
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.val.glm <- (2 *precision.val.glm * recall.val.glm) /
    (precision.val.glm + recall.val.glm))
```

```
## [1] 0.6285714
```

Validation SVM

```
svm.pred.val <- predict(svm.best.fit, all.star.norm.validate[-ncol(all.star.norm.validate)],
                        type = "response")

#View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.norm.validate$ALL.STAR, svm.pred.val,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
```

```
##
##
## Total Observations in Table:  257
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       194 |        26 |       220 |
##              |     0.755 |     0.101 |           |
## -------------|-----------|-----------|-----------|
##            1 |        17 |        20 |        37 |
##              |     0.066 |     0.078 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       211 |        46 |       257 |
## -------------|-----------|-----------|-----------|
##
##
```

Validation Stats SVM

```
# (True positive + True negative)/Total
(accuracy.val.svm <- (25+ 188)/257)
```

```
## [1] 0.8287938
```

```
# True positive / (True positive + False positive)
(precision.val.svm <- 25/(25 + 24))
```

```
## [1] 0.5102041
```

```
# True positive / (True positive + False negative)
(recall.val.svm <- 25/(25 + 20))
```

```
## [1] 0.5555556
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.val.svm <- (2 *precision.val.svm * recall.val.svm) /
    (precision.val.svm + recall.val.svm))
```

```
## [1] 0.5319149
```

Validation Random Forest

```
rf.pred.val <- predict(rf.bag.fit, all.star.validate[-ncol(all.star.validate)], type = "class")

#View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.validate$ALL.STAR, rf.pred.val,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  257
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       184 |        36 |       220 |
##              |     0.716 |     0.140 |           |
## -------------|-----------|-----------|-----------|
##            1 |        11 |        26 |        37 |
##              |     0.043 |     0.101 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       195 |        62 |       257 |
## -------------|-----------|-----------|-----------|
##
##
```

Validation Stats RF

```
# (True positive + True negative)/Total
(accuracy.val.rf <- (32+ 187)/257)
```

```
## [1] 0.8521401
```

```
# True positive / (True positive + False positive)
(precision.val.rf <- 32/(32 + 25))
```

```
## [1] 0.5614035
```

```
# True positive / (True positive + False negative)
(recall.val.rf <- 32/(32 + 13))
```

```
## [1] 0.7111111
```

```
# 2 * (precision/recall) / (precision + recall)
(f1.val.rf <- (2 *precision.val.rf * recall.val.rf) /
    (precision.val.rf + recall.val.rf))
```

```
## [1] 0.627451
```

Validation Ensemble Model

```
ensemble.val <- ensemble(all.star.validate, all.star.norm.validate)

#View predictions with gmodels confusion matrix
gmodels::CrossTable(all.star.validate$ALL.STAR, ensemble.val,
                    prop.chisq = F, prop.c = F, prop.r = F,
                    dnn = c("All Star", "Predicted All Star"))
```

```
##
##
##    Cell Contents
## |-------------------------|
## |                       N |
## |         N / Table Total |
## |-------------------------|
##
##
## Total Observations in Table:  257
##
##
##              | Predicted All Star
##     All Star |         0 |         1 | Row Total |
## -------------|-----------|-----------|-----------|
##            0 |       197 |        23 |       220 |
##              |     0.767 |     0.089 |           |
## -------------|-----------|-----------|-----------|
##            1 |        13 |        24 |        37 |
##              |     0.051 |     0.093 |           |
## -------------|-----------|-----------|-----------|
## Column Total |       210 |        47 |       257 |
## -------------|-----------|-----------|-----------|
##
##
```

Validation Stats Ensemble

```
# (True positive + True negative)/Total
(accuracy.val.ens <- (26 + 197)/257)
```

```
## [1] 0.8677043
```

```
# True positive / (True positive + False positive)
(precision.val.ens <- 26/(26 + 15))
```

```
## [1] 0.6341463
```

```r
# True positive / (True positive + False negative)
(recall.val.ens <- 26/(26 + 19))
```

```
## [1] 0.5777778
```

```r
# 2 * (precision/recall) / (precision + recall)
(f1.val.ens <- (2 *precision.val.ens * recall.val.ens) /
    (precision.val.ens + recall.val.ens))
```

```
## [1] 0.6046512
```

Comparing All Models with Validation Set

```r
# Gathering all stats into individual vectors
accuracy.val <- c(accuracy.val.glm, accuracy.val.rf, accuracy.val.svm, accuracy.val.ens)
precision.val <- c(precision.val.glm, precision.val.rf, precision.val.svm, precision.val.ens)
recall.val <- c(recall.val.glm, recall.val.rf, recall.val.svm, recall.val.ens)
f1.score.val <- c(f1.val.glm, f1.val.rf, f1.val.svm, f1.val.ens)

# Binding stats into data frame for comparison
validation.comparison.df <- rbind.data.frame(accuracy.val, precision.val, recall.val, f1.score.val)

# Naming columns & rows for data frame
colnames(validation.comparison.df) <- c("GLM", "RF", "SVM", "Ensemble")
rownames(validation.comparison.df) <- c("Accuracy", "Precision", "Recall", "F1 Score")

validation.comparison.df
```

```
##                 GLM        RF       SVM  Ensemble
## Accuracy  0.8988327 0.8521401 0.8287938 0.8677043
## Precision 0.8800000 0.5614035 0.5102041 0.6341463
## Recall    0.4888889 0.7111111 0.5555556 0.5777778
## F1 Score  0.6285714 0.6274510 0.5319149 0.6046512
```

Conclusion

The goal of this project was to be able to see whether NBA All Stars could be effectively predicted by advanced analytics. This is an interesting question to answer for the NBA community because there has been a great increase in how advanced analytics are viewed with respect to signing players to new contracts. Additionally, understanding what players just missed the cut, or where more deserving than actual all stars is important because it may help teams find players that are undervalued for what they are actually offering. To test this I found a data set on Kaggle that had interesting advanced player metrics. Once downloaded I then had to manually impute the all stars into the data set. I then created 3 different models for predicting All Stars, Logistic Regression, Support Vector Machine and Decision Trees with Bagging (Random Forest). Since the prediction variable was highly imbalanced (85:15), I used the Smote Library where necessary to give models training data that was much more balanced with non all stars and all stars. I left out a validation data set at the end on which none of the models had seen in any way at any point.

As this is an imbalanced classification the accuracy of the model is not very important. The Logistic Regression model performed by far the best with respect to precision at 88% precision on the validation data set. The model with the best recall was the Random Forest model at around 71%. Although the Support Vector Machine seemed to perform incredibly on the test set, its performed was not matched on the validation set. Lastly, the ensemble combines all 3 and finds the mode, and you can see that the precision and recall are in between the high values shown by GLM & RF for Precision and Recall respectively.

Which model an NBA analyst might chose to use is now up to them. If they were very interested in predicting All Stars and were looking to minimize the amount of False positives, the Logistic Regression model would be the way to go. This may be useful for them, if they want to test players and see if they truly definitely belong as all-stars. On the other hand, the high Recall of Random Forest can be very useful as well. This may be useful for an analyst if they are more interested in fringe all stars and want to see what false positives the model predicted in order to find a hidden gem in the rough. Otherwise, for exploratory analysis it may be best to use the ensemble as it has decent precision and recall.