Exploring the Predictive Power of the Real Plus-Minus Stat of Small Forwards in the NBA

Introduction

The Rise of Advanced Metrics and RPM in the NBA

The NBA is a sport with over 70 years of history. With its longevity comes historical data; the first records of games and box scores are dated to its inaugural season in 1946. As the game evolved, methods for data collection, the data collected, and metrics for analysis have evolved, especially in the modern era of analytics. Advanced models have been developed over the spam of years of data, and certain applications of AWS are projecting statistics for a game, calculating the likelihood of a certain team reaching the finals, and even gauging how likely a shot is going to go in the basket based on the player's shooting percentage from that location, the amount of defense enforced on that shot, even using the arena being played in as a predictor.

All of this goes to show the following trend: as the amount of data we collect increases, the more models and simulations we are able to produce. In light of a pursuit to find a way to more effectively measure a player's comprehensive effect on the game- as people began to stray away from considering points scored as the discerning factor in a player's effectiveness- a former member of the Phoenix Suns named Jeremias Englemann developed a groundbreaking statistic: Regularized Adjusted Plus-Minus, or RAPM, also known as Real Plus-Minus, or RPM. RPM is an enhanced form of adjusted plus-minus that achieves two major goals: 1) it is a stronger measure of one's effect on offense and defense than basic BPM and 2) it is a better predictive tool of a person's contributions than BPM or APM, making it the most advanced choice for measuring and integrating a plus-minus statistic . ESPN has an article introducing the concept of RPM, and they briefly explain the method behind the model: "The RPM model sifts through more than 230,000 possessions each NBA season to tease apart the 'real' plus-minus effects attributable to each player."

The ultimate result of using an RPM model yields a number-usually single digits- that represents how many points a player contibutes and gives up over 100 possessions, which is the same basis on which Plus-Minus normally works. The higher the number, the higher the likelihood of this player being totally effective on the floor- either the player has a balanced positive effect offense and defense, or their offense salvages their poor defense or vice versa. RPM is calculated as the sum of 0RPM and DRPM, which again demonstrates how both phases are considered. We can expect a player with a high RPM to have both a positive 0RPM and positive DRPM, as at the professional level you could expect all players to be efficient in both phases of the game.

Putting Expectations to the Test: The Questions and the Data

In light of this ideal, we want to see if the a high RPM yields more success: wins, fans, and money. In other words: can we predict how well a player ranks with respect to how many wins they exceedingly contribute to, how much money and how many fans a player can make given their RPM statistics? In the current meta, commentators and analysts consider the league as predominantly offensive: more emphasis is supposedly put towards offense rather than defense, thus leading to higher point totals per player, shots per game, and higher end game scores. Supposedly, offense and scoring is what sells to the market, and that brings excitement to the fans and the league and the players get paid for it. A lockdown defender like Patrick Beverly likely isn't as well known as a scoring machine like Stephen Curry or James Harden, especially since scores are what make the highlights and are directly related to winning. In essence: we want to see if having a successful career and franchise in this league is still reliant on dominance in all phases of the game or if offense alone can yield wins, fans and money.

The base data we will be using is the NBA Real Plus-Minus for the 2018-2019 season dataset provided by Jeremias Engelmann and Steve Ilardi of ESPN. Because there are hundreds of NBA players and positions make a significant difference on pay and RPM, the focus will be on Small Forwards, as the annual stars, money makers, and MVP frontrunners (Paul George, LeBron, Kevin Durant, Kawhi Leonard) play this position. Reducing the complexity invites more exploration of the data. Furthermore, Paul George and Kawhi Leonard are annually considered the best defensive players in the game, thus it makes for an interesting comparison between 0RPM and DRPM since offense and defense will play as similarly large factors, as compared to positions like Center where it is heavily polarized that a player is either an offensive monster or a defensive wall.

The data is population data, as it is derived the complete set of NBA players, but SFs are filtered out. This is not considered a sample, as we will be using all SFs in this study. The data was collected throughout the NBA season, likely though analyzing player box scores and the times they appeared in the NBA play-by-play list that is generated each game that descriptively lists the scorer and defender on each possession.

The dataset is modified to include the reported salaries of each player on the list during that season and the number of all star votes the player received, as that is understood as the strongest universal gague of popularity in the league. The dataset is a csv file with the following columns:

NAME- (character) The Name of the player in "FirstName LastName" format

TEAM- (character) The Abbreviation of the organization the player ended with in the 2018-2019 season

GP- (numeric) The number of games played in the regular season (out of 82). Used to see if the played was significantly hindered by injury.

MPG- (numeric) The average minutes per game a player was on the court. Used to see if a player was a franchise player (ie. 32+ minutes), a rotation player, or a bench player

ORPM- (numeric) Offensive Real Plus-Minus- the RPM with respect to possessions where the player's team has the ball

DRPM- (numeric) Defensive Real Plus-Minus- the RPM with respect to possessions where the player's team does not have the ball

RPM- (numeric) The sum of ORPM and DRPM; the main predictor in this study

RRANK- (numeric) An alternative identifier; the rank of a player with respect to RPM

SALARY- (numeric) Contractual salary the player made in the 2018-2019 season in dollars

VOTES- (numeric) Number of votes received for the 2019 All-Star game; popularity measure

PLAYOFFS- (factor) Denotes if the player was on a playoff team; used as a success/winningness factor

HIGH.SALARY- (factor) Denotes if the player had a salary over the NBA average ($7.7M)

POPULAR- (factor) Denotes of the player had over 900,000 All Star Votes

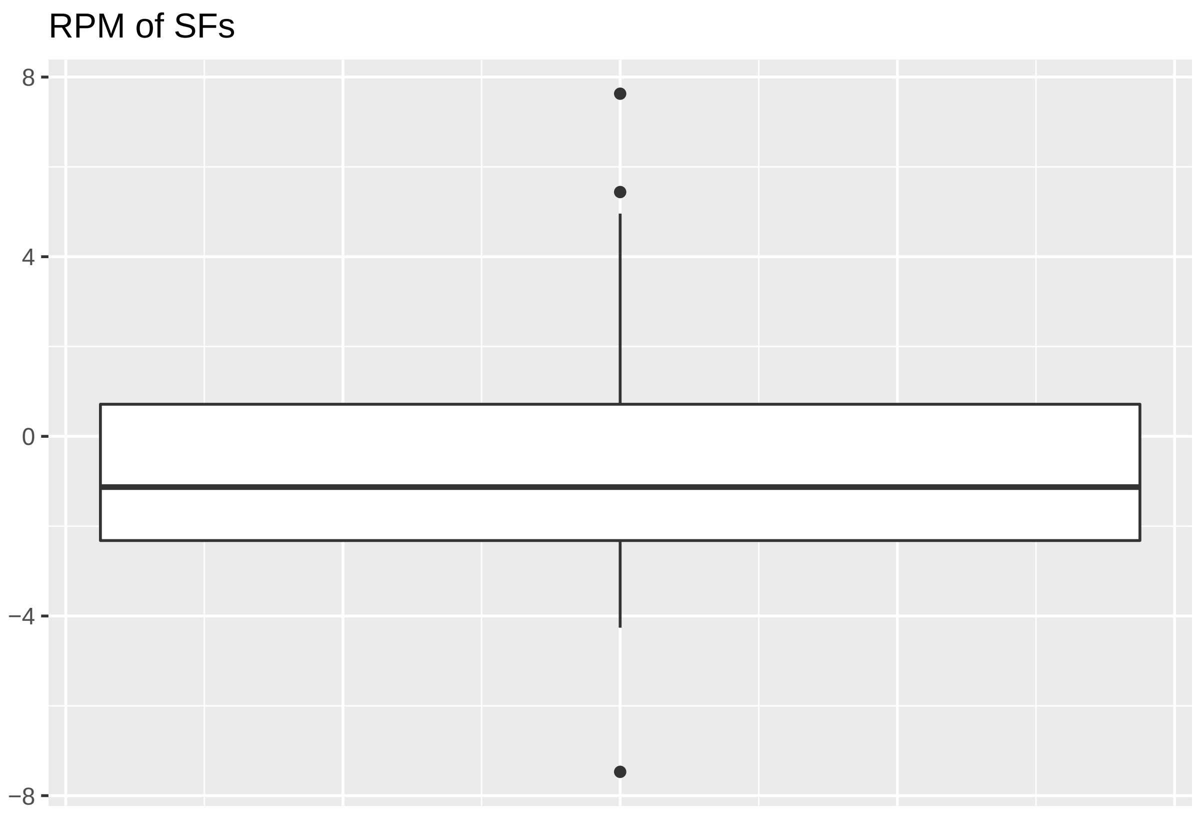
To study this, we will explore various models and test their effectivity in order to have a stronger grasp of the predictive power of RPM and the limits and strengths of certain modeling executions on a dataset like this.

Methods/Analysis

Looking at the data, there seems to be no problem areas. Upon loading the csv file, the `colClasses` argument was used to ensure that all the values are the correct data type. A quick check of rpmdata$SALARY indicates that all players have values in Salary, which was the only concerning spot for this dataset, as 0.0 in any other category is likely correct/sensible (i.e, a benchwarmer getting 0 All-Star votes). One could argue that the incredibly low All-Star votes (i.e, 0 for Glenn Robinson III or 2744 for Wes Iwundu) would skew the data, but they should be kept in because they also reflect the lower end of the basic model we hypothesize: players with low RPMS will tend to make less money or be less popular.

Another consideration in NBA stats is the nature of a player's schedule and how important their specific contribution was in their situation. For instance, an NBA team desperate to make a high seed in the playoffs (like OKC) or a team trying to make the playoffs (like the Lakers) is more likely to have high output, superstar players playing longer minutes and thus collecting more stats as opposed to a rookie on a hopeless team (namely JaKarr Sampson of the Chicago Bulls, playing only in 4 games and having the lowest RPG out of all NBA Small Forwards). It's important to consider that the latter example is particularly rare, as the situation is rare, and thus the statistical anomaly may not entirely represent the capabilities of the player. We will remember this in our analysis later.

This being out of the way, we can look for some trends. Here is a boxplot of the RPM of all SFs in the NBA in the 2018-2019 season:

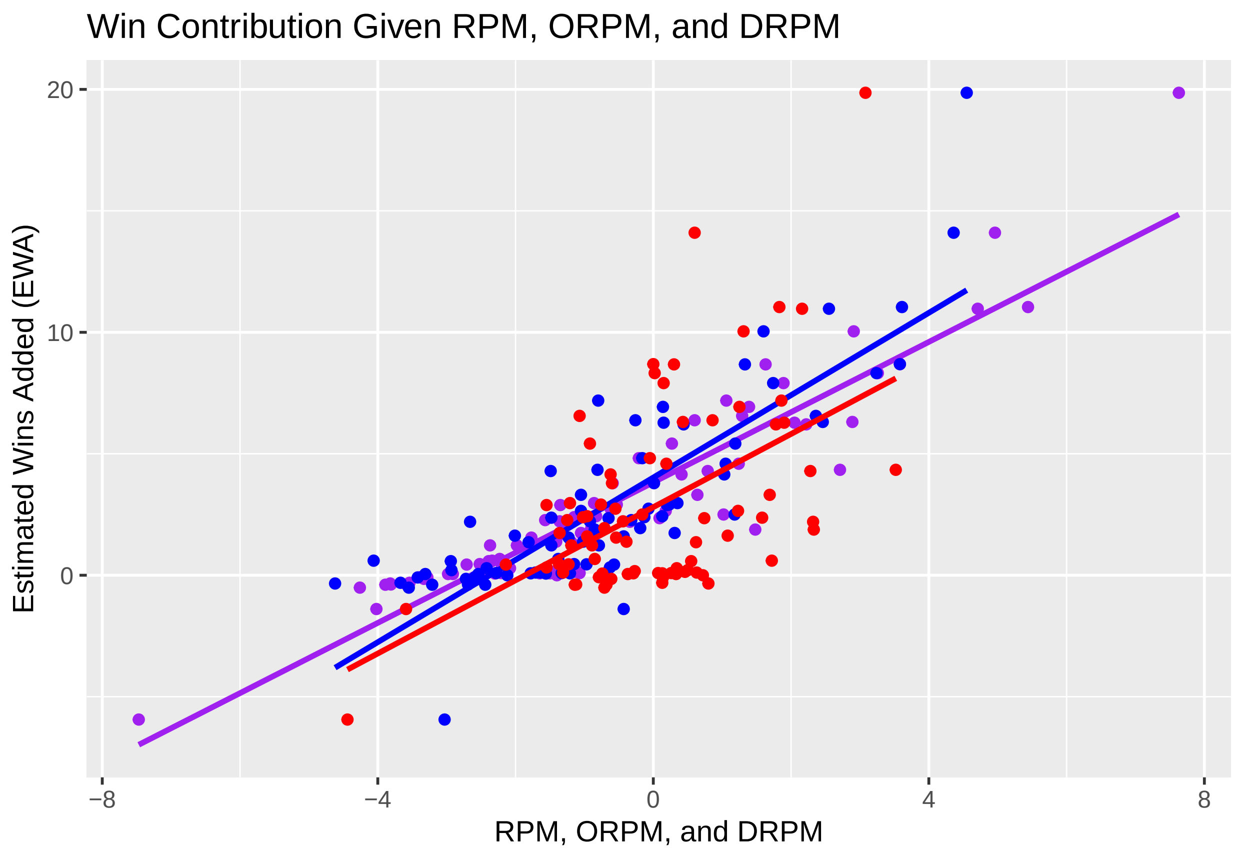


```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
## -7.4700 -2.3200 -1.1300 -0.6848  0.7150  7.6300
```

We can see that the data is mostly contrained in the box plot, so not too many anomalies exist. We can see that almost all the data lies between -4.0 and 5, with 50% of it existing under -1.13 and 50% of it existing over. The IQR is approximately 3.0. This means we have quite a concentration of players that have RPMs from -2.3 to 0.7, and the average for all SFs is -0.68. All of this tells us that we can expect a given Small Forward in the NBA this season to have had a slightly negative RPM. The basic interpretation of this is that for 75% of all SFs in the league, they either barely made a positive contribution to the team's overall point differential, or their rotation on the court resulted in the opposing team outscoring them more often than not. This sets not only a low expectation for SFs, but also puts high regard on the starters and true MVP performers. The two positive outliers on the list are Paul George and LeBron James: two household names and perrenial All Stars. They are highly paid, highly popular, and evidently have high performance. The bottom outlier is our friend JaKarr Sampson from the Bulls, who was in a much different situation than PG and

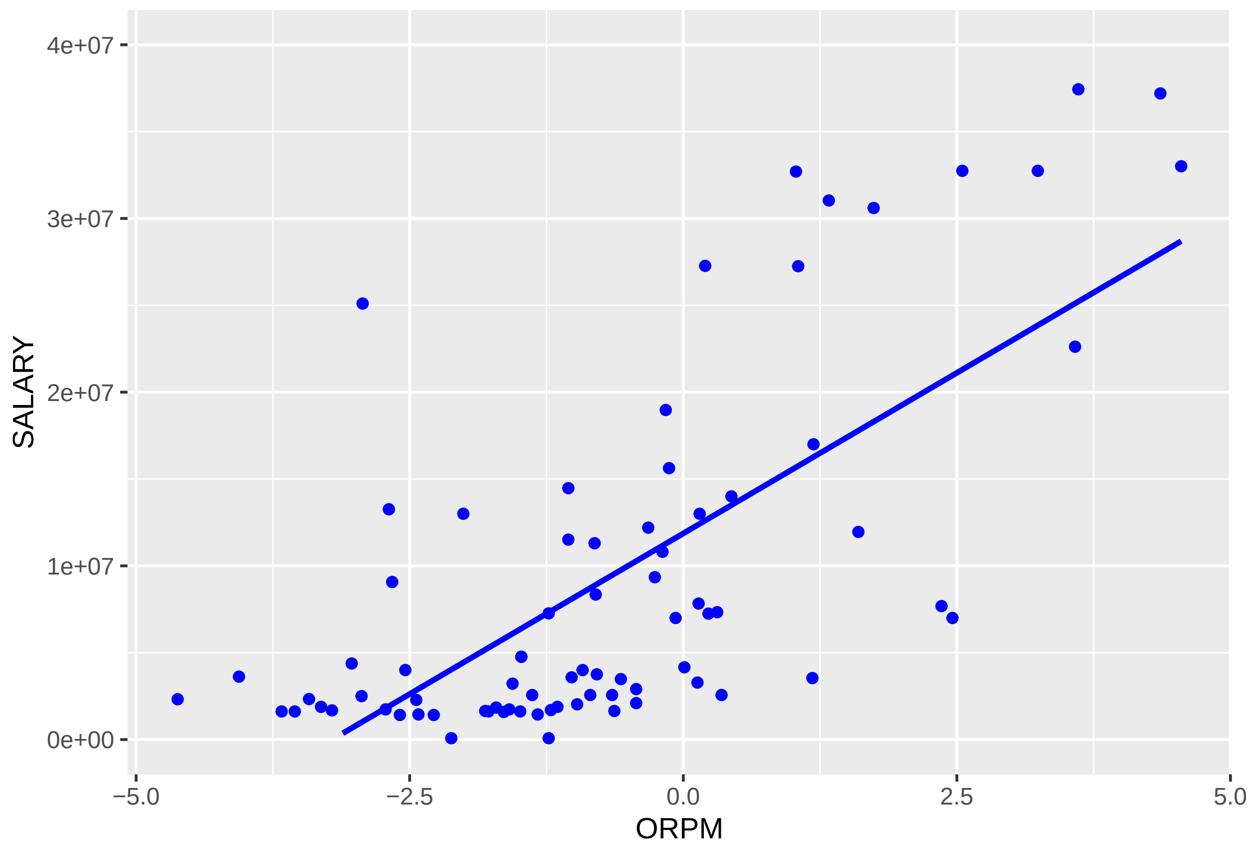
LBJ. For the most part, our expectations are set for the distribution of RPM. Let's look at some more graphs!

We want to quickly see what kind of effect RPM, ORPM, and DRPM have on how useful a player's overall contributions are to his team compared to if another player was in their position. Let's see if a graph between these 3 variables and EWA can point out a significant avenue:



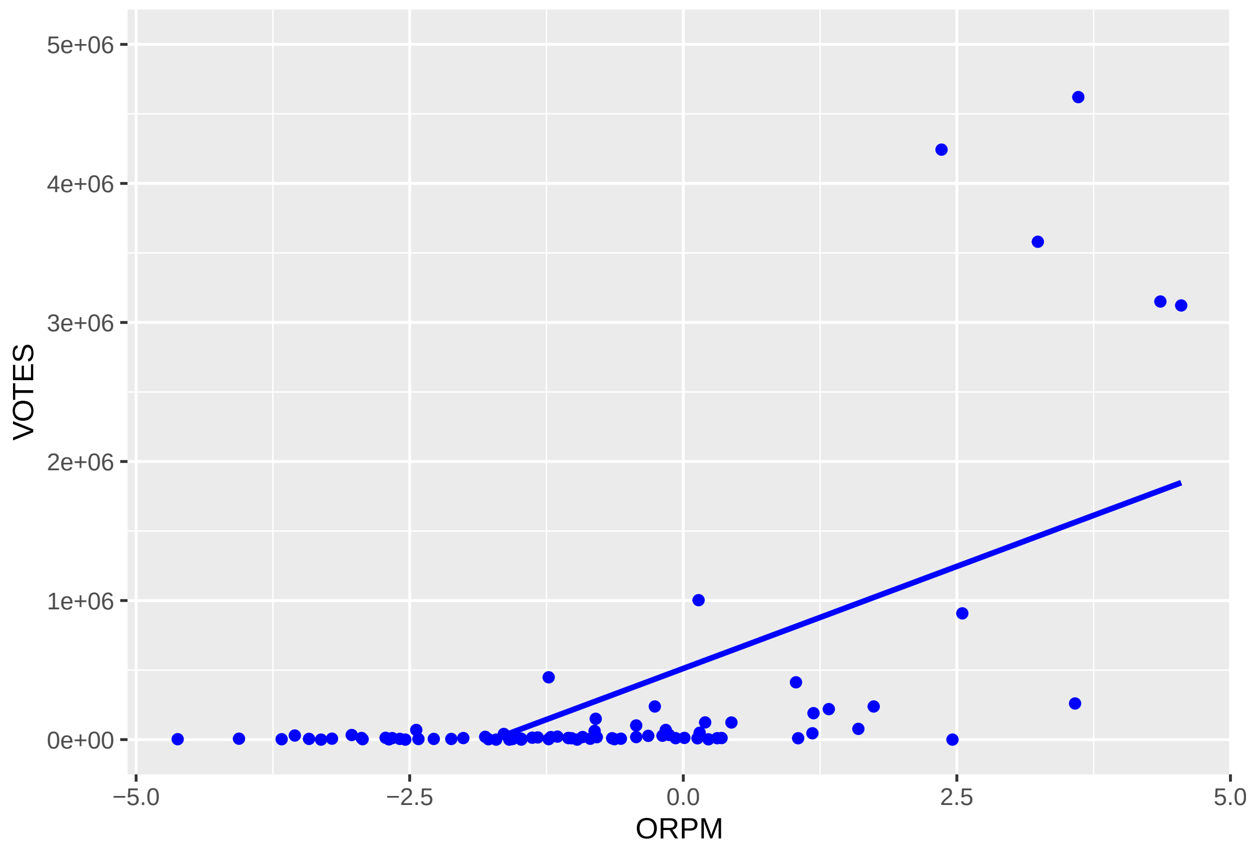
We can quickly recognize something: ORPM (blue) has a higher slope than RPM (purple) and DRPM (red). Can we conclude that ORPM is the best indicator of winning contribution? Well inherently, a higher ORPM means you're scoring more than your opponent can defend you, and the objective of the game is to have a higher final score than your opponent. Arguably, a higher ORPM being a more weighted factor in your EWA stat is obvious, but it feels like this one dimensional analysis is again missing those other critical biases, like minutes played. Let's see if we can establish a link between money earned and ORPM as we would expect it to:

```
ggplot(rpmdata)+
  geom_point(aes(x=ORPM,y=SALARY),color="blue")+
  geom_smooth(aes(x=ORPM,y=SALARY),color="blue",method=lm,se=F)+
  scale_y_continuous(limits=c(0,40000000))
```



Clearly, we can see a linear model does not fit the data well. There is a lot of deviation from the regression line, definitively known as having low accuracy. Do we have better luck with popularity?

```
ggplot(rpmdata)+
  geom_point(aes(x=ORPM,y=VOTES),color="blue")+
  geom_smooth(aes(x=ORPM,y=VOTES),color="blue",method=lm,se=F)+
  scale_y_continuous(limits=c(0,5000000))
```



We see even less of an association. But why is that, when logically, all 3- EWA, Popularity, and Salary- should go together? The highest paid, most popular, winningest datapoint should be all in association, so what does that say about linear modeling in this sense? It could be dissuaded due to the various confounding variables.

The original question being explored sought to prove or disprove the ability of a player's RPM metric to predict their rank in popularity, salary, and wins added. Furthermore, we wanted to see if ORPM- or offensive contribution and play- did an even better job. We can clearly see how single dimensional modeling won't cut it, so let's try a method that tries to limit the biases in our data to see if we can successfully build a model to predict these things.

Model Discussion

Looking at KNN

We already previously looked through the data. We've run `anyNa(rpmdata)` and we appear to be safe. Let's continue to partitioning the dataset into a training and testing set:

```
sampsize <- floor(.8*nrow(rpmdata))
set.seed(3080)
train_indeces <- sample(1:nrow(rpmdata), size=sampsize)
trainer <- rpmdata[train_indeces,]
tester <- rpmdata[-train_indeces,]
```

As indicated by the subheading, the first model we'll be looking at is KNN. Because we are dealing with a lot of categorical data, KNN seems to be a strong choice. We also aren't considering real-time data. Caret also optimizes our k between k=5,7 and 9. As a side note, we must not forget to preProcess our data, especially with how relatively small our sample size is. Let's look at the initial performance, comparing RPM to a player's popularity metric:

```
set.seed(3080)
train_knn_pop <- train(POPULAR~RPM, #Testing "Popular" (Votes > 500000)
                   data=trainer,
                   trControl= trainControl(method = "repeatedcv",
                                           number = 10, repeats = 3),
                   method="knn",
                   preProcess = c("center", "scale"))
y_hat_knn_pop <- predict(train_knn_pop, tester)
confusionMatrix(data=y_hat_knn_pop, reference = tester$POPULAR)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 14  2
##          1  0  0
##
##                Accuracy : 0.875
##                  95% CI : (0.6165, 0.9845)
##     No Information Rate : 0.875
##     P-Value [Acc > NIR] : 0.6771
##
##                   Kappa : 0
##
##  Mcnemar's Test P-Value : 0.4795
##
##             Sensitivity : 1.000
##             Specificity : 0.000
##          Pos Pred Value : 0.875
##          Neg Pred Value :   NaN
##              Prevalence : 0.875
##          Detection Rate : 0.875
##    Detection Prevalence : 1.000
##       Balanced Accuracy : 0.500
##
##        'Positive' Class : 0
##
```

Remarkable! We have a high accuracy rate and a "positive" Accuracy:NIR ratio! The only concern would be specificity, which is at .5. That could go either way with a larger sample set. But for the most part, this is a satisfactory model for popularity. We'll discuss this more later.

Let's move on to the case for the relationship between High Salary and RPM:

```
set.seed(3080)
train_knn_sal <- train(HIGH.SALARY~RPM, #Testing "High Salary" (>$7.7M)
                       data=trainer,
                       trControl= trainControl(method = "repeatedcv",
                                               number = 10, repeats= 3),
                       method="knn",
                       preProcess = c("center", "scale"))
y_hat_knn_sal <- predict(train_knn_sal, tester)
confusionMatrix(data=y_hat_knn_sal, reference = tester$POPULAR)
```

```
## Confusion Matrix and Statistics
```

```
##
##           Reference
## Prediction  0  1
##          0 11  0
##          1  3  2
##
##                Accuracy : 0.8125
##                  95% CI : (0.5435, 0.9595)
##     No Information Rate : 0.875
##     P-Value [Acc > NIR] : 0.8698
##
##                   Kappa : 0.4783
##
##  Mcnemar's Test P-Value : 0.2482
##
##             Sensitivity : 0.7857
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.4000
##              Prevalence : 0.8750
##          Detection Rate : 0.6875
##    Detection Prevalence : 0.6875
##       Balanced Accuracy : 0.8929
##
##        'Positive' Class : 0
##
```

83% accuracy and 78.6% balanced accuracy is a mediocre result. We have a kappa of .25 which is pretty poor, and a shocking 1.0 specificity, which means this model might not work too well pertaining to salary. Again, we'll remark on this later. Let's move on to winningness:

```
set.seed(3080)
train_knn_win <- train(PLAYOFFS~RPM, #Testing Winningness (Made Playoffs)
                       data=trainer,
                       trControl= trainControl(method = "repeatedcv",
                                               number = 10, repeats = 3),
                      method="knn",
                      preProcess = c("center", "scale"))
y_hat_knn_win <- predict(train_knn_win, tester)
confusionMatrix(data=y_hat_knn_win, reference = tester$POPULAR)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction  0  1
##          0 10  0
##          1  4  2
##
##                Accuracy : 0.75
##                  95% CI : (0.4762, 0.9273)
##     No Information Rate : 0.875
##     P-Value [Acc > NIR] : 0.9593
##
##                   Kappa : 0.3846
##
##  Mcnemar's Test P-Value : 0.1336
##
##             Sensitivity : 0.7143
##             Specificity : 1.0000
##          Pos Pred Value : 1.0000
##          Neg Pred Value : 0.3333
##              Prevalence : 0.8750
##          Detection Rate : 0.6250
##    Detection Prevalence : 0.6250
##       Balanced Accuracy : 0.8571
##
##        'Positive' Class : 0
##
```

And this result is the poorest out of the 3, at 68.8% accuracy and a "negative" Accuracy:NIR ratio. Again, it reached 1.0 specificity. Maybe the quickest consideration for why this was so poor is because playoffs and wins in general are a team metric, and it's difficult to individualize. Furthermore, we can read the results of the confusion matrix and see that the true distribution in the testing set is 14:2 in favor of a 0 state (not being in the playoffs). It's easy to see why a categorization-based algorithm failed with quite obscure data.

In light of this, let's explore the observation we had earlier- maybe ORPM would improve the strength of this algorithm?

```
set.seed(3080)
train_knn_orwin <- train(PLAYOFFS~ORPM, #Testing Winningness with ORPM
                       data=trainer,
                       trControl= trainControl(method = "repeatedcv",
                                               number = 10, repeats = 3),
                       method="knn",
                       preProcess = c("center", "scale"))
train_knn_orwin$results
```

```
##   k  Accuracy      Kappa AccuracySD   KappaSD
## 1 5 0.5387302 0.05701833  0.1800979 0.3568778
## 2 7 0.5823810 0.13326589  0.1920345 0.3892923
## 3 9 0.6077778 0.18705377  0.1810538 0.3622015
```

```
y_hat_knn_orwin <- predict(train_knn_orwin, tester)
confusionMatrix(data=y_hat_knn_orwin, reference = tester$PLAYOFFS)
```

```
## Confusion Matrix and Statistics
##
##           Reference
## Prediction 0 1
##          0 7 6
##          1 1 2
##
##                Accuracy : 0.5625
##                  95% CI : (0.2988, 0.8025)
##     No Information Rate : 0.5
##     P-Value [Acc > NIR] : 0.4018
##
##                   Kappa : 0.125
```

```
##
##  Mcnemar's Test P-Value : 0.1306
##
##             Sensitivity : 0.8750
##             Specificity : 0.2500
##          Pos Pred Value : 0.5385
##          Neg Pred Value : 0.6667
##              Prevalence : 0.5000
##          Detection Rate : 0.4375
##    Detection Prevalence : 0.8125
##       Balanced Accuracy : 0.5625
##
##        'Positive' Class : 0
##
```

We can see this performed shockingly worse. An accuracy of .5 is no better than a coin flip. The specificity was lower, but that's because it interestingly guessed the "0" condition 7 times incorrectly instead of guessing the"1" condition incorrectly.

Consulting other models

Because of KNN performing poorly especially on popularity, let's explore and see if any other models can perform better:

We can see that combined, none of these models performed as well as the knn setup we had earlier. Let's exmaine the individual observations:

```
list<-c(fits$glm$results$Accuracy,
     fits$lda$results$Accuracy,
     fits$naive_bayes$results$Accuracy,
     fits$svmLinear$results$Accuracy,
     fits$knn$results$Accuracy,
     fits$gamLoess$results$Accuracy,
     fits$multinom$results$Accuracy,
     fits$qda$results$Accuracy,
     fits$rf$results$Accuracy,
     fits$adaboost$results$Accuracy)
```

```
list
```

```
##  [1] 0.6470965 0.6205314 0.6482626 0.6264670 0.6455967 0.5997067 0.6051112
##  [8] 0.6138351 0.6181362 0.6411938 0.6411938 0.6430120 0.6364759 0.6361112
## [15] 0.5850233 0.6087214 0.5943976 0.6087214 0.5991976 0.6087214
```

We can see overall, none of them performed as well as our knn model earlier. Upon further review- interestingly enough- the two simpler, generative models of lda and naive bayes performed the best. And the simplest of all the models-glm- performed pretty average compared to the others. That truly speaks to the interesting case of trying to predict a playoff condition by using RPM.

Maybe it's the relatively small size of the population (N=79) or the small range in data (-8 to 8), but a glm did admittedly perform above expectations. This could be in part because we preprocessed the data to slightly normalize it. Perhaps it's because of the relative simplicity of our factor/predictor combination. Regardless, we can see the result of not having a large dataset and normalizing it: we may be forced to limit ourselves to simpler methods, as the complex, "mature" models seem to overthink- overfit- and thus underperform.

Conclusion

We are unable to make full conclusions pertaining to the predictability of High Salary, Winningness, and Popularity with respect to RPM. We can say that there appears to be a strong correlation between RPM and Popularity. Maybe because high RPMs imply great defensive skill and great offense. Perhaps these players are on highlight reels or have a lot of primetime games. Perhaps they are organization studs that sell jerseys, too. But as we can see with the accuracy of attempting to predict an above average salary, it's remotely accurate but easy to assume someone makes more money than they actually do. And the worst out of the 3- playoff contention and winningness- is likely that way because a bench player with a low RPM could be on the same team as Paul George or Kawhi Leonard and be ultimately carried to playoff status. Or, a first ballot hall of famer like Lebron James could be on a poor organization and miss the playoffs despite having incredible metrics.

The struggle and evolution of metrics in sports and the NBA will continue to evolve. Perhaps the methods used should have been put under a "real time" condition consideration because of the fact that we are only using single season RPM data in this study. Further modeling could be developed by gathering RPM data of previous seasons- albeit even that has concerns with retiring players and the change in the metagame.

In terms of the practice of modeling, the strongest revelations from the modeling exercise and analysis is that relatively small populations or specific popultions tend to have better accuracy with simpler models. Perhaps multiple factors need to be considered when it comes to modeling- or perhaps the question itself was not fit for some of the models- but the important takeaway is that a model's performance is independent of its complexity, rather, it is how it is developed and executed that impacts its effectivity on a dataset. We can see that a knn- a relatively basic model- performed the best out of a massive list of models.