Estimating Player Impact Estimation from the 2016-2017 NBA Season

PSTAT 131/231 Final Project – Spring 2020

Peter Ayral
Jacob Miller
Jake Simon
Gavin Tieng

Project Report

Nature of the Data and the Data Collection Process

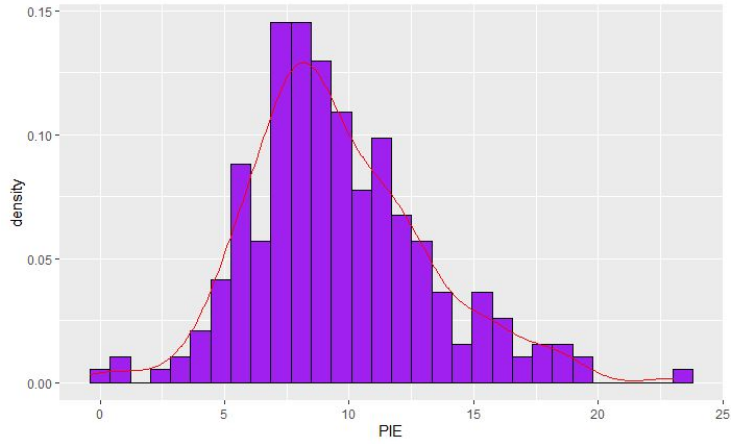
We decided to do our final project on the social power of NBA players. The dataset includes NBA player on-court performance statistics from the 2016-2017 NBA season, as well as Twitter engagement, salary, and Wikipedia pageviews. The data was collected from ESPN, Basketball-Reference, Twitter, Five-ThirtyEight, and Wikipedia. We found this dataset on Kaggle, and it was written in csv format.

Data Exploration

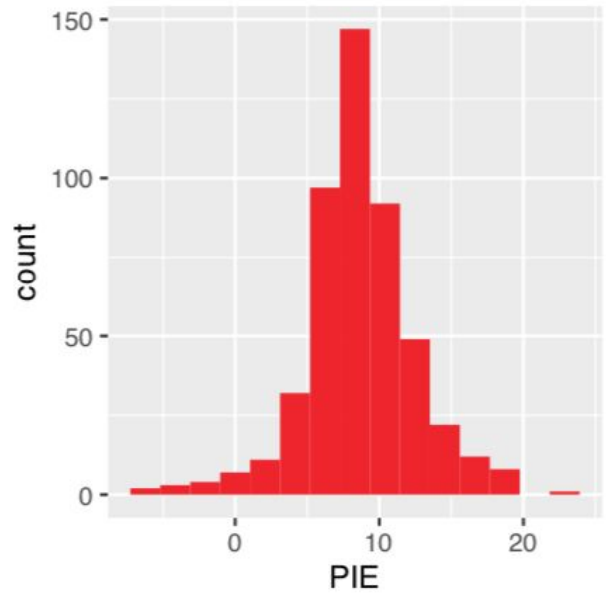
We quickly realized that only 239 of the 446 NBA players in the data had twitter accounts. We also noticed that NBA players with twitter accounts had 0.4 higher median PIE than the average NBA player.

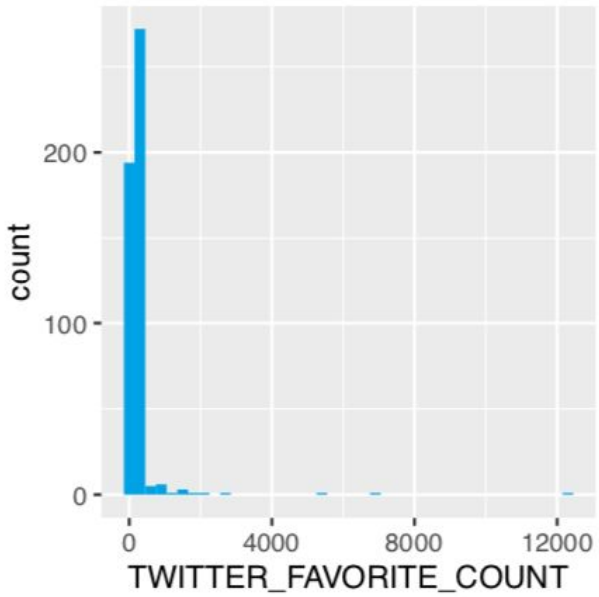
```
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-6.100   6.825   8.500   8.845  10.700  23.000
   Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
-0.400   7.250   8.900   9.559  11.700  23.000
```

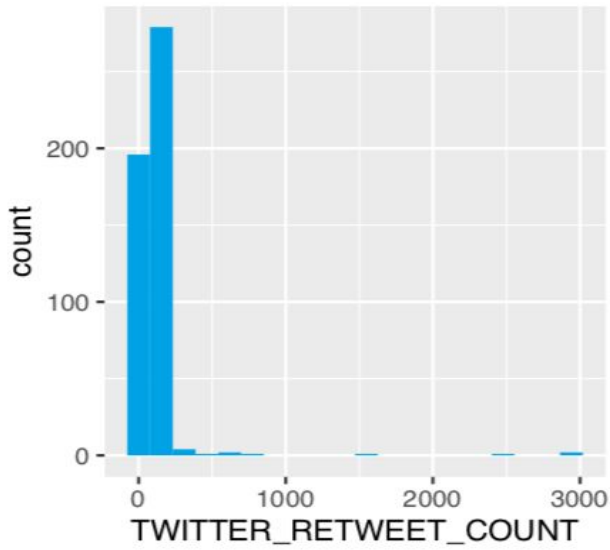
We also noticed that our response variable did not appear to follow a normal distribution, but closely resembled a Chi-squared distribution instead.

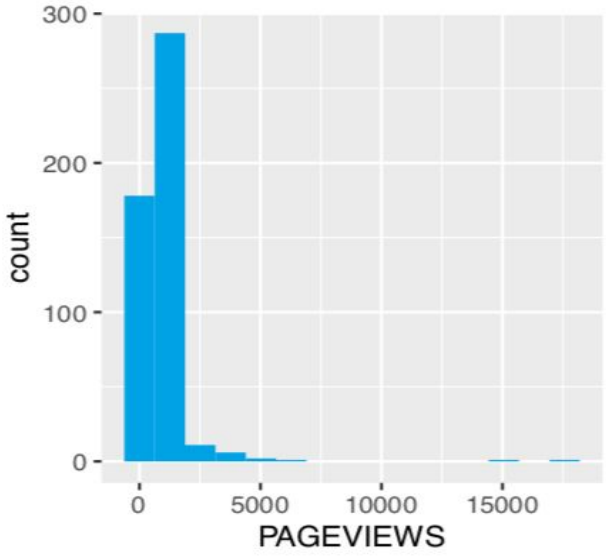


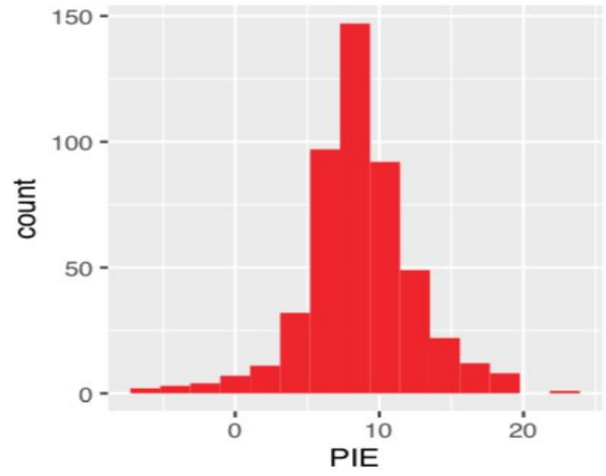
The following histograms plot the PIE, Twitter, Wikipedia, and salary data when NA values are replaced by the mean value of their column:

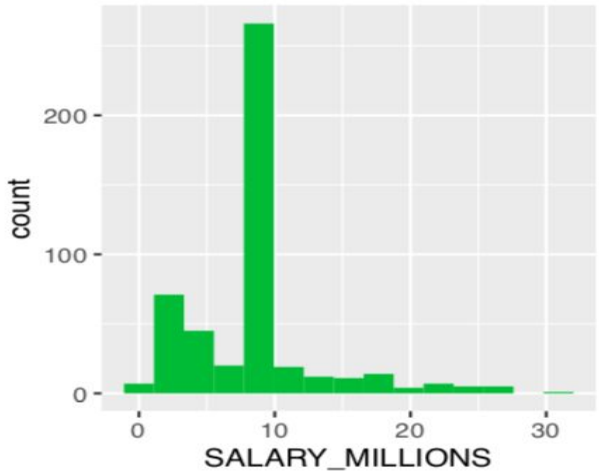

150
100
50
0
count
0
10
20
PIE


200
100
0
count
0
4000
8000
12000
TWITTER_FAVORITE_COUNT


200
100
0
count
0
1000
2000
3000
TWITTER_RETWEET_COUNT

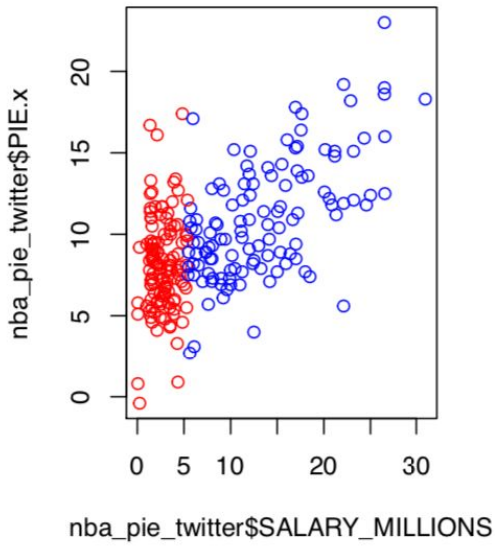

300
200
100
0
count
0
5000
10000
15000
PAGEVIEWS


150
100
50
0
count
0
10
20
PIE


200
100
0
count
0
10
20
30
SALARY_MILLIONS

The histograms above display a heavier skewed right distribution when compared to PIE. This makes sense for Twitter and Wikipedia, since only 239 players were reported to use Twitter to some capacity. For salary, players will grow their salary over time if their team values them more over time, so it makes sense there's more players below the salary average than above the salary average.

This is a scatter plot that shows the relationship between player salary and PIE. The red dots correspond to the players whose salaries are considered "low", and the blue dots correspond to the players whose salaries are considered "high". The split appears fairly linear, and we further investigated this when performing KNN.



Statement of Modeling Goals

Our modeling goal was to predict an NBA player's performance that season, and we had to decide on a response variable to use. Initially we thought we should utilize net rating, but looking at the results of this variable, we noticed that the top players of this category are all on the same team. Therefore, net rating is highly dependent on what team a player played for. We also considered usage rating to determine performance, as it seems logical that the best players will be used the most by the team. This could also be dependent on the team because a decent player on a bad team will most likely be used equally, if not more, than a great player on a great team. Finally we decided our response variable to be PIE, or Performance Impact Estimation, which is a number that quantifies a players performance impact, and the formula is as follows:

PIE = (PTS + FGM + FTM - FGA - FTA + DREB + (.5 * OREB) + AST + STL + (.5 * BLK) - PF - TO) / (GmPTS + GmFGM + GmFTM - GmFGA - GmFTA + GmDREB + (.5 * GmOREB) + GmAST + GmSTL + (.5 * GmBLK) - GmPF - GmTO)

As you can see, PIE is a comprehensive variable encompassing all aspects of a basketball player, so we felt it was the best estimator of performance. However, many of our basketball statistics in our data were in the formula for PIE. Therefore we made sure in our regressions to not include these variables, as that would mean we would be basically predicting performance with performance. Our main goal was to see whether people on social media or the owners of NBA teams were better at knowing a good player from a bad one. We quantified this by comparing engagement on social media to a player's salary when predicting PIE, and we were surprised what we found.

Statement and Brief Methodology Used

We utilized five different machine learning methods in order to predict PIE. One method was linear regression where we created a few different linear models using different variables as possible predictors and 4-fold cross validation each time. For one of the models, we also included interaction terms between the social media stats in order to optimize the model. By comparing the coefficients of determination of each model we can see which model, and therefore, which variables in combination with each other are best for accurately predicting the performance, PIE, of a player.

Next, the logistic regression method was performed. For this model's formulas, a forward step function was used to compare three categories: one that started with Twitter / Wikipedia variables, one that started with the Salary variable, and one that did not start with any additional variables (i.e. just the intercept). From there, the best formula for each category is used to calculate a logistic regression with the response being whether or not the observation is above or below the PIE mean. Then, to compare for model accuracy, the training and test errors were measured for each method. In addition, this section did some exploratory analysis with the second piece of analysis based on the logistic regression formulas. First, each category's variables and PIE were graphed on histograms to determine any pattern in frequency symmetry. Second, each formula had their AV plots graphed to determine the most impactful variables on PIE based on correlation.

After we performed a logistic regression, we decided to do a decision tree to predict performance. For this method, we decided it was best to split PIE at the median PIE into high and low factors to have a qualitative response variable. We included social media engagement and salary in the same classification tree formula because we wanted to compare them based on

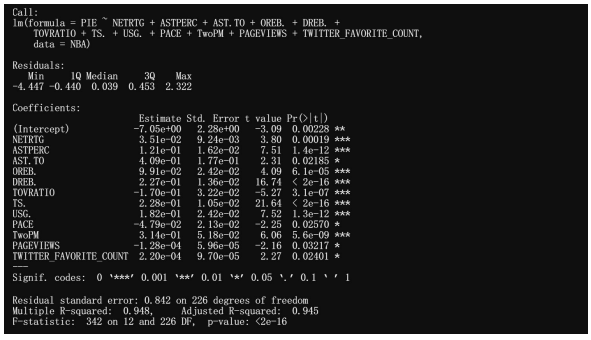
which one was the first partition of the tree. We went on to prune the data and also use cross validation to eliminate overfitting.

Following this, we thought it would be interesting to perform a random forest on the data in order to eliminate the high variance of a decision tree. Although we had a high classification error rate, we were only interested in comparing the salary and social media variables. We measured this using the mean decrease in impurity mechanism, or the Gini importance mechanism. Using this, we were able to get an accurate understanding of which variables were most relevant to classifying NBA player performance.

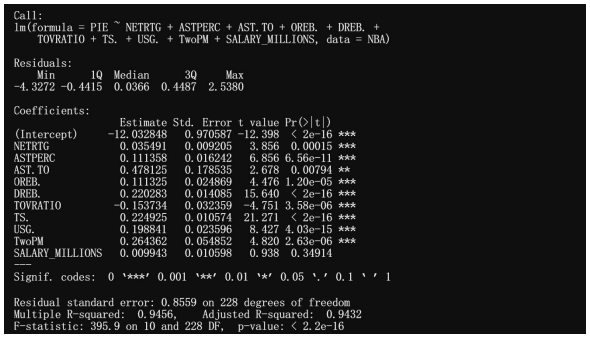
Finally, we used the K-Nearest Neighbors method to compare the error rate when using player salary to classify PIE as “high” or “low” versus the error rate when using social media engagement to classify PIE as “high” or “low”. For both salary and social media, we started by splitting the data into 50% testing and 50% training, and then proceeded to calculate the testing and training error rates when using the KNN method with k=5 nearest neighbors. After this, we used Leave One Out Cross Validation to find the optimal number of neighbors to use, and then computed the testing and training error again. This helped us determine whether social media engagement or salary was more effective in classifying PIE.

Application and Comparison of Methodology:

Overall, each of the methods gave us interesting insight into PIE. Linear regression showed that, alone, neither salary nor social media are great predictors for PIE, but a model solely based on social media outperformed a model solely based on salary. Plus, a model that included social media stats with in game stats performs almost equally to one with only in game stats. First, we created a model with all possible variables as possible predictors to see if any of the social media variables are useful alongside the in game variables, and do the same with the salary variable to compare effectiveness of prediction between salary and social media. We utilized a step function to determine the two best possible models, one using salary and one using social media. With the 16 candidate models we settled on using these two.

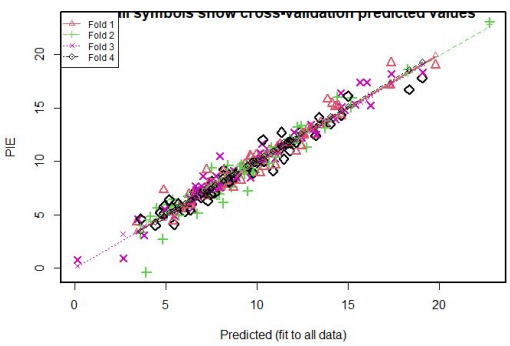


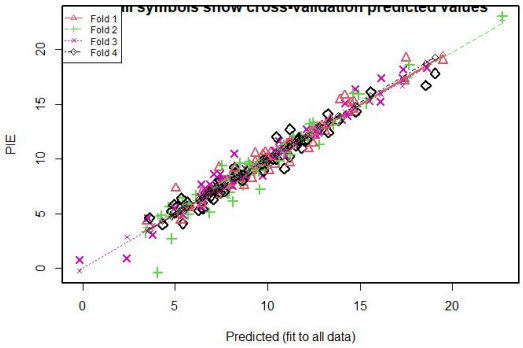
```
Call:
lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS + TWITTER_FAVORITE_COUNT,
    data = NBA)

Residuals:
    Min      1Q  Median      3Q     Max
-4.447 -0.440  0.039  0.453  2.322

Coefficients:
                       Estimate Std. Error t value Pr(>|t|)
(Intercept)          -7.05e+00   2.28e+00   -3.09  0.00228 **
NETRTG                3.51e-02   9.24e-03    3.80  0.00019 ***
ASTPERC               1.21e-01   1.62e-02    7.51  1.4e-12 ***
AST.TO                4.09e-01   1.77e-01    2.31  0.02185 *
OREB.                 9.91e-02   2.42e-02    4.09  6.1e-05 ***
DREB.                 2.27e-01   1.36e-02   16.74  < 2e-16 ***
TOVRATIO             -1.70e-01   3.22e-02   -5.27  3.1e-07 ***
TS.                   2.28e-01   1.05e-02   21.64  < 2e-16 ***
USG.                  1.82e-01   2.42e-02    7.52  1.3e-12 ***
PACE                 -4.79e-02   2.13e-02   -2.25  0.02570 *
TwoPM                 3.14e-01   5.18e-02    6.06  5.6e-09 ***
PAGEVIEWS            -1.28e-04   5.96e-05   -2.16  0.03217 *
TWITTER_FAVORITE_COUNT  2.20e-04   9.70e-05    2.27  0.02401 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.842 on 226 degrees of freedom
Multiple R-squared:  0.948,     Adjusted R-squared:  0.945
F-statistic:  342 on 12 and 226 DF,  p-value: <2e-16
```



```
Call:
lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + TwoPM + SALARY_MILLIONS, data = NBA)

Residuals:
    Min      1Q  Median      3Q     Max
-4.3272 -0.4415  0.0366  0.4487  2.5380

Coefficients:
                Estimate Std. Error t value Pr(>|t|)
(Intercept)    -12.032848   0.970587 -12.398  < 2e-16 ***
NETRTG           0.035491   0.009205   3.856  0.00015 ***
ASTPERC          0.111358   0.016242   6.856 6.56e-11 ***
AST.TO           0.478125   0.178535   2.678  0.00794 **
OREB.            0.111325   0.024869   4.476 1.20e-05 ***
DREB.            0.220283   0.014085  15.640  < 2e-16 ***
TOVRATIO        -0.153734   0.032359  -4.751 3.58e-06 ***
TS.              0.224925   0.010574  21.271  < 2e-16 ***
USG.             0.198841   0.023596   8.427 4.03e-15 ***
TwoPM            0.264362   0.054852   4.820 2.63e-06 ***
SALARY_MILLIONS  0.009943   0.010598   0.938  0.34914
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 0.8559 on 228 degrees of freedom
Multiple R-squared:  0.9456,    Adjusted R-squared:  0.9432
F-statistic: 395.9 on 10 and 228 DF,  p-value: < 2.2e-16
```

From the adjusted R Squared values and the significance of each predictor, we notice that the model that utilized social media stats performs slightly better. Running 4-fold cross validation on these two models we get





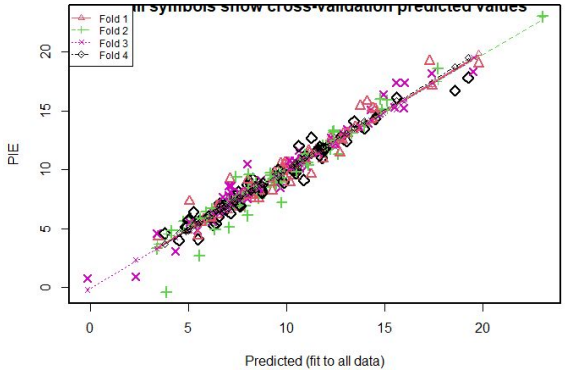

```
Linear Regression

239 samples
 10 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 180, 178, 180, 179
Resampling results:

  RMSE  Rsquared  MAE
  0.94  0.931     0.681

Tuning parameter 'intercept' was held constant at a value of TRUE
Linear Regression

239 samples
 12 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 179, 179, 179, 180
Resampling results:

  RMSE   Rsquared  MAE
  0.896  0.942     0.643
```

Which again shows that our second, social media, model has a higher R Squared and a lower root mean squared error, implying that it is the better model even though the graphs of the predicted vs fitted values appear to be nearly identical. From here we compare these models to a linear regression model that does not utilize salary or social media. To determine this model, we once again utilized a step function and got

```
Call:
lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
    USG. + REB. + PACE + AST.TO, data = NBA)

Residuals:
    Min      1Q  Median      3Q     Max
-4.177 -0.453  0.037  0.486  2.533

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -5.9582     2.3144   -2.57   0.0107 *
TwoPM        0.2585     0.0526    4.92  1.7e-06 ***
RPM          0.0880     0.0314    2.80   0.0055 **
DREB.        0.0986     0.0348    2.83   0.0050 **
ASTPERC      0.1127     0.0163    6.94  4.1e-11 ***
TS.          0.2204     0.0117   18.86  < 2e-16 ***
TOVRATIO    -0.1635     0.0324   -5.05  9.2e-07 ***
USG.         0.1991     0.0235    8.48  2.9e-15 ***
REB.         0.2239     0.0489    4.58  7.7e-06 ***
PACE        -0.0547     0.0214   -2.56   0.0110 *
AST.TO       0.4190     0.1811    2.31   0.0216 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.857 on 228 degrees of freedom
Multiple R-squared:  0.945,     Adjusted R-squared:  0.943
F-statistic:  395 on 10 and 228 DF,  p-value: <2e-16
```

From here, we took out the predictor AST.TO in order to further simplify the model as it has the least significance out of all the predictors. Once again we run 4-fold cross validation on this model and get an output of

```
Linear Regression

239 samples
  9 predictor

No pre-processing
Resampling: Cross-Validated (4 fold)
Summary of sample sizes: 180, 178, 180, 179
Resampling results:

  RMSE   Rsquared  MAE
  0.936  0.934     0.668
```



These results show that even without social media or salary, we can still produce a great model for predicting PIE. Now we made a model solely from salary and social media stats making sure to check for possible transformations, interaction terms, and possible use of weighted least squares. After checking these we fund that no transformations were necessary, weighted least squares had no positive effects, but interaction terms aided in strengthening the model. This resulted in a final model:

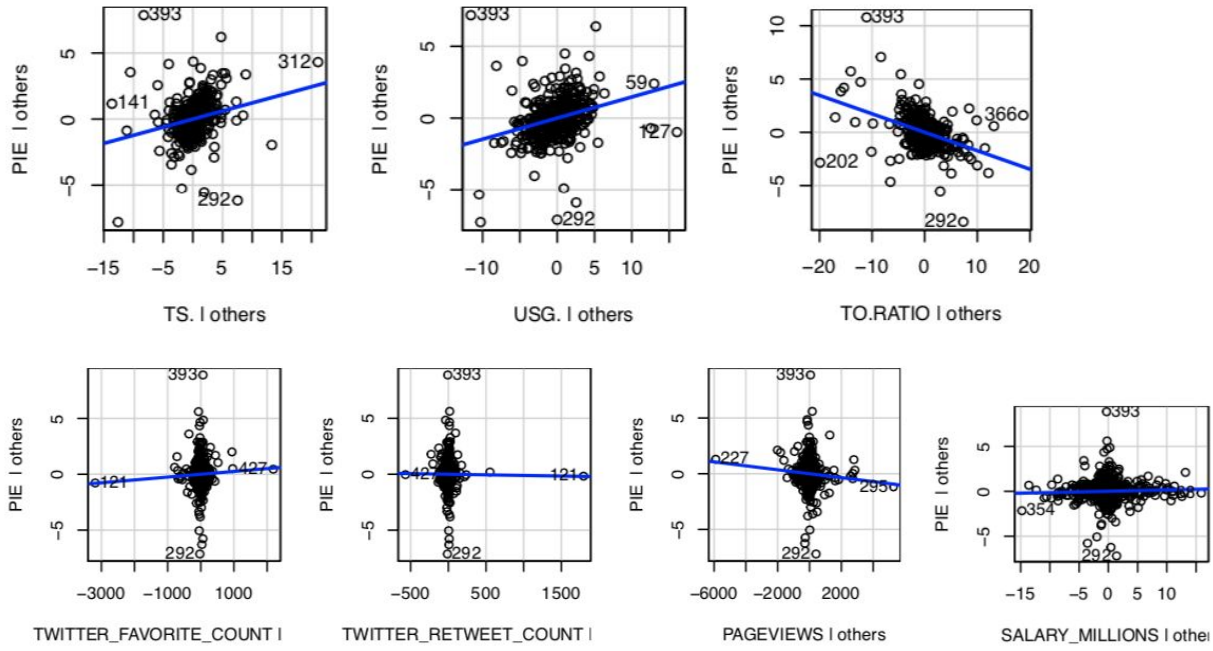
```
Call:
lm(formula = PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
    TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT +
    PAGEVIEWS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
    TWITTER_RETWEET_COUNT, data = NBA)

Residuals:
    Min      1Q  Median      3Q     Max
-7.5078 -1.7901 -0.0943  1.7697  9.3931

Coefficients:
                                          Estimate Std. Error t value Pr(>|t|)
(Intercept)                              6.980e+00  2.932e-01  23.807  < 2e-16 ***
SALARY_MILLIONS                          2.292e-01  3.114e-02   7.362 3.16e-12 ***
PAGEVIEWS                                2.257e-04  3.085e-04   0.732  0.46507
TWITTER_FAVORITE_COUNT                   1.059e-04  1.497e-03   0.071  0.94368
TWITTER_RETWEET_COUNT                    1.310e-02  5.743e-03   2.281  0.02349 *
PAGEVIEWS:TWITTER_FAVORITE_COUNT         1.031e-06  4.182e-07   2.466  0.01441 *
PAGEVIEWS:TWITTER_RETWEET_COUNT         -4.260e-06  1.573e-06  -2.708  0.00727 **
TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT -1.221e-06  4.833e-07  -2.527  0.01216 *
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.752 on 231 degrees of freedom
Multiple R-squared:  0.4299,    Adjusted R-squared:  0.4126
F-statistic: 24.88 on 7 and 231 DF,  p-value: < 2.2e-16
```

It is clear that out of all previous models, this one is by far the worst as it has less than half the adjusted R Squared value and significantly higher residual error. With these results, it appears that social media, not salary, is a better predictor for PIE when combined with in game statistics.

The logistic regression model showed that while salary and Twitter / Wikipedia slightly impact test accuracy, they do not produce a significant improvement in predicting PIE. Salary had the lowest test error, while Twitter / Wikipedia had the lowest training error. This means Twitter / Wikipedia are better at predicting model data, but salary (in addition to its additional model variables) does the best job at predicting values not already in the model data.

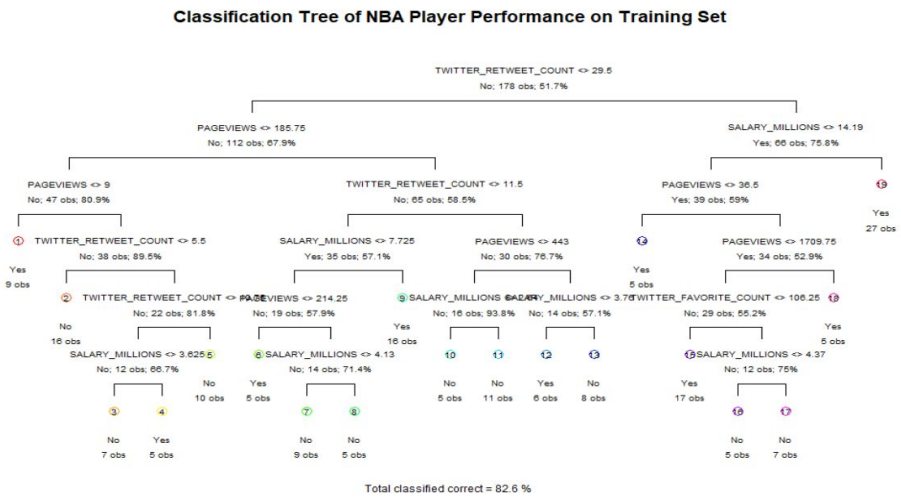
```
## [1] "The training and test errors for each method: "

##                    train.error       test.error
## Twitter       0.0862527135264616  0.12533870749221
## Salary        0.0876225765401602 0.119174231133993
## No additions 0.0924152256905457 0.121223411461862
```

The most correlated variables amongst all three groups included: Usage Percentage, Turnover Ratio, and True Shooting Percentage, while social media and salary hardly had impact on PIE; the most impactful of the four variables was Wikipedia PAGEVIEWS with a slight negative correlation.

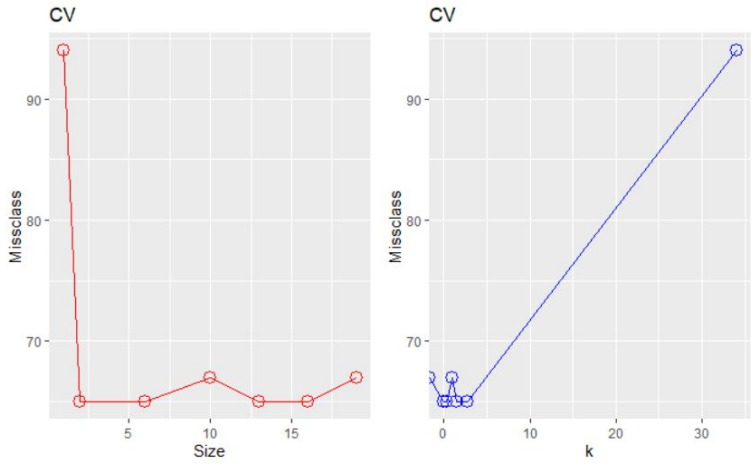


For application, we can apply the Salary logistic regression formula to truly see if Salary and its other variables would be good indicators for PIE over the course of multiple seasons. For example, we could take a player like Lebron James, and see how his increase in salary correlated with his PIE over time.

The decision tree method gave us interesting results. Using this method, we found that twitter retweet count was the first split in the decision tree, meaning it created the largest split in classifying players' PIE:



We went on to prune the tree and found that the best number of nodes was 16. We also performed cross validation on our training set and also found the best number of nodes was 16. Therefore, there would be no difference in our classification error rates for either one.



Here we can see that the misclassification error drops significantly as size increases, while on the other hand as the k-complexity increases, so does misclassification.

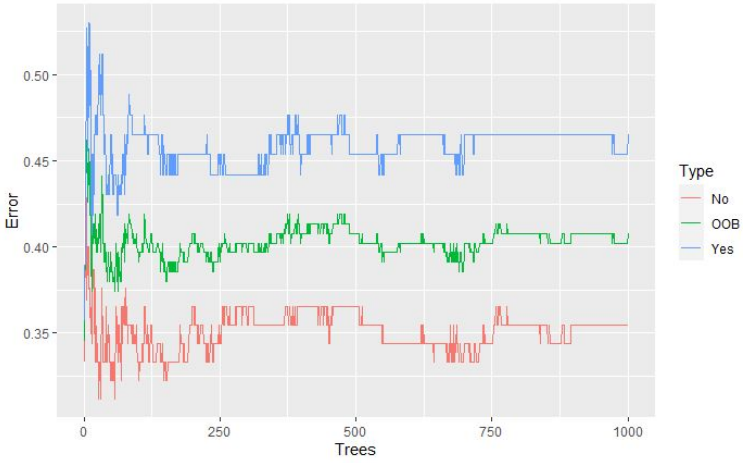
```
              nba_test_high
pred.pt.prune No Yes
          No  18  13
          Yes 10  19
[1] "Classification Error for Pruned Model:"
[1] 0.3833333
```

```
              nba_test_high
pred.pt.cv No Yes
       No  18  13
       Yes 10  19
[1] "Classification Error for CV Model:"
[1] 0.3833333
```

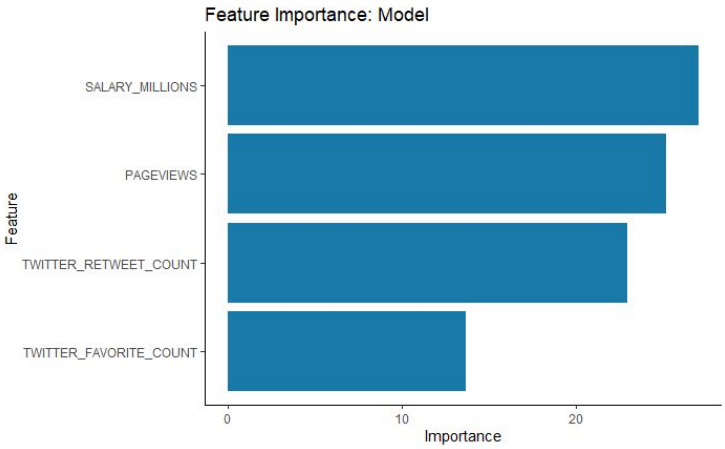
Here we can see that the prediction error on the test set is 38% for both models. In conclusion, in comparison to the other models, a decision tree was one of the worst methods for predicting performance. However, using this method we did find that twitter retweet count was a better indicator of performance than salary, meaning that people on social media were better at

knowing a good player than the team owners. Some limitations of this was the high variance of the decision tree method, and the fact that we only classified the model into high and low components. So we went further and performed a random forest of the data.

The random forest gave us a very similar classification error of 38%.

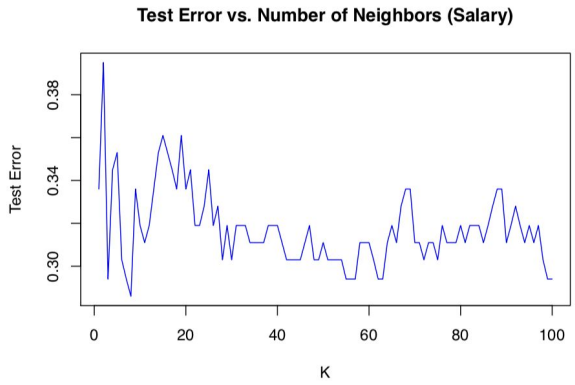


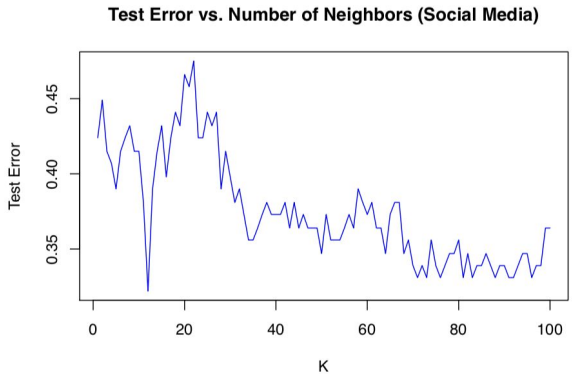
We can see that the forest misclassified players as high performance about 10% more often than low performance players. Therefore, a random forest had little improvement over a decision tree and it was still a bad method for classifying performance, but we were also interested in variable importance. We used a Gini importance mechanism to measure this, and the results matched what other methods had discovered:



We found that salary was the most important measure for classifying performance, followed by pageviews, and then retweets on twitter. This allowed us to conclude that NBA teams are better at measuring the talent of a player through salary, as opposed to a player's social media activity.

The KNN method gave us results that differed from the decision tree method. After performing LOOCV for both salary and social media, we found that the optimal number of neighbors differed:





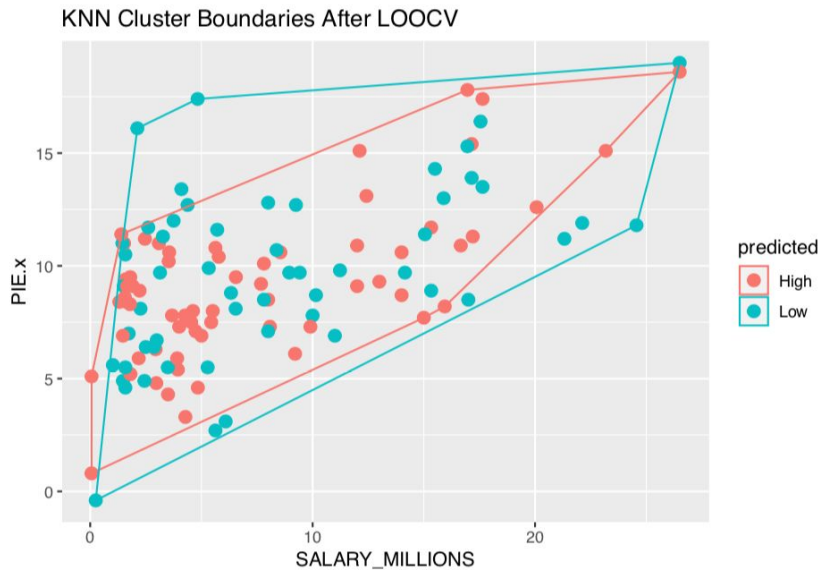
After comparing the testing error rates when using salary versus the testing error rates when using social media engagement, we found that the error rate was actually lower when using salary:

```
##   A                          B
## A Training Accuracy Rate 0.72
## B Testing Accuracy Rate  0.534
## C Training Error Rate    0.28
## D Testing Error Rate     0.466
```

```
##   A                          B
## A Training Accuracy Rate 0.765
## B Testing Accuracy Rate  0.697
## C Training Error Rate    0.235
## D Testing Error Rate     0.303
```

This result makes sense in the context of our problem; a player who performs better (ie. plays for a higher-ranked team, or scores more points per game) is more likely to be paid more, as he is more desirable to have on a team. On the other hand, social media is a huge indicator of player popularity. Some players are more liked because of the way that they engage with their fans on social media, and may be paid more because of their social media influence.

We also plotted the KNN boundary after computing the best number of neighbors with LOOCV, which had interesting results.



The red and blue represent boundary lines for classifying PIE as high or low. The plot itself does not tell us much, besides that salary may not be the best predictor to classify PIE by. The low boundary eclipses the high boundary, which makes it difficult to make any definitive conclusions about usefulness of this classifier. This makes sense because PIE is computed using a combination of player statistics, such as points, turnovers, possessions, etc.

In conclusion, through cross validation, we discovered that linear regression was the best model in predicting player performance. We also found that while salary was the most important variable, combining all of the social engagement variables predicts player performance better than just salary. Overall, if we were to repeat this experiment, we would have wanted more metrics of social media engagement, such as instagram followers or likes. We would also have liked more metrics of wage, such as sponsor deals. It was surprising to us how we could use machine learning to predict how good a player is in the NBA, and we hope to be able to learn and apply more methods in the future.

Contribution Statement:

Peter Ayral: Coded the decision tree and random forest methods and described these methodologies in the section above, wrote out the statement of modeling goals as well as organized meet up times for the group to work on the project.

Jacob Miller: Coded the linear regression method and described the methodology above. Communicated with the group about real life meanings and interactions between basketball statistics.

Jake Simon: Coded the logistic regression method and described the methodology above. Was responsible for coding a majority of the data exploration and double checked everyone else's code for mistakes.

Gavin Tieng: Coded the K Nearest Neighbors method and described the methodology above. Took everyones code and organized it into a single document for the final report.

More analysis and comments are included in the RMarkdown.

Linear Regression Method

```
nba_twitter <- read.csv('nba_2017_players_with_salary_wiki_twitter.csv')
nba_pie <- read.csv('nba_2017_pie.csv')

names(nba_pie)[1] <- "PLAYER"

nba_pie_twitter <- merge(nba_pie, nba_twitter, by='PLAYER')
nba_pie_twitter <- nba_pie_twitter[,-grep('.y',names(nba_pie_twitter))]
```

```
#rename certain variables so they make more sense
NBA<-nba_pie_twitter %>%
  rename(
    TEAM = TEAM.x,
    AGE = AGE.x,
    GP = GP.x,
    WINS = W.x,
    LOSSES = L,
    ASTPERC = AST.,
    ASTRATIO = ASTÂ.RATIO,
    TOVRATIO = TOÂ.RATIO,
    PACE = PACE.x,
    PIE = PIE.x,
    ThreePM = X3P,
    ThreePA = X3PA,
    ThreePPerc = X3P.,
    TwoPM = X2P,
    TwoPA = X2PA,
    TwoPPerc = X2P.,
    FTperc = FT.
  ) %>%
  select(-c(eFG., MIN, MP, X, OFFRTG, DEFRTG, WINS_RPM, ORPM, DRPM))
#Get rid of duplicate and irrelavent variables
#Now i noticed that ThreePPerc has NA values that, realistically, are 0% because they were inputed as
#Same with FTperc if a player never shot a freethrow
#same with the twitter stats if a player does not have a twitter
NBA$ThreePPerc[which(is.na(NBA$ThreePPerc))] <- 0
NBA$FTperc[which(is.na(NBA$FTperc))] <- 0
NBA$TWITTER_FAVORITE_COUNT[which(is.na(NBA$TWITTER_FAVORITE_COUNT))] <- 0
NBA$TWITTER_RETWEET_COUNT[which(is.na(NBA$TWITTER_RETWEET_COUNT))] <- 0
```

```
mean(NBA[,'PIE'])
```

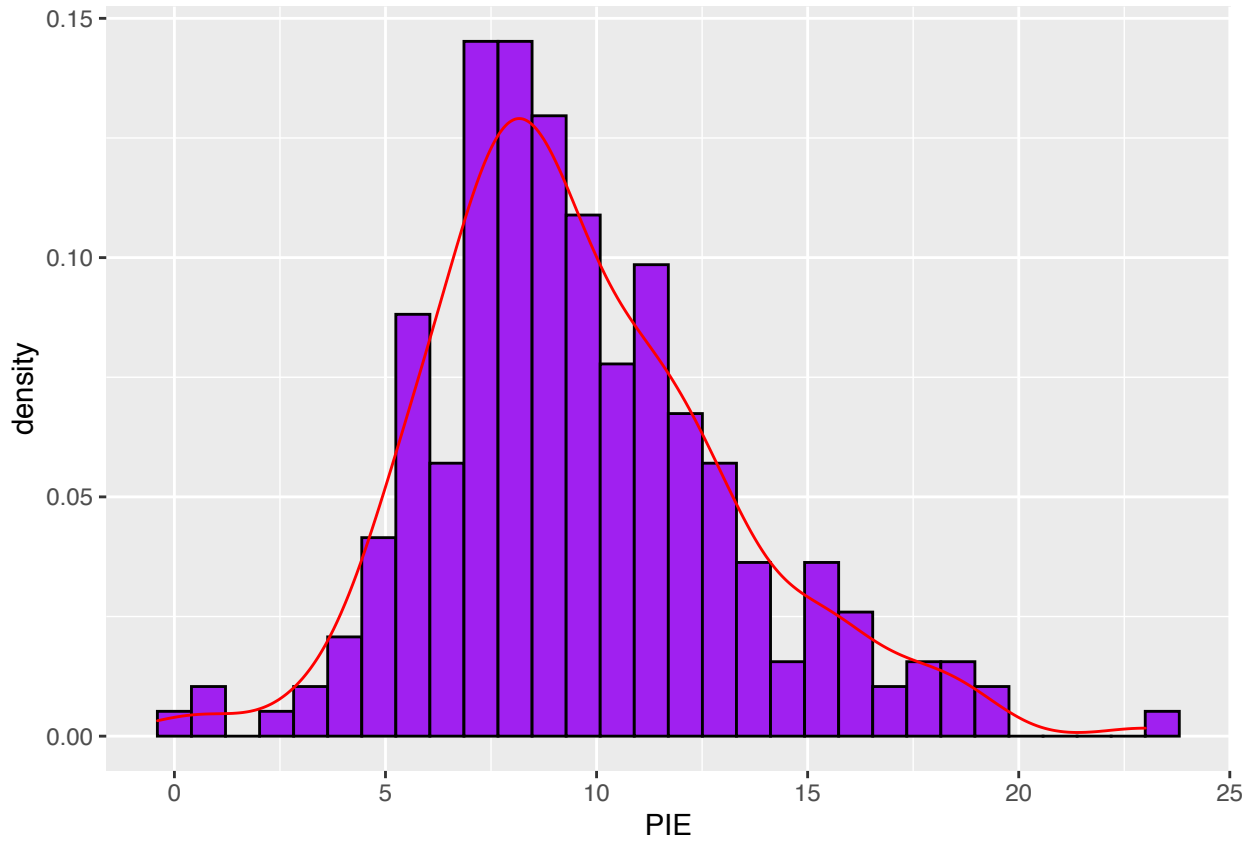
```
## [1] 9.558996
```

```
sd(NBA[,'PIE'])
```

```
## [1] 3.590322
```

```
ggplot(data = NBA, aes(PIE, after_stat(density))) + geom_histogram(color = 'black', fill = 'purple')
```

```
## `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



Looking at this histogram of our response variable, PIE, we can notice that its distribution appears to follow a Chi-Square distribution with degrees of freedom of about 9, with an approximate mean of 9.558996 and sample standard deviation of 3.590322.

```
shapiro.test(NBA$PIE)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  NBA$PIE
## W = 0.97408, p-value = 0.0002317
```

With this test, We now know that PIE does not follow a normal distribution. With this knowledge in mind, i will continue with linear regression as normal, but will normalize data at the end and compare the models.

I want to utilize linear regression to see if Twitter can be utilized as an effective tool to predict the performance of an NBA player. I will create three linear regression models, one using all possible variables, one using strictly social media variables, and one using strictly non-social media attributes. Using these variables, I will predict the PIE (Player Impact Estimation) score of a player.

First, I will create a model with all possible variables as possible predictors to see if any of the social media variables are useful alongside the in game variables, and do the same with the salary variable to compare effectiveness of prediction between salary and social media

```
#take out PTS, + FGM + FTM - FGA - FTA + DREB + (.5 * OREB) + AST + STL + (.5 * BLK) - PF - TO) /
#(GmPTS + GmFGM + GmFTM - GmFGA - GmFTA + GmDREB + (.5 * GmOREB) + GmAST + GmSTL + (.5 * GmBLK) - GmP
#MODEL 1 Possibility (SALARY)
fullSalary<-lm(PIE~ AGE + GP + WINS + LOSSES + NETRTG + ASTPERC + AST.TO +
                 ASTRATIO + OREB. + DREB. + REB. + TOVRATIO + EFG. + TS. +
                 USG. + PACE + Rk + FG. + ThreePM + ThreePA + ThreePPerc +
                 TwoPM + TwoPA + TwoPPerc + FTperc + MPG + RPM + SALARY_MILLIONS,
                 data = NBA)

#MODEL 2 Possibility (SOCAIL MEDIA)
fullSocial<- lm(PIE~ AGE + GP + WINS + LOSSES + NETRTG + ASTPERC + AST.TO +
                  ASTRATIO + OREB. + DREB. + REB. + TOVRATIO + EFG. + TS. +
                  USG. + PACE + Rk + FG. + ThreePM + ThreePA + ThreePPerc +
                  TwoPM + TwoPA + TwoPPerc + FTperc + MPG + RPM + PAGEVIEWS +
                  TWITTER_FAVORITE_COUNT + TWITTER_RETWEET_COUNT, data = NBA)
red_model <- lm(PIE~1, data = NBA)
n <- length(NBA[,'PIE'])
```

```
#Model 1.1
step(red_model, scope = list(lower = red_model, upper = fullSalary), direction = 'forward',
     trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO + NETRTG + TwoPPerc + MPG + OREB.,
##     data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -7.25389      0.41461      0.04153      0.39923      0.11525      0.25409
##    TOVRATIO         USG.         REB.         PACE       AST.TO       NETRTG
##    -0.15109      0.16962     -0.36658     -0.04057      0.39333      0.03543
##    TwoPPerc          MPG        OREB.
##    -2.75072     -0.02424      0.27586
```

```
#Model 1.2
step(red_model, scope = list(lower = red_model, upper = fullSalary), direction = 'forward',
     k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -5.95817      0.25853      0.08802      0.09859      0.11274      0.22040
##    TOVRATIO         USG.         REB.         PACE       AST.TO
##    -0.16354      0.19911      0.22389     -0.05475      0.41897
```

```r
#Model 1.3
step(red_model, scope = list(lower = red_model, upper = fullSalary), direction = 'backward',
     trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
##
## Coefficients:
## (Intercept)
##       9.559
```

```r
#Model 1.4
step(red_model, scope = list(lower = red_model, upper = fullSalary), direction = 'backward',
     k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
##
## Coefficients:
## (Intercept)
##       9.559
```

```r
#Model 1.5
step(fullSalary, scope = list(lower = red_model, upper = fullSalary), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + PACE + TwoPM + TwoPPerc, data = NBA)
##
## Coefficients:
## (Intercept)       NETRTG      ASTPERC       AST.TO        OREB.        DREB.
##    -7.49397      0.03378      0.11618      0.43269      0.11259      0.22194
##    TOVRATIO          TS.         USG.         PACE        TwoPM     TwoPPerc
##    -0.15678      0.24826      0.18611     -0.04623      0.30465     -2.07755
```

```r
#Model 1.6
step(fullSalary, scope = list(lower = red_model, upper = fullSalary), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + TwoPM, data = NBA)
##
## Coefficients:
## (Intercept)       NETRTG      ASTPERC       AST.TO        OREB.        DREB.
##   -12.03969      0.03629      0.11295      0.47552      0.10706      0.22349
##    TOVRATIO          TS.         USG.        TwoPM
##    -0.15662      0.22574      0.19749      0.28249
```

```r
#Model 1.7
step(red_model, scope = list(lower = red_model, upper = fullSalary), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + PACE + AST.TO + NETRTG + TwoPPerc + OREB., data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM        DREB.      ASTPERC          TS.     TOVRATIO
##    -7.49397      0.30465      0.22194      0.11618      0.24826     -0.15678
##        USG.         PACE       AST.TO       NETRTG     TwoPPerc        OREB.
##     0.18611     -0.04623      0.43269      0.03378     -2.07755      0.11259
```

```r
#Model 1.8
step(red_model, scope = list(lower = red_model, upper = fullSalary), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -5.95817      0.25853      0.08802      0.09859      0.11274      0.22040
##    TOVRATIO         USG.         REB.         PACE       AST.TO
##    -0.16354      0.19911      0.22389     -0.05475      0.41897
```

```r
#Model 2.1
step(red_model, scope = list(lower = red_model, upper = fullSocial), direction = 'forward',
     trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO + NETRTG + TwoPPerc + MPG + OREB.,
##     data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -7.25389      0.41461      0.04153      0.39923      0.11525      0.25409
##    TOVRATIO         USG.         REB.         PACE       AST.TO       NETRTG
##    -0.15109      0.16962     -0.36658     -0.04057      0.39333      0.03543
##    TwoPPerc          MPG        OREB.
##    -2.75072     -0.02424      0.27586
```

```r
#Model 2.2
step(red_model, scope = list(lower = red_model, upper = fullSocial), direction = 'forward',
     k = log(n), trace = 0)
```

```
##
```

```
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -5.95817      0.25853      0.08802      0.09859      0.11274      0.22040
##    TOVRATIO         USG.         REB.         PACE       AST.TO
##    -0.16354      0.19911      0.22389     -0.05475      0.41897
```

```
#Model 2.3
step(red_model, scope = list(lower = red_model, upper = fullSocial), direction = 'backward',
     trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
##
## Coefficients:
## (Intercept)
##       9.559
```

```
#Model 2.4
step(red_model, scope = list(lower = red_model, upper = fullSocial), direction = 'backward',
     k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
##
## Coefficients:
## (Intercept)
##       9.559
```

```
#Model 2.5
step(fullSocial, scope = list(lower = red_model, upper = fullSocial), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + PACE + TwoPM + TwoPPerc + MPG + PAGEVIEWS +
##     TWITTER_FAVORITE_COUNT, data = NBA)
##
## Coefficients:
##            (Intercept)                  NETRTG                 ASTPERC
##              -6.8817393               0.0361781               0.1255202
##                  AST.TO                   OREB.                   DREB.
##               0.3578745               0.0886667               0.2263792
##                TOVRATIO                     TS.                    USG.
##              -0.1725553               0.2536495               0.1548050
##                    PACE                   TwoPM                TwoPPerc
##              -0.0428875               0.4380689              -2.5712376
##                     MPG                PAGEVIEWS  TWITTER_FAVORITE_COUNT
##              -0.0221681              -0.0001312               0.0002263
```

```
#Model 2.6
step(fullSocial, scope = list(lower = red_model, upper = fullSocial), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + TwoPM, data = NBA)
##
## Coefficients:
## (Intercept)       NETRTG      ASTPERC       AST.TO        OREB.        DREB.
##   -12.03969      0.03629      0.11295      0.47552      0.10706      0.22349
##    TOVRATIO          TS.         USG.        TwoPM
##    -0.15662      0.22574      0.19749      0.28249
```

```
#Model 2.7
step(red_model, scope = list(lower = red_model, upper = fullSocial), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + PACE + AST.TO + NETRTG + TwoPPerc + OREB., data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM        DREB.      ASTPERC          TS.     TOVRATIO
##    -7.49397      0.30465      0.22194      0.11618      0.24826     -0.15678
##        USG.         PACE       AST.TO       NETRTG     TwoPPerc        OREB.
##     0.18611     -0.04623      0.43269      0.03378     -2.07755      0.11259
```

```
#Model 2.8
step(red_model, scope = list(lower = red_model, upper = fullSocial), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -5.95817      0.25853      0.08802      0.09859      0.11274      0.22040
##    TOVRATIO         USG.         REB.         PACE       AST.TO
##    -0.16354      0.19911      0.22389     -0.05475      0.41897
```

From looking at all 16 candidate models we can see that only 1 of them contains any possible predictors that are not in game stats. This tells us that salary and social media are not very important for determining PIE of a player overall. For the sake of figuring out which is a better predictor between salary and social media, despite neither extremely important, I will move forward with models 1.6 + SALARY_MILIONS because it was the simplest of all the salary models that wasn't just the intercept and 2.5 because it was the only candidate model that utilized social media predictors.

```
SALARYmodel<-lm(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + TwoPM + SALARY_MILLIONS, data = NBA)
```

```r
SOCIALmodel<-lm(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + PACE + TwoPM + TwoPPerc + MPG + PAGEVIEWS +
    TWITTER_FAVORITE_COUNT, data = NBA)

summary(SALARYmodel)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + TwoPM + SALARY_MILLIONS, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.3272 -0.4415  0.0366  0.4487  2.5380
##
## Coefficients:
##                   Estimate Std. Error t value Pr(>|t|)
## (Intercept)     -12.032848   0.970587 -12.398  < 2e-16 ***
## NETRTG            0.035491   0.009205   3.856  0.00015 ***
## ASTPERC           0.111358   0.016242   6.856 6.56e-11 ***
## AST.TO            0.478125   0.178535   2.678  0.00794 **
## OREB.             0.111325   0.024869   4.476 1.20e-05 ***
## DREB.             0.220283   0.014085  15.640  < 2e-16 ***
## TOVRATIO         -0.153734   0.032359  -4.751 3.58e-06 ***
## TS.               0.224925   0.010574  21.271  < 2e-16 ***
## USG.              0.198841   0.023596   8.427 4.03e-15 ***
## TwoPM             0.264362   0.054852   4.820 2.63e-06 ***
## SALARY_MILLIONS   0.009943   0.010598   0.938  0.34914
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8559 on 228 degrees of freedom
## Multiple R-squared:  0.9456, Adjusted R-squared:  0.9432
## F-statistic: 395.9 on 10 and 228 DF,  p-value: < 2.2e-16
```

```r
summary(SOCIALmodel)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + PACE + TwoPM + TwoPPerc + MPG + PAGEVIEWS +
##     TWITTER_FAVORITE_COUNT, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4389 -0.4461  0.0275  0.4733  2.2004
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -6.882e+00  2.272e+00  -3.030 0.002737 **
## NETRTG                  3.618e-02  9.362e-03   3.864 0.000146 ***
## ASTPERC                 1.255e-01  1.618e-02   7.756 3.06e-13 ***
```

```
## AST.TO                   3.579e-01  1.776e-01   2.015 0.045090 *
## OREB.                    8.867e-02  2.732e-02   3.245 0.001354 **
## DREB.                    2.264e-01  1.355e-02  16.703  < 2e-16 ***
## TOVRATIO                -1.726e-01  3.207e-02  -5.380 1.87e-07 ***
## TS.                      2.536e-01  1.652e-02  15.353  < 2e-16 ***
## USG.                     1.548e-01  2.717e-02   5.697 3.81e-08 ***
## PACE                    -4.289e-02  2.133e-02  -2.011 0.045552 *
## TwoPM                    4.381e-01  8.715e-02   5.027 1.02e-06 ***
## TwoPPerc                -2.571e+00  1.329e+00  -1.935 0.054296 .
## MPG                     -2.217e-02  1.422e-02  -1.559 0.120445
## PAGEVIEWS               -1.312e-04  5.946e-05  -2.206 0.028366 *
## TWITTER_FAVORITE_COUNT   2.263e-04  9.661e-05   2.343 0.020030 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8364 on 224 degrees of freedom
## Multiple R-squared:  0.9489, Adjusted R-squared:  0.9457
## F-statistic: 297.2 on 14 and 224 DF,  p-value: < 2.2e-16
```

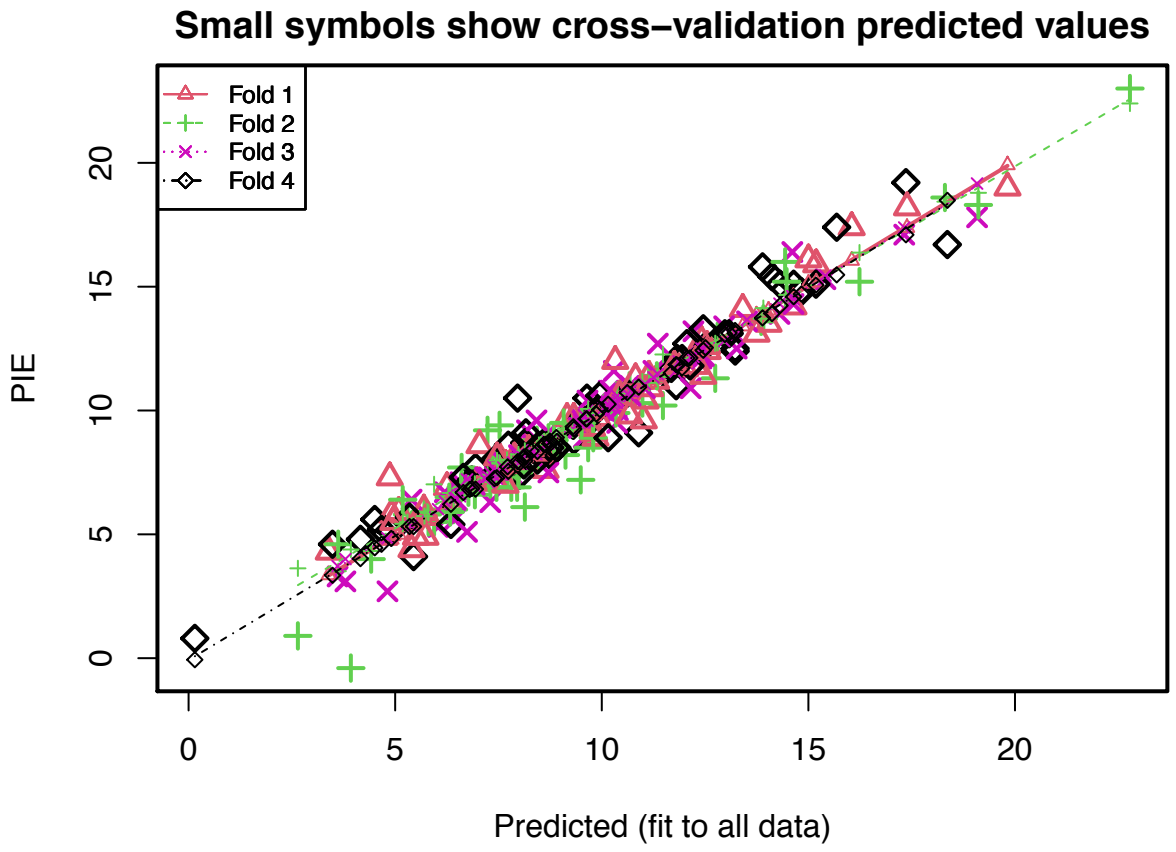
```
#take out MPG and TwoPPerc from SOCIAL
SOCIALnew<-lm(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS +
    TWITTER_FAVORITE_COUNT, data = NBA)
summary(SOCIALnew) #now all predictors are significant
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS + TWITTER_FAVORITE_COUNT,
##     data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.4469 -0.4399  0.0389  0.4534  2.3221
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)             -7.046e+00  2.283e+00  -3.086 0.002281 **
## NETRTG                   3.507e-02  9.237e-03   3.796 0.000189 ***
## ASTPERC                  1.214e-01  1.616e-02   7.508 1.37e-12 ***
## AST.TO                   4.091e-01  1.772e-01   2.309 0.021849 *
## OREB.                    9.909e-02  2.424e-02   4.088 6.06e-05 ***
## DREB.                    2.269e-01  1.356e-02  16.738  < 2e-16 ***
## TOVRATIO                -1.697e-01  3.217e-02  -5.274 3.11e-07 ***
## TS.                      2.276e-01  1.051e-02  21.645  < 2e-16 ***
## USG.                     1.817e-01  2.416e-02   7.519 1.28e-12 ***
## PACE                    -4.787e-02  2.132e-02  -2.246 0.025697 *
## TwoPM                    3.139e-01  5.179e-02   6.062 5.59e-09 ***
## PAGEVIEWS               -1.284e-04  5.956e-05  -2.156 0.032173 *
## TWITTER_FAVORITE_COUNT   2.204e-04  9.701e-05   2.272 0.024011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8417 on 226 degrees of freedom
```

```
## Multiple R-squared:  0.9478, Adjusted R-squared:  0.945
## F-statistic: 342.1 on 12 and 226 DF,  p-value: < 2.2e-16
```

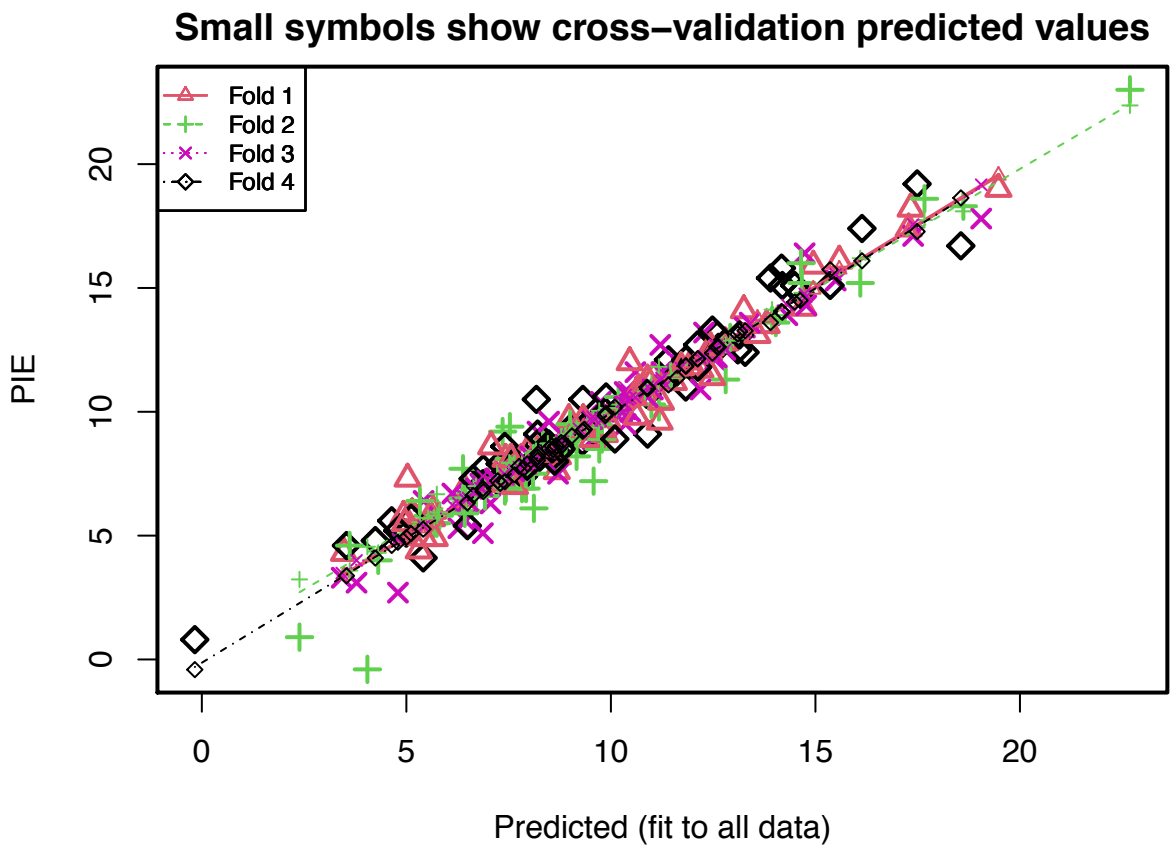
```
#SALARY Cross Validation
cv.lm(NBA, form.lm = formula(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + TwoPM + SALARY_MILLIONS), m =4, seed =1, printit = FALSE )
```

```
## Warning in cv.lm(NBA, form.lm = formula(PIE ~ NETRTG + ASTPERC + AST.TO + :
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
#SOCIAL Cross Validation
cv.lm(NBA, form.lm = formula(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS +
    TWITTER_FAVORITE_COUNT), m =4, seed =1, printit = FALSE )
```

```
## Warning in cv.lm(NBA, form.lm = formula(PIE ~ NETRTG + ASTPERC + AST.TO + :
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```




```
set.seed(1)
train.control<-trainControl(method = "cv", number = 4)

SALARYcv <- train(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + TwoPM + SALARY_MILLIONS, data = NBA, method = "lm",
                trControl = train.control)

SOCIALcv <- train(PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
    TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS +
    TWITTER_FAVORITE_COUNT, data = NBA, method = "lm",
                trControl = train.control)

print(SALARYcv)
```

```
## Linear Regression
##
## 239 samples
##  10 predictor
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 179, 179, 180, 179
## Resampling results:
##
##   RMSE       Rsquared   MAE
```

```
##   0.9422932  0.9343185  0.6734699
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
print(SOCIALcv)
```

```
## Linear Regression
##
## 239 samples
##  12 predictor
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 179, 180, 180, 178
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9249472  0.9359005  0.6714359
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

Although the graphs of models appear to be near identical, the model that includes social predictors has a slightly lower root Mean square error (0.896 vs 0.94) as well as a higher adjusted R-squared (0.942 vs 0.931). From this we can see that, although only slightly, social media predictors can be used as more effective predictors for PIE compared to salary. From here though id like to compare this result to an absolute best model that may not include salary or social media as predictors.

```
#All Possible Variables
full_model1<- lm(PIE~ AGE + GP + WINS + LOSSES + NETRTG + ASTPERC + AST.TO + ASTRATIO + OREB. +
                   DREB. + REB. + TOVRATIO + EFG. + TS. + USG. + PACE + Rk + FG + FGA + FG. +
                   ThreePM + ThreePA + ThreePPerc + TwoPM + TwoPA + TwoPPerc +FT + FTA +
                   FTperc + ORB + DRB + TRB + AST + STL + BLK + TOV + PF + POINTS + MPG +
                   RPM + SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
                   TWITTER_RETWEET_COUNT, data = NBA)
red_model <- lm(PIE~1, data = NBA)
n <- length(NBA[,'PIE'])

#take out PTS, + FGM + FTM - FGA - FTA + DREB + (.5 * OREB) + AST + STL + (.5 * BLK) - PF - TO)
#/ (GmPTS + GmFGM + GmFTM - GmFGA - GmFTA + GmDREB + (.5 * GmOREB) + GmAST +
#GmSTL + (.5 * GmBLK) - GmPF - GmTO)
full_model2<- lm(PIE~ AGE + GP + WINS + LOSSES + NETRTG + ASTPERC + AST.TO + ASTRATIO + OREB. +
                   DREB. + REB. + TOVRATIO + EFG. + TS. + USG. + PACE + Rk + FG. + ThreePM +
                   ThreePA + ThreePPerc +TwoPM + TwoPA + TwoPPerc + FTperc +
                   MPG + RPM + SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
                   TWITTER_RETWEET_COUNT, data = NBA)
```

```
#Model 1
step(red_model, scope = list(lower = red_model, upper = full_model1), direction = 'forward',
     trace = 0)
```

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

```
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + PF +
##     USG. + TOVRATIO + ORB + STL + BLK + ASTRATIO + TwoPPerc +
##     OREB. + PACE + NETRTG + FT + FTA + FTperc + AST.TO + REB.,
##     data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -8.30155      0.28367     -0.06160      0.44179      0.08725      0.27638
##          PF         USG.     TOVRATIO          ORB          STL          BLK
##    -1.06365      0.19775     -0.12187      0.21927      1.05843      0.80701
##    ASTRATIO     TwoPPerc        OREB.         PACE       NETRTG           FT
##     0.02719     -3.31308      0.30660     -0.03681      0.03095      0.83617
##         FTA       FTperc       AST.TO         REB.
##    -0.61053     -1.17706      0.32737     -0.43067
```

```
#Model 2
step(red_model, scope = list(lower = red_model, upper = full_model1), direction = 'forward',
     k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + PF +
##     USG. + TOVRATIO + ORB + STL + BLK + ASTRATIO + TwoPPerc +
##     OREB. + PACE, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -7.34646      0.29767      0.02250      0.22148      0.08744      0.26740
##          PF         USG.     TOVRATIO          ORB          STL          BLK
##    -1.09802      0.20546     -0.17088      0.12501      1.06073      0.71159
##    ASTRATIO     TwoPPerc        OREB.         PACE
##     0.05481     -3.51064      0.10662     -0.04226
```

```
#Model 3
step(red_model, scope = list(lower = red_model, upper = full_model1), direction = 'backward',
     trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
##
## Coefficients:
## (Intercept)
##       9.559
```

```
#Model 4
step(red_model, scope = list(lower = red_model, upper = full_model1), direction = 'backward',
     k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ 1, data = NBA)
```

```
##
## Coefficients:
## (Intercept)
##       9.559
```

```
#Model 5
step(full_model1, scope = list(lower = red_model, upper = full_model1), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     REB. + TOVRATIO + EFG. + TS. + PACE + Rk + ThreePPerc + TwoPA +
##     FT + FTA + AST + STL + BLK + PF + MPG + RPM + POINTS, data = NBA)
##
## Coefficients:
## (Intercept)       NETRTG      ASTPERC       AST.TO        OREB.        DREB.
##    3.313239     0.040001     0.133615     0.419872     0.372596     0.455632
##        REB.     TOVRATIO         EFG.          TS.         PACE           Rk
##   -0.461611    -0.064552     0.109207     0.085104    -0.068323    -0.008731
##  ThreePPerc        TwoPA           FT          FTA          AST          STL
##    1.039923     0.147760     0.975370    -0.502779    -0.143029     1.157622
##         BLK           PF          MPG          RPM       POINTS
##    0.848155    -1.039855    -0.146950    -0.066343     0.109527
```

```
#Model 6
step(full_model1, scope = list(lower = red_model, upper = full_model1), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + EFG. + PACE + Rk + ThreePPerc + TwoPA + FT + FTA +
##     STL + BLK + PF + MPG, data = NBA)
##
## Coefficients:
## (Intercept)       NETRTG      ASTPERC       AST.TO        OREB.        DREB.
##     5.70099      0.02340      0.10709      0.37892      0.12844      0.21575
##    TOVRATIO         EFG.         PACE           Rk   ThreePPerc        TwoPA
##    -0.06360      0.18655     -0.06995     -0.01045      1.12413      0.20937
##          FT          FTA          STL          BLK           PF          MPG
##     1.45815     -0.71931      0.99894      0.73134     -1.03700     -0.16008
```

```
#Model 7
step(red_model, scope = list(lower = red_model, upper = full_model1), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + PF +
##     TOVRATIO + STL + BLK + OREB. + PACE + NETRTG + FT + FTA +
##     AST.TO + AST + Rk + MPG + EFG. + REB. + ThreePPerc + FGA,
##     data = NBA)
##
## Coefficients:
```

```
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    3.482705     0.257981    -0.064963     0.451030     0.133696     0.085142
##          PF     TOVRATIO          STL          BLK        OREB.         PACE
##   -1.037489    -0.061710     1.149336     0.840322     0.371618    -0.071086
##      NETRTG           FT          FTA       AST.TO          AST           Rk
##    0.039925     1.103166    -0.516172     0.430622    -0.149410    -0.008845
##         MPG         EFG.         REB.   ThreePPerc          FGA
##   -0.147791     0.110593    -0.454496     1.143114     0.126557
```

```
#Model 8
step(red_model, scope = list(lower = red_model, upper = full_model1), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + DREB. + ASTPERC + TS. + PF + USG. +
##     TOVRATIO + STL + BLK + ASTRATIO + TwoPPerc + OREB. + PACE,
##     data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM        DREB.      ASTPERC          TS.           PF
##    -7.52895      0.31927      0.22587      0.08810      0.27196     -1.08500
##        USG.     TOVRATIO          STL          BLK     ASTRATIO     TwoPPerc
##     0.20161     -0.17580      1.13741      0.75131      0.05617     -3.69708
##       OREB.         PACE
##     0.12525     -0.04311
```

```
#After taking out variables used to actually generte the value of PIE illrepeat to see any
#possible differences
step(red_model, scope = list(lower = red_model, upper = full_model2), k = log(n), trace = 0)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -5.95817      0.25853      0.08802      0.09859      0.11274      0.22040
##    TOVRATIO         USG.         REB.         PACE       AST.TO
##    -0.16354      0.19911      0.22389     -0.05475      0.41897
```

After using AIC and BIC and multiple step methods to determine a sufficient model to predict PIE, we notice that not a single one utilizes the social media variables in the model. We can interpret this as saying that the social media variables have less influence in predicting PIE. From this we notice that the model with the least amount of predictors that is not just the intercept is model 8, so we will continue with this as our "best model".From here I will incorporate the social media variables by creating two other possible models, one simply adding them to our best model and one model only utilizing the social media variables. First I will check the significance of the predictors in their respective models.

```
test1 <- lm(PIE~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO + USG. + REB. +
              PACE + AST.TO, data = NBA)

summary(test1)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE + AST.TO, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.1768 -0.4532  0.0368  0.4855  2.5332
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -5.95817    2.31440  -2.574  0.01067 *
## TwoPM        0.25853    0.05258   4.917 1.68e-06 ***
## RPM          0.08802    0.03139   2.804  0.00548 **
## DREB.        0.09859    0.03478   2.835  0.00500 **
## ASTPERC      0.11274    0.01625   6.937 4.09e-11 ***
## TS.          0.22040    0.01168  18.863  < 2e-16 ***
## TOVRATIO    -0.16354    0.03240  -5.047 9.16e-07 ***
## USG.         0.19911    0.02349   8.476 2.93e-15 ***
## REB.         0.22389    0.04889   4.580 7.66e-06 ***
## PACE        -0.05475    0.02136  -2.563  0.01102 *
## AST.TO       0.41897    0.18113   2.313  0.02161 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8574 on 228 degrees of freedom
## Multiple R-squared:  0.9454, Adjusted R-squared:  0.943
## F-statistic: 394.6 on 10 and 228 DF,  p-value: < 2.2e-16
```

After looking at the significance of predictors, AST.TO is the least significant variable in our first model so I will remove it, and run 4-fold cross validation with this model.

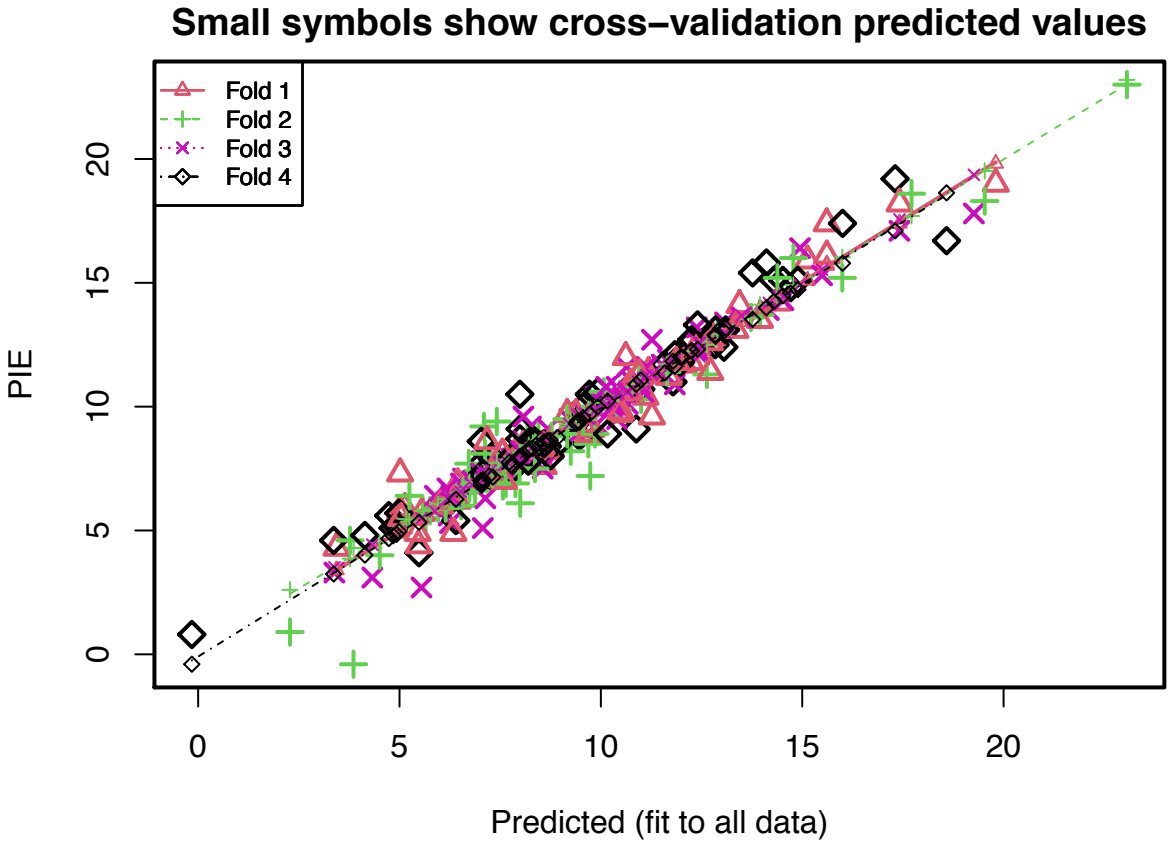
```
set.seed(1)
train.control<-trainControl(method = "cv", number = 4)

DIM_model <- train(PIE~TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO + USG. +
                     REB. + PACE, data = NBA,
                   method = "lm",
                trControl = train.control)
print(DIM_model)
```

```
## Linear Regression
##
## 239 samples
##   9 predictor
##
## No pre-processing
## Resampling: Cross-Validated (4 fold)
## Summary of sample sizes: 179, 179, 180, 179
## Resampling results:
##
##   RMSE       Rsquared   MAE
##   0.9495762  0.9334022  0.6712875
```

```
##
## Tuning parameter 'intercept' was held constant at a value of TRUE
```

```
DIM_cv <- cv.lm(NBA, form.lm = formula(PIE~TwoPM + RPM + DREB. +
                                       ASTPERC + TS. + TOVRATIO + USG. + REB. + PACE)
                , m =4, seed =1 , printit = FALSE)
```

```
## Warning in cv.lm(NBA, form.lm = formula(PIE ~ TwoPM + RPM + DREB. + ASTPERC + :
##
##  As there is >1 explanatory variable, cross-validation
##  predicted values for a fold are not a linear function
##  of corresponding overall predicted values.  Lines that
##  are shown for the different folds are approximate
```



With these results it becomes especially clear from the graphs, although approximated, and the average MSE (ms at bottom of each output) that this model is virtually identical in performance to our previous two models. This means that, overall, social media appears to be insignificant for predicting performance of an NBA player, or at least insignificant compared to in game statistics. From this, we can say that our original "best model" without utilizing AST.TO should be the model used for further predictions as it uses less predictors resulting in a slightly less complex and easier to interpret model.

```
#want to double checkconstant variance criteria within our model
DIMtest<- lm(PIE~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO + USG. +
               REB. + PACE, data = NBA)
ncvTest(DIMtest)
```

```
## Non-constant Variance Score Test
## Variance formula: ~ fitted.values
## Chisquare = 2.924866, Df = 1, p = 0.087224
```

ncvTest tells us that our model has a constant variance, and therefore does not require any weighted least squares added to our model.

```
#From this we conclude that our final model is
DIM_model$finalModel
```

```
##
## Call:
## lm(formula = .outcome ~ ., data = dat)
##
## Coefficients:
## (Intercept)        TwoPM          RPM        DREB.      ASTPERC          TS.
##    -3.96441      0.26620      0.10212      0.09592      0.14410      0.21686
##    TOVRATIO         USG.         REB.         PACE
##    -0.21780      0.16272      0.21906     -0.05686
```

Okay from here we showed what our best possible model is with the data that we have. Now we want to see just how accurate a model solely based on salary and social media can be compared to this best model.

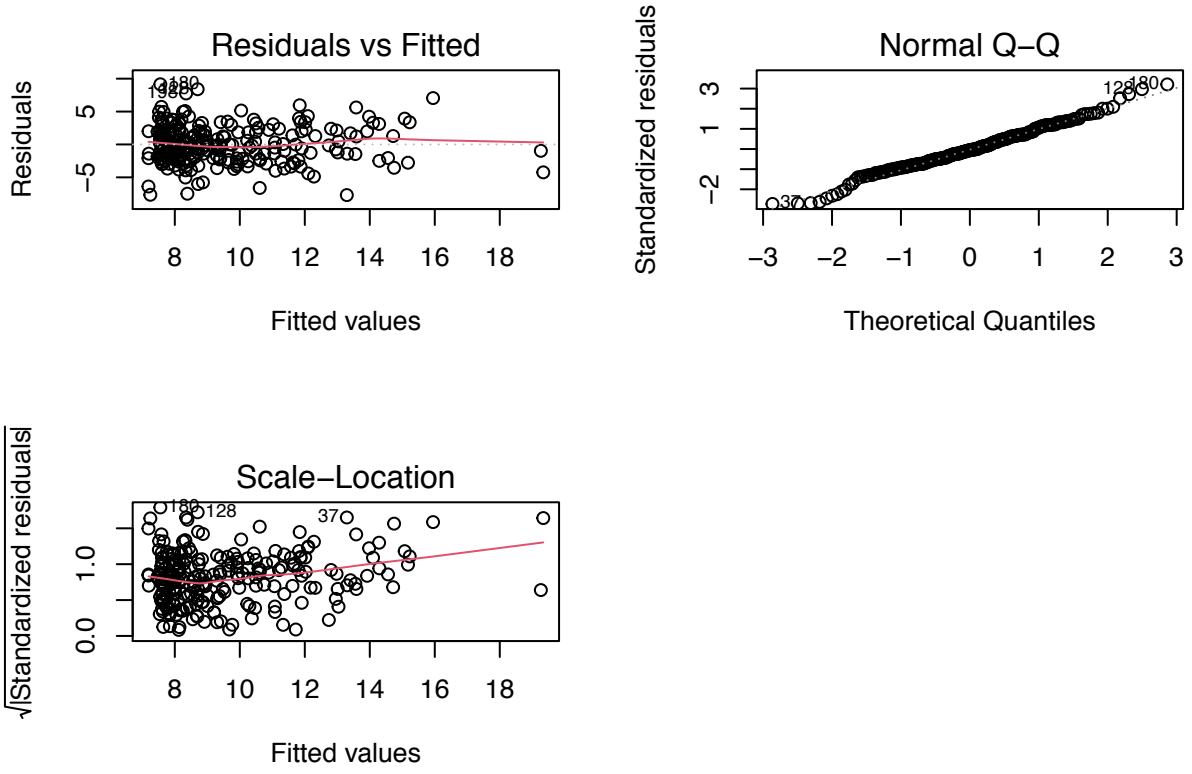
```
#recall from above that test2 was our prospective model with only
#salary and social predictors
test2<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
            TWITTER_RETWEET_COUNT, data = NBA)
summary(test2)
```

```
##
## Call:
## lm(formula = PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.6937 -2.0457 -0.1536  1.8740  9.1449
##
## Coefficients:
##                          Estimate Std. Error t value Pr(>|t|)
## (Intercept)             7.180e+00  2.848e-01  25.210   <2e-16 ***
## SALARY_MILLIONS         2.759e-01  3.047e-02   9.057   <2e-16 ***
## PAGEVIEWS              -6.907e-05  2.133e-04  -0.324    0.746
## TWITTER_FAVORITE_COUNT  7.727e-04  5.438e-04   1.421    0.157
## TWITTER_RETWEET_COUNT   1.879e-04  1.271e-03   0.148    0.883
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.862 on 234 degrees of freedom
## Multiple R-squared:  0.3751, Adjusted R-squared:  0.3644
## F-statistic: 35.11 on 4 and 234 DF,  p-value: < 2.2e-16
```

```
anova(test2)
```

```
## Analysis of Variance Table
##
## Response: PIE
##                           Df  Sum Sq Mean Sq  F value    Pr(>F)
## SALARY_MILLIONS            1 1021.79 1021.79 124.7162 < 2.2e-16 ***
## PAGEVIEWS                  1   72.67   72.67   8.8704  0.003203 **
## TWITTER_FAVORITE_COUNT     1   56.12   56.12   6.8501  0.009442 **
## TWITTER_RETWEET_COUNT      1    0.18    0.18   0.0219  0.882593
## Residuals                234 1917.15    8.19
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
par(mfrow=c(2,2))
plot(test2, which = c(1,2,3))
```



Overall, with just these predictors alone in simple multiple linear regression, we end up with a relatively poor model for predicting PIE, so I will take steps to try to improve this model.

```
#Test for possible interaction terms
#PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
#TWITTER_RETWEET_COUNT, data = NBA
#all posible
```

```
int1.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT + SALARY_MILLIONS*PAGEVIEWS +
              SALARY_MILLIONS*TWITTER_FAVORITE_COUNT +
              SALARY_MILLIONS*TWITTER_RETWEET_COUNT +
              PAGEVIEWS*TWITTER_FAVORITE_COUNT +
              PAGEVIEWS*TWITTER_RETWEET_COUNT +
              TWITTER_FAVORITE_COUNT*TWITTER_RETWEET_COUNT,
            data = NBA)
#no salary
int2.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT +
              PAGEVIEWS*TWITTER_FAVORITE_COUNT +
              PAGEVIEWS*TWITTER_RETWEET_COUNT +
              TWITTER_FAVORITE_COUNT*TWITTER_RETWEET_COUNT,
            data = NBA)
#no pageview
int3.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT + SALARY_MILLIONS*TWITTER_FAVORITE_COUNT +
              SALARY_MILLIONS*TWITTER_RETWEET_COUNT +
              TWITTER_FAVORITE_COUNT*TWITTER_RETWEET_COUNT,
            data = NBA)
#no twitter favorite
int4.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT + SALARY_MILLIONS*PAGEVIEWS +
              SALARY_MILLIONS*TWITTER_RETWEET_COUNT +
              PAGEVIEWS*TWITTER_RETWEET_COUNT,
            data = NBA)
#no twitter retweet
int5.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT + SALARY_MILLIONS*PAGEVIEWS +
              SALARY_MILLIONS*TWITTER_FAVORITE_COUNT +
              PAGEVIEWS*TWITTER_FAVORITE_COUNT,
            data = NBA)
#no salary or pageview
int6.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT +
              TWITTER_FAVORITE_COUNT*TWITTER_RETWEET_COUNT,
            data = NBA)
#no twitters
int7.lm<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
              TWITTER_RETWEET_COUNT +
              SALARY_MILLIONS*PAGEVIEWS,
            data = NBA)
anova(test2, int1.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + SALARY_MILLIONS * PAGEVIEWS + SALARY_MILLIONS *
##     TWITTER_FAVORITE_COUNT + SALARY_MILLIONS * TWITTER_RETWEET_COUNT +
##     PAGEVIEWS * TWITTER_FAVORITE_COUNT + PAGEVIEWS * TWITTER_RETWEET_COUNT +
```

```
##     TWITTER_FAVORITE_COUNT * TWITTER_RETWEET_COUNT
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    234 1917.2
## 2    228 1738.2  6    178.98 3.9128 0.0009666 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int2.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT +
##     PAGEVIEWS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
##     TWITTER_RETWEET_COUNT
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    234 1917.2
## 2    231 1749.1  3    168.08 7.3996 9.38e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int3.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + SALARY_MILLIONS * TWITTER_FAVORITE_COUNT +
##     SALARY_MILLIONS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
##     TWITTER_RETWEET_COUNT
##   Res.Df    RSS Df Sum of Sq      F    Pr(>F)
## 1    234 1917.2
## 2    231 1785.1  3    132.02 5.6946 0.0008882 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int4.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + SALARY_MILLIONS * PAGEVIEWS + SALARY_MILLIONS *
##     TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_RETWEET_COUNT
##   Res.Df    RSS Df Sum of Sq     F  Pr(>F)
## 1    234 1917.2
## 2    231 1798.8  3    118.37 5.067 0.00204 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int5.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + SALARY_MILLIONS * PAGEVIEWS + SALARY_MILLIONS *
##     TWITTER_FAVORITE_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    234 1917.2
## 2    231 1801.7  3     115.5 4.9362 0.002426 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int6.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT * TWITTER_RETWEET_COUNT
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    234 1917.2
## 2    233 1838.7  1    78.407 9.9355 0.001834 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
anova(test2, int7.lm)
```

```
## Analysis of Variance Table
##
## Model 1: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT
## Model 2: PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + SALARY_MILLIONS * PAGEVIEWS
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    234 1917.2
## 2    233 1915.5  1    1.6814 0.2045 0.6515
```

Of these possible models with interaction terms, int2.lm has the lowest p-value so we will continue with this model as our best moving forward.

```
#looking for possible transformations of data
#need PIE to be strictly positive to test for transformation,
#PIE only has one negative value so make it 0.1 for this test real quick
pie<-NBA$PIE
pie[147]=0.1
int2test<-lm(pie~NBA$SALARY_MILLIONS + NBA$PAGEVIEWS + NBA$TWITTER_FAVORITE_COUNT +
               NBA$TWITTER_RETWEET_COUNT + NBA$PAGEVIEWS*NBA$TWITTER_FAVORITE_COUNT +
```

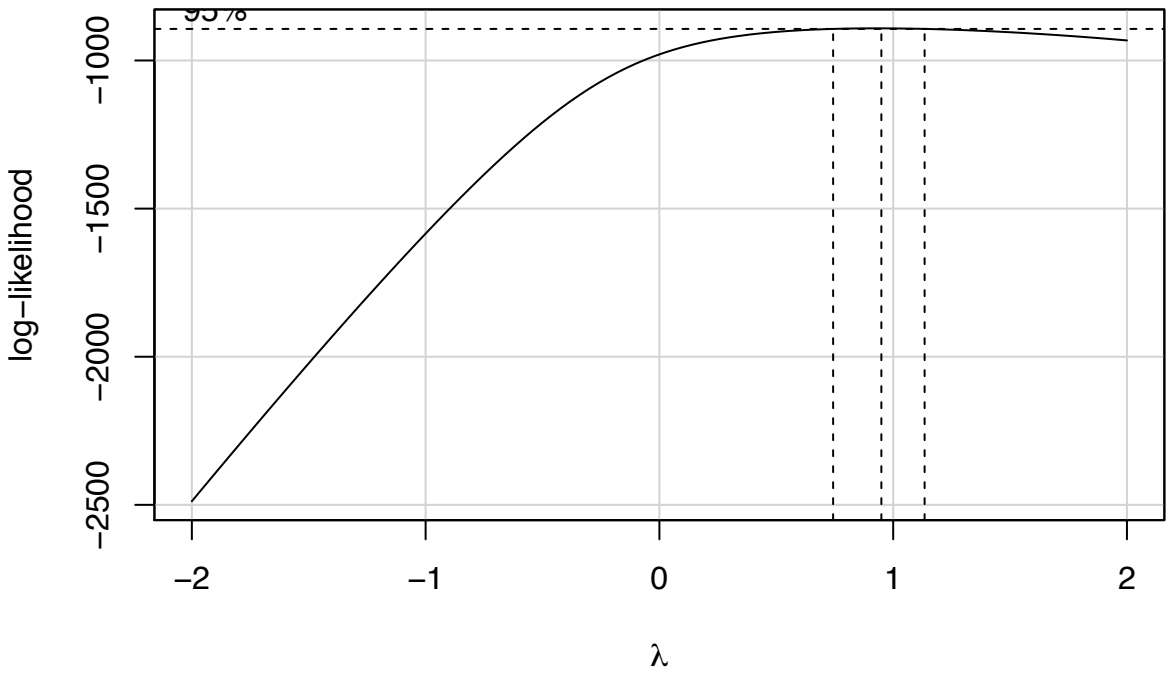
```
              NBA$PAGEVIEWS*NBA$TWITTER_RETWEET_COUNT +
              NBA$TWITTER_FAVORITE_COUNT*NBA$TWITTER_RETWEET_COUNT)
summary(powerTransform(int2test))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.9296           1       0.7341        1.125
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df       pval
## LR test, lambda = (0) 175.7143  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                             LRT df   pval
## LR test, lambda = (1) 0.482481  1 0.4873
```

```
summary(powerTransform(pie~1, NBA))
```

```
## bcPower Transformation to Normality
##    Est Power Rounded Pwr Wald Lwr Bnd Wald Upr Bnd
## Y1    0.7103        0.71       0.5305       0.8901
##
## Likelihood ratio test that transformation parameter is equal to 0
##  (log transformation)
##                             LRT df       pval
## LR test, lambda = (0) 112.5598  1 < 2.22e-16
##
## Likelihood ratio test that no transformation is needed
##                             LRT df      pval
## LR test, lambda = (1) 8.451276  1 0.0036479
```

```
boxCox(int2test)
```



Box-Cox tells us that the best exponent for transformation is 1, which just means that we don't need to add any transformation.

```
#testing validity of adding weighted least squares
weight_model<-lm(PIE~SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
                   TWITTER_RETWEET_COUNT + PAGEVIEWS*TWITTER_FAVORITE_COUNT +
                   PAGEVIEWS*TWITTER_RETWEET_COUNT +
                   TWITTER_FAVORITE_COUNT*TWITTER_RETWEET_COUNT,
                 data = NBA, weights = int2.lm$fitted.values)
summary(int2.lm)
```

```
##
## Call:
## lm(formula = PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT +
##     PAGEVIEWS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
##     TWITTER_RETWEET_COUNT, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5078 -1.7901 -0.0943  1.7697  9.3931
##
## Coefficients:
##                                          Estimate Std. Error t value
## (Intercept)                             6.980e+00  2.932e-01  23.807
## SALARY_MILLIONS                         2.292e-01  3.114e-02   7.362
```

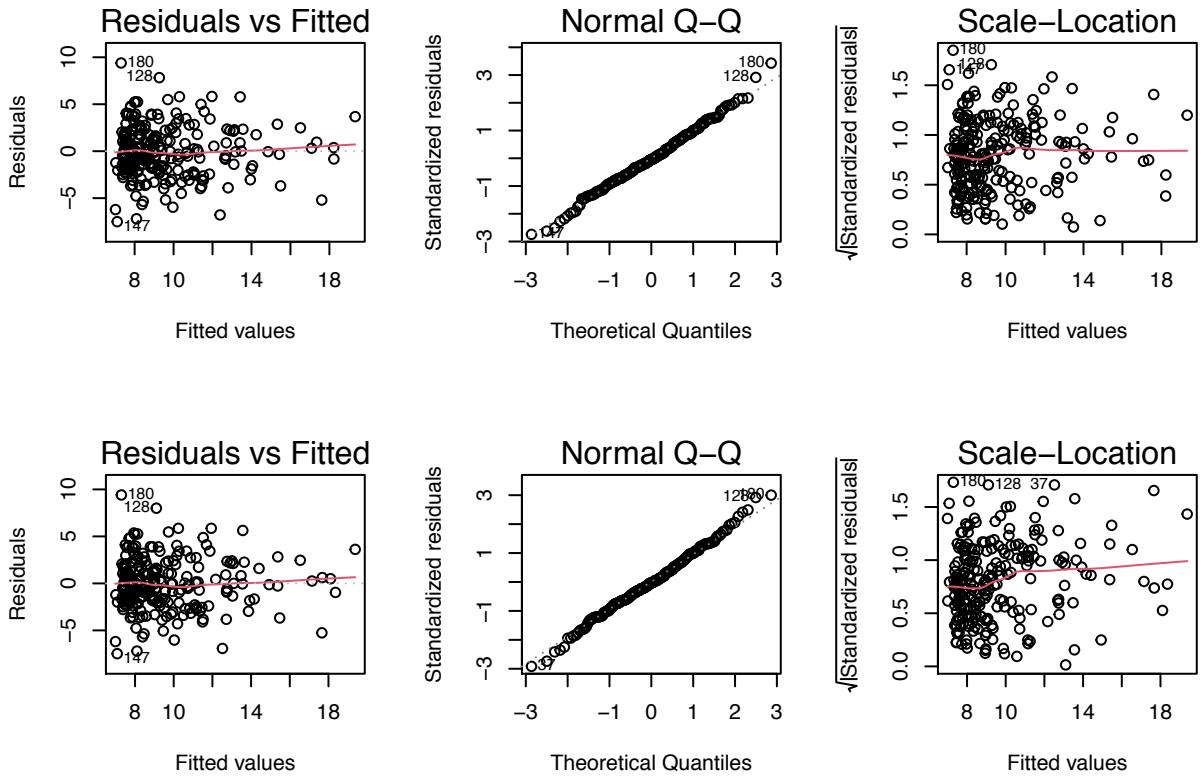
```
## PAGEVIEWS                                        2.257e-04  3.085e-04   0.732
## TWITTER_FAVORITE_COUNT                           1.059e-04  1.497e-03   0.071
## TWITTER_RETWEET_COUNT                            1.310e-02  5.743e-03   2.281
## PAGEVIEWS:TWITTER_FAVORITE_COUNT                 1.031e-06  4.182e-07   2.466
## PAGEVIEWS:TWITTER_RETWEET_COUNT                 -4.260e-06  1.573e-06  -2.708
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT -1.221e-06  4.833e-07  -2.527
##                                                 Pr(>|t|)
## (Intercept)                                      < 2e-16 ***
## SALARY_MILLIONS                                 3.16e-12 ***
## PAGEVIEWS                                        0.46507
## TWITTER_FAVORITE_COUNT                           0.94368
## TWITTER_RETWEET_COUNT                            0.02349 *
## PAGEVIEWS:TWITTER_FAVORITE_COUNT                 0.01441 *
## PAGEVIEWS:TWITTER_RETWEET_COUNT                  0.00727 **
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT     0.01216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.752 on 231 degrees of freedom
## Multiple R-squared:  0.4299, Adjusted R-squared:  0.4126
## F-statistic: 24.88 on 7 and 231 DF,  p-value: < 2.2e-16
```

```
summary(weight_model)
```

```
##
## Call:
## lm(formula = PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT +
##     PAGEVIEWS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
##     TWITTER_RETWEET_COUNT, data = NBA, weights = int2.lm$fitted.values)
##
## Weighted Residuals:
##     Min      1Q  Median      3Q     Max
## -24.354  -5.331  -0.226   5.402  25.426
##
## Coefficients:
##                                                  Estimate Std. Error t value
## (Intercept)                                     6.958e+00  3.048e-01  22.824
## SALARY_MILLIONS                                 2.377e-01  2.941e-02   8.082
## PAGEVIEWS                                       1.683e-04  2.732e-04   0.616
## TWITTER_FAVORITE_COUNT                          5.271e-04  1.293e-03   0.408
## TWITTER_RETWEET_COUNT                           1.150e-02  5.093e-03   2.258
## PAGEVIEWS:TWITTER_FAVORITE_COUNT                9.212e-07  3.647e-07   2.526
## PAGEVIEWS:TWITTER_RETWEET_COUNT                -3.835e-06  1.373e-06  -2.794
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT -1.153e-06  4.133e-07  -2.789
##                                                 Pr(>|t|)
## (Intercept)                                      < 2e-16 ***
## SALARY_MILLIONS                                 3.54e-14 ***
## PAGEVIEWS                                        0.53832
## TWITTER_FAVORITE_COUNT                           0.68386
## TWITTER_RETWEET_COUNT                            0.02487 *
## PAGEVIEWS:TWITTER_FAVORITE_COUNT                 0.01220 *
## PAGEVIEWS:TWITTER_RETWEET_COUNT                  0.00564 **
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT     0.00572 **
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 8.503 on 231 degrees of freedom
## Multiple R-squared:  0.5114, Adjusted R-squared:  0.4966
## F-statistic: 34.54 on 7 and 231 DF,  p-value: < 2.2e-16
```

```
par(mfrow=c(2,3))
plot(int2.lm, which = c(1,2,3))
plot(weight_model, which = c(1,2,3))
```



Of these two models, our weighted model has a higher adjusted R-squared value than our int2.lm model , but has significantly higher variation in residuals, so our int2.lm model appears to be our best possible model when only using salary and social media stats as predictors. This being said, the adjusted R-squared of this model is significantly less than that of our model above (DIMtest) without using these predictors.

```
summary(SOCIALnew)
```

```
##
## Call:
## lm(formula = PIE ~ NETRTG + ASTPERC + AST.TO + OREB. + DREB. +
##     TOVRATIO + TS. + USG. + PACE + TwoPM + PAGEVIEWS + TWITTER_FAVORITE_COUNT,
##     data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
```

```
## -4.4469 -0.4399  0.0389  0.4534  2.3221
##
## Coefficients:
##                           Estimate Std. Error t value Pr(>|t|)
## (Intercept)            -7.046e+00  2.283e+00  -3.086 0.002281 **
## NETRTG                  3.507e-02  9.237e-03   3.796 0.000189 ***
## ASTPERC                 1.214e-01  1.616e-02   7.508 1.37e-12 ***
## AST.TO                  4.091e-01  1.772e-01   2.309 0.021849 *
## OREB.                   9.909e-02  2.424e-02   4.088 6.06e-05 ***
## DREB.                   2.269e-01  1.356e-02  16.738  < 2e-16 ***
## TOVRATIO               -1.697e-01  3.217e-02  -5.274 3.11e-07 ***
## TS.                     2.276e-01  1.051e-02  21.645  < 2e-16 ***
## USG.                    1.817e-01  2.416e-02   7.519 1.28e-12 ***
## PACE                   -4.787e-02  2.132e-02  -2.246 0.025697 *
## TwoPM                   3.139e-01  5.179e-02   6.062 5.59e-09 ***
## PAGEVIEWS              -1.284e-04  5.956e-05  -2.156 0.032173 *
## TWITTER_FAVORITE_COUNT  2.204e-04  9.701e-05   2.272 0.024011 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8417 on 226 degrees of freedom
## Multiple R-squared:  0.9478, Adjusted R-squared:  0.945
## F-statistic: 342.1 on 12 and 226 DF,  p-value: < 2.2e-16
```

```
summary(DIMtest)
```

```
##
## Call:
## lm(formula = PIE ~ TwoPM + RPM + DREB. + ASTPERC + TS. + TOVRATIO +
##     USG. + REB. + PACE, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -4.2577 -0.4345  0.0229  0.4574  2.5109
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) -3.964408   2.168202  -1.828  0.06879 .
## TwoPM        0.266198   0.052975   5.025 1.01e-06 ***
## RPM          0.102120   0.031082   3.285  0.00118 **
## DREB.        0.095919   0.035088   2.734  0.00675 **
## ASTPERC      0.144098   0.009047  15.927  < 2e-16 ***
## TS.          0.216856   0.011693  18.546  < 2e-16 ***
## TOVRATIO    -0.217799   0.022562  -9.654  < 2e-16 ***
## USG.         0.162722   0.017612   9.239  < 2e-16 ***
## REB.         0.219063   0.049303   4.443 1.38e-05 ***
## PACE        -0.056859   0.021544  -2.639  0.00888 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.8655 on 229 degrees of freedom
## Multiple R-squared:  0.9441, Adjusted R-squared:  0.9419
## F-statistic: 429.6 on 9 and 229 DF,  p-value: < 2.2e-16
```

```r
summary(int2.lm)
```

```
##
## Call:
## lm(formula = PIE ~ SALARY_MILLIONS + PAGEVIEWS + TWITTER_FAVORITE_COUNT +
##     TWITTER_RETWEET_COUNT + PAGEVIEWS * TWITTER_FAVORITE_COUNT +
##     PAGEVIEWS * TWITTER_RETWEET_COUNT + TWITTER_FAVORITE_COUNT *
##     TWITTER_RETWEET_COUNT, data = NBA)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -7.5078 -1.7901 -0.0943  1.7697  9.3931
##
## Coefficients:
##                                              Estimate Std. Error t value
## (Intercept)                                 6.980e+00  2.932e-01  23.807
## SALARY_MILLIONS                             2.292e-01  3.114e-02   7.362
## PAGEVIEWS                                   2.257e-04  3.085e-04   0.732
## TWITTER_FAVORITE_COUNT                      1.059e-04  1.497e-03   0.071
## TWITTER_RETWEET_COUNT                       1.310e-02  5.743e-03   2.281
## PAGEVIEWS:TWITTER_FAVORITE_COUNT            1.031e-06  4.182e-07   2.466
## PAGEVIEWS:TWITTER_RETWEET_COUNT            -4.260e-06  1.573e-06  -2.708
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT -1.221e-06  4.833e-07  -2.527
##                                             Pr(>|t|)
## (Intercept)                                  < 2e-16 ***
## SALARY_MILLIONS                             3.16e-12 ***
## PAGEVIEWS                                    0.46507
## TWITTER_FAVORITE_COUNT                       0.94368
## TWITTER_RETWEET_COUNT                        0.02349 *
## PAGEVIEWS:TWITTER_FAVORITE_COUNT             0.01441 *
## PAGEVIEWS:TWITTER_RETWEET_COUNT              0.00727 **
## TWITTER_FAVORITE_COUNT:TWITTER_RETWEET_COUNT  0.01216 *
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 2.752 on 231 degrees of freedom
## Multiple R-squared:  0.4299, Adjusted R-squared:  0.4126
## F-statistic: 24.88 on 7 and 231 DF,  p-value: < 2.2e-16
```

Comparing these three models, it becomes clear that, although social media can be used as a sufficient tool for predicting the performance (PIE) of a player when combined with in game stats, in game statistics alone are just as accurate at predicting PIE, and utilizes a less complex model to do so.

Logistic Regression Method

Setup

The following chunk downloads the pie and twitter csv files, and removes the PIE variables:

```
nba_pie_twitter <- merge(nba_pie, nba_twitter, by='PLAYER', all=TRUE)
# remove duplicate columns
nba_pie_twitter <- nba_pie_twitter[,-grep('.y',names(nba_pie_twitter))]
colnames(nba_pie_twitter) <- gsub('.x','',colnames(nba_pie_twitter))

#write.csv(nba_pie_twitter,
#          "~/Documents/School Assignments Folder/2019-2020 School Year/PSTAT 131/Final Project/nba_2

# Creates dataframe without names (used for logistic regression functions)
nba_pt <- select(nba_pie_twitter, -c('PLAYER','POSITION','TEAM','POINTS','FG','FT',
                                     'FGA','FTA','ORB','DRB','AST','STL','BLK','PF',
                                     'TOV'))

# NOTE: SUPER IMPORTANT to list the levels in order (so glm makes 1=high, 0=low)
nba_pt['PIE_RESPONSE'] <- factor(ifelse(nba_pt$PIE>=mean(nba_pt$PIE),'high','low'),
                                 order=TRUE, levels=c('low','high'))

grps <- 4 # 4 folds / groups
response <- 'PIE_RESPONSE'
```

Then, the data is split into 4 groups for 4-fold cross validation.

```
set.seed(69)
id <- cut(1:nrow(nba_pt),breaks=grps,labels=FALSE) %>% sample()
resp_col <- which(colnames(nba_pt) == response)

# replaces NA values with mean of that column.
for (i in colnames(nba_pt)) {
  if (i != response) {
    nba_pt[,i] = nba_pt[,i] %>% replace_na(mean(nba_pt[,i], na.rm=T))
  }
}

# Data Frame containing training and test error for each predictor along with its formula
error_df <- data.frame(matrix(NA,nrow=3,ncol=2),
                       row.names=c('Twitter','Salary','No additions'))
colnames(error_df) <- c('train.error','test.error')
error_df['formula'] <- c('','','')
```

Each method mentioned above takes said variables, and puts them into an initial glm model, so that a forward step function can be used to add the least amount of additional variables necessary for the propsed `best_formula`. The prediction function takes the `best_formula` and `method` used (Twitter, Salary, None), and runs a logistic regression model on the formula, calculating the average training and test error between the 4 folds.

Logistic Regression

The next method used is logistic regression. Excluding the variables used to calculate PIE, we found the best predictors for the logistic regression formula, and then added Twitter or Salary-based predictors to see

if a better test rate could be achieved.

The following functions produce the best logistic regression formula, and calculate the error rates for each method:

```
# This will determine which terms create the best model for PIE
best_formula_func <- function(starting_vars) {
  starting_vars <- as.formula(paste0('PIE ~ ', starting_vars))

  test0.glm <- glm(PIE ~ 1, data=nba_pt)
  testN.glm <- glm(PIE ~ . - PIE_RESPONSE, data=nba_pt)
  test.glm <- glm(starting_vars, data=nba_pt)

  best_model <- step(test.glm, scope=list(lower=test0.glm, upper=testN.glm),
                     direction='forward', trace=0)
  best_formula <- best_model$formula
  best_formula <- as.formula(paste0(response,' ~ ',
                                    gsub('~, PIE, ','',toString(best_formula))))
  return(best_formula)
}
# This will predict the training and test error rates for the given formula / method.
predict_func <- function(formula, method) {
  train.error <- 0
  test.error <- 0

  if ((is_formula(formula)) == FALSE) { formula <- as.formula(formula) }

  for (i in 1:grps) {
    train <- (i != id)
    nba_pt.train <- nba_pt[train,]
    nba_pt.test <- nba_pt[!train,]

    Ytr <- factor(nba_pt.train[,response], order=T, levels=c('low','high'))
    Yvl <- factor(nba_pt.test[,response], order=T, levels=c('low','high'))

    twitter.glm <- glm(formula, data=nba_pt.train, family=binomial(link="logit"))
    if (i==grps) {
      print(paste0("The coefficients (and variables) for method = ", method, " when id = ",
                   i, ":"))
      print(twitter.glm$coefficients)
    }
    predYtr <- predict.glm(twitter.glm, nba_pt.train, type='response')
    predYtr <- factor(ifelse(predYtr>=0.5,'high','low'), order=TRUE, levels=c('low','high'))

    predYvl <- predict.glm(twitter.glm, nba_pt.test, type='response')
    predYvl <- factor(ifelse(predYvl>=0.5,'high','low'), order=TRUE, levels=c('low','high'))

    train.error <- train.error + mean(predYtr != Ytr)
    test.error <- test.error + mean(predYvl != Yvl)
  }
  error_df[method,] <- c(train.error/grps, test.error/grps,
                         gsub('~, PIE_RESPONSE,','', toString(formula)))
  return(error_df)
}
```

The following methods categorize the PIE into two categories: 'high' and 'low'. The high PIE values (the ones equal to or higher than the full dataset's PIE mean), and the low PIE values (the ones less than the full dataset's PIE mean). These logistic regression models will predict whether or not a player falls on the high end or the low end of the PIE values.

For the first logistic regression formula, we will test the Twitter and Wikipedia variables and their relation to PIE.

```
str <- 'TWITTER_FAVORITE_COUNT + TWITTER_RETWEET_COUNT + PAGEVIEWS'
best_formula <- best_formula_func(str)
error_df <- predict_func(best_formula,'Twitter')
```

```
## [1] "The coefficients (and variables) for method = Twitter when id = 4:"
##            (Intercept) TWITTER_FAVORITE_COUNT  TWITTER_RETWEET_COUNT
##          -5.042582e+01           6.527610e-03          -2.016040e-02
##              PAGEVIEWS                    TS.                   USG.
##           1.914534e-04           3.953072e-01           6.853803e-01
##                  DREB.                   AST.               TO.RATIO
##           4.122972e-01           2.511730e-01          -1.330276e-01
##                NETRTG                    TRB                   REB.
##           1.008851e-01           5.023192e-01           1.235712e-01
##                   MIN                    MPG                    X2P
##           4.008688e-02          -1.224105e-01           4.940177e-01
##                 OREB.                    EFG.                   PACE
##           2.913709e-01           7.363703e-02          -3.209443e-03
##               AST.TO              AST.RATIO        SALARY_MILLIONS
##           2.231853e+00          -6.844800e-02           2.251229e-02
##                   FT.                   DRPM                      L
##          -2.106730e+00          -1.816526e-01           2.921754e-02
```

The second logistic regression formula will test the impact of a player's salary in relation to PIE.

```
str <- 'SALARY_MILLIONS'
best_formula <- best_formula_func(str)
error_df <- predict_func(best_formula, 'Salary')
```

```
## [1] "The coefficients (and variables) for method = Salary when id = 4:"
##     (Intercept) SALARY_MILLIONS             TS.            USG.           DREB.
##    -48.88471000      0.03888011      0.39525183      0.70145427      0.36289499
##            AST.        TO.RATIO          NETRTG             TRB            REB.
##      0.21139155     -0.13364681      0.09864255      0.43679149      0.21921133
##             MIN             MPG             X2P            EFG.           OREB.
##      0.04168964     -0.10598720      0.36546404      0.06615545      0.23975735
##            PACE          AST.TO       AST.RATIO             FT.            DRPM
##     -0.01239693      2.21550174     -0.03423393     -2.72267580     -0.14005051
##               L
##      0.02582863
```

The final logistic regression formula is the control group of the three models, analyzing the difference in accuracy if no initial variables were added.

```
str <- '1'
best_formula <- best_formula_func(str)
error_df <- predict_func(best_formula,'No additions')
```

```
## [1] "The coefficients (and variables) for method = No additions when id = 4:"
##   (Intercept)           TS.          USG.         DREB.          AST.
## -49.032331759   0.398427708   0.690240060   0.387160561   0.220700631
##      TO.RATIO        NETRTG           TRB          REB.           MIN
##  -0.140598047   0.099694070   0.423649619   0.167044602   0.038905698
##           MPG           X2P         OREB.          EFG.          PACE
##  -0.103993378   0.430626630   0.260330886   0.063117396  -0.008467829
##        AST.TO     AST.RATIO           FT.          DRPM             L
##   2.142168980  -0.037819935  -2.345670075  -0.088224873   0.026216945
```

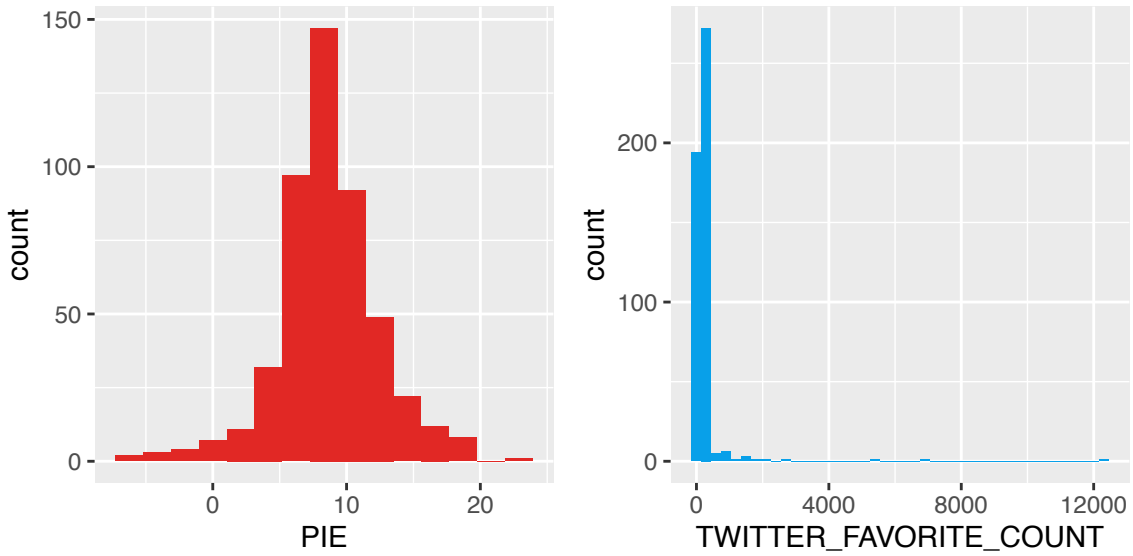
These are the formulas for each method, and their training / test accuracy. Based on accuracy, Salary does the best job out of all three methods in predicting PIE. This is likely, because players who perform better will be given more money to stay with their team (or move to a new one). Surpringly, no additions has the worst training error, and the second worst testing error. However, since all error values (for each column) are within 1 percent, there is not a big difference between adding, and leaving Twitter / salary variables out. They give the data slightly better accuracy, but the third method's predictors already do a reasonably accurate job predicting PIE.

```
## [1] "The training and test errors for each method: "
```

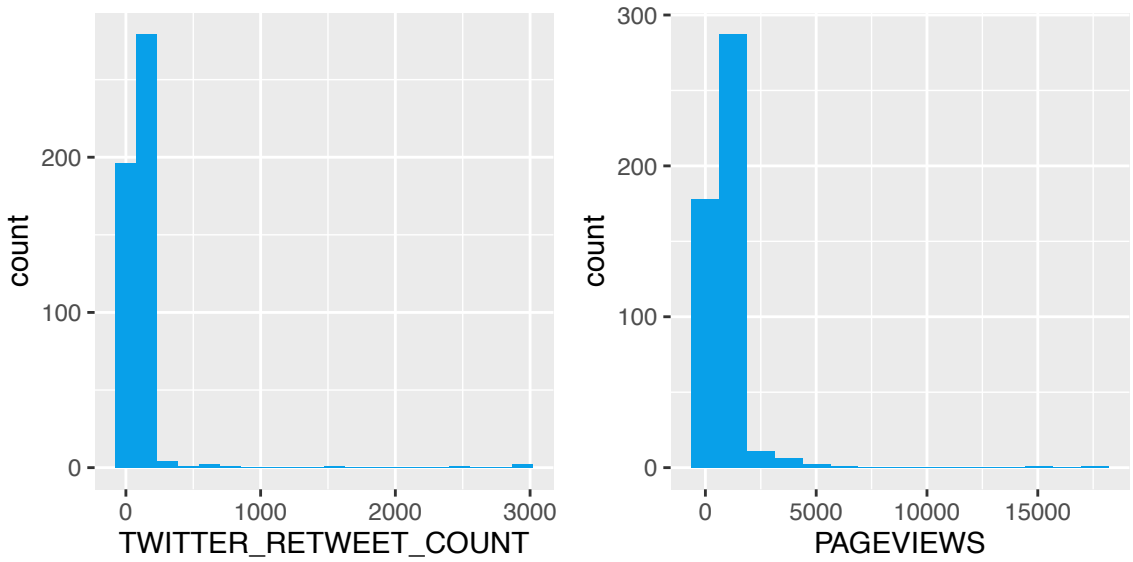
```
##                     train.error        test.error
## Twitter      0.0862527135264616  0.12533870749221
## Salary       0.0876225765401602 0.119174231133993
## No additions 0.0924152256905457 0.121223411461862
```

The graphs below compare histogram frequncies of Player Impact Estimate (PIE) to Twitter, Wikipedia (PAGEVIEWS), and salary values.

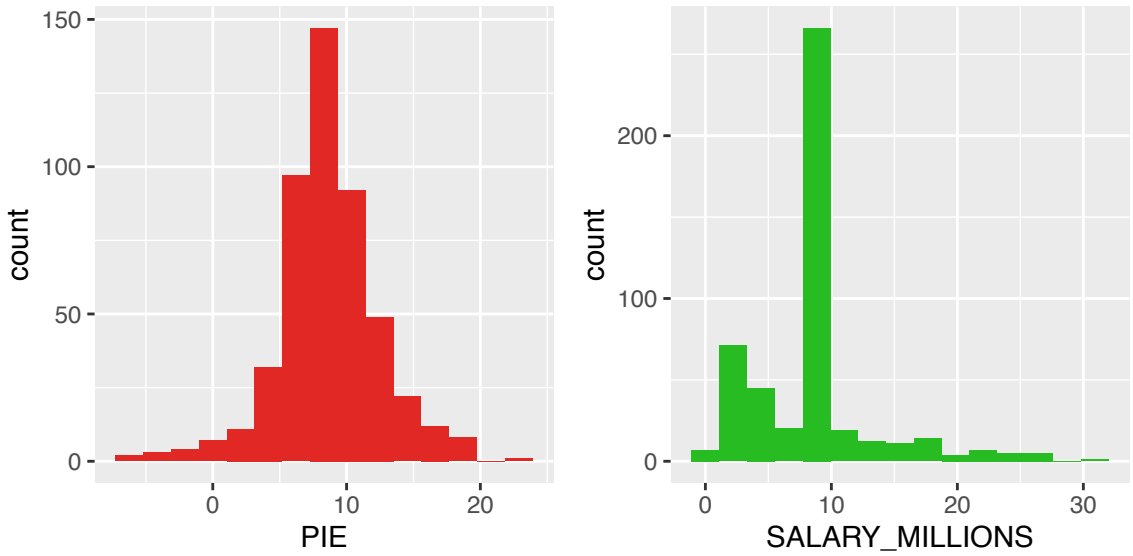
```
plot1 <- ggplot(data=nba_pt, aes(x=PIE)) + geom_histogram(fill='#e12825', bins=15)
plot2 <- ggplot(data=nba_pt, aes(x=TWITTER_FAVORITE_COUNT)) +
  geom_histogram(fill='#08a0e9', binwidth=300)
plot3 <- ggplot(data=nba_pt, aes(x=TWITTER_RETWEET_COUNT)) +
  geom_histogram(fill='#08a0e9', bins=20)
plot4 <- ggplot(data=nba_pt, aes(x=PAGEVIEWS)) + geom_histogram(fill='#08a0e9', bins=15)

grid.arrange(plot1, plot2, ncol=2, nrow=1)
```




```
grid.arrange(plot3, plot4, ncol=2, nrow=1)
```




```
plot1 <- ggplot(data=nba_pt, aes(x=PIE)) + geom_histogram(fill='#e12825', bins=15)
plot2 <- ggplot(data=nba_pt, aes(x=SALARY_MILLIONS)) +
  geom_histogram(fill='#27bd22', bins=15)

grid.arrange(plot1, plot2, ncol=2, nrow=1)
```



Twitter and Wikipedia appear to have a far different distribution than PIE, likely because a majority of NBA players do not have Twitter accounts. As a result, the most values are around the mean value for each category, depicting a chi-square distribution in terms of graph shape. Wikipedia page views also have a similar distribution, and that might occur due to certain players being far more noticed than others based on branding, sponsorship, star players being highlighted by media, etc. If suffices to say that the frequency distribution for Wikipedia and Twitter differ from that of the PIE distribution.

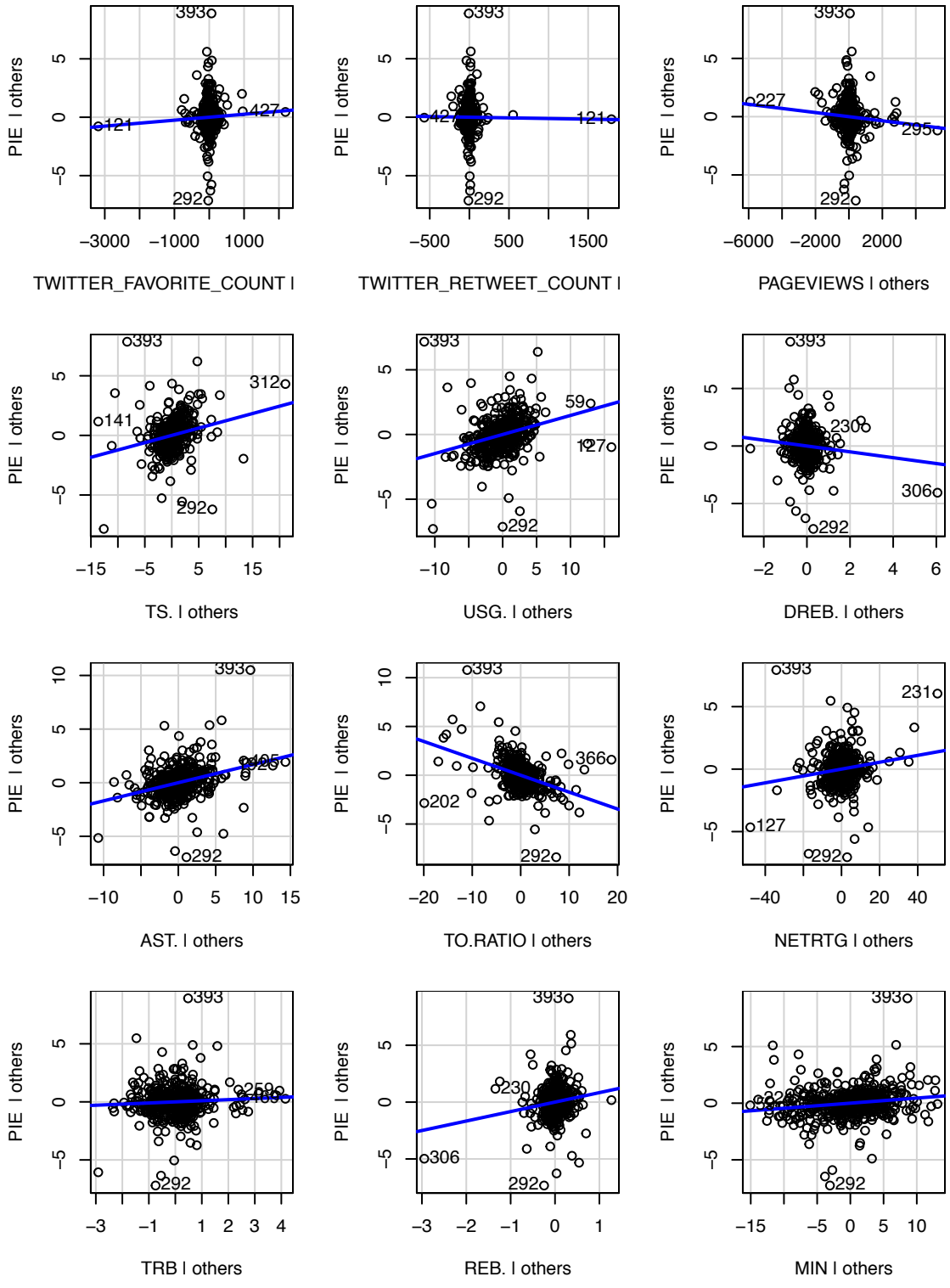
Salary values also seem to be skewed right compared to the visually normal (yet slightly left skewed) distribution of PIE, because the most frequent values appear around 10 million, and more values appaer on the left side of 10 million in a somewhat ascending order than on the right side. It seems like the distribution of PIE and salary do not correlate, because salary resembles more of a chi-squared distribution (like Twitter and Wikipedia) compared to the more bell-shaped curve of PIE's distribution.

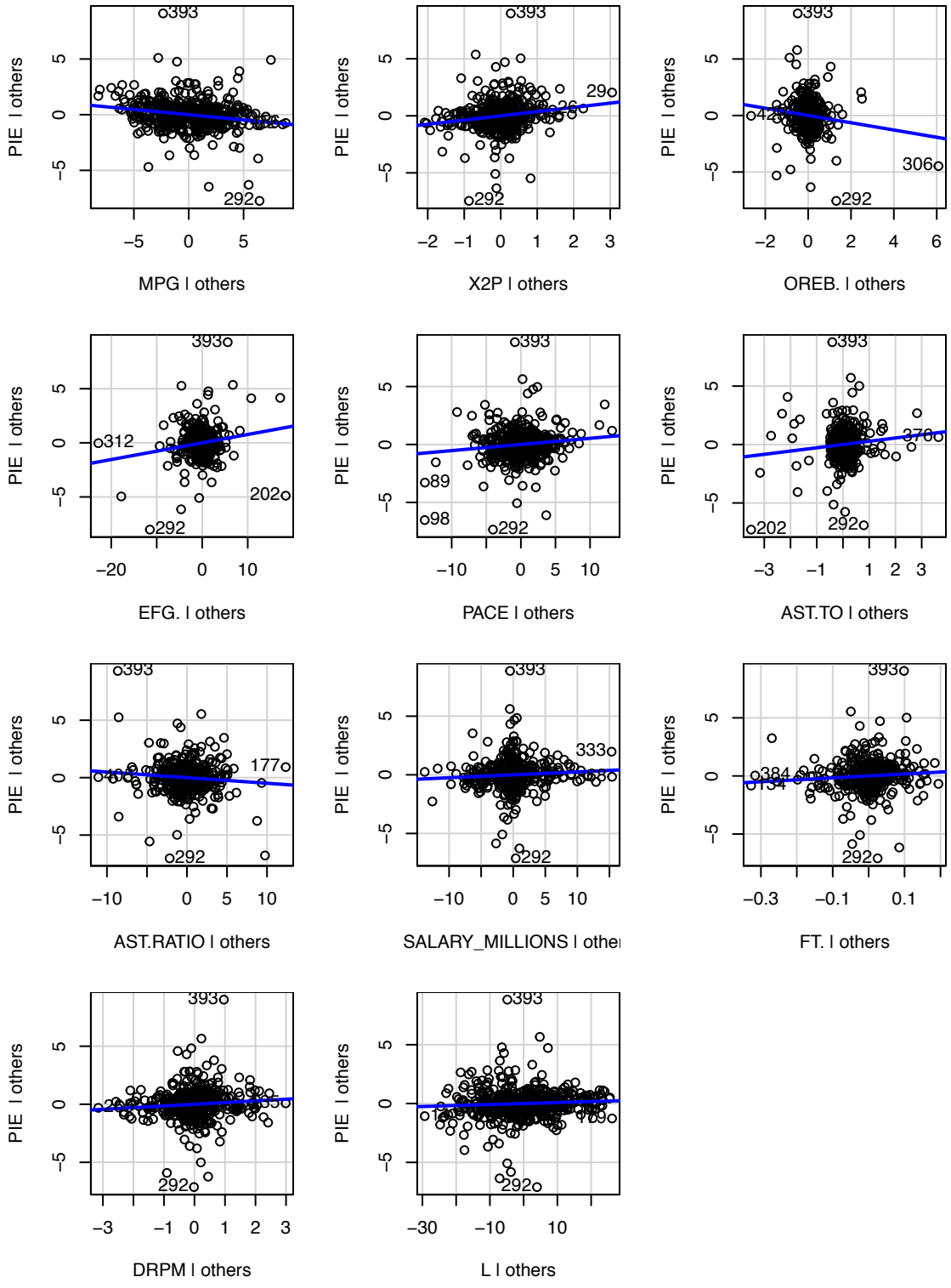
Avplots

The following graphs measure correlation between each variable used in the three formulas above and Player Impact Estimate (PIE). The first avPlot we will test is Twitter's logistic regression formula, and its relation to PIE.

Twitter / Wikipedia (PAGEVIEWS) Logistic Regression

```
three_str <- paste('PIE ~',error_df[,3])
formula1 <- as.formula(three_str[1])
glm1 <- glm(formula1, data=nba_pt)

par(mfrow=c(1,3), mai=c(0.76,0.76,0,0))
avPlots(glm1, layout=NA)
```


PIE | others
TWITTER_FAVORITE_COUNT |
TWITTER_RETWEET_COUNT |
PAGEVIEWS | others
TS. | others
USG. | others
DREB. | others
AST. | others
TO.RATIO | others
NETRTG | others
TRB | others
REB. | others
MIN | others

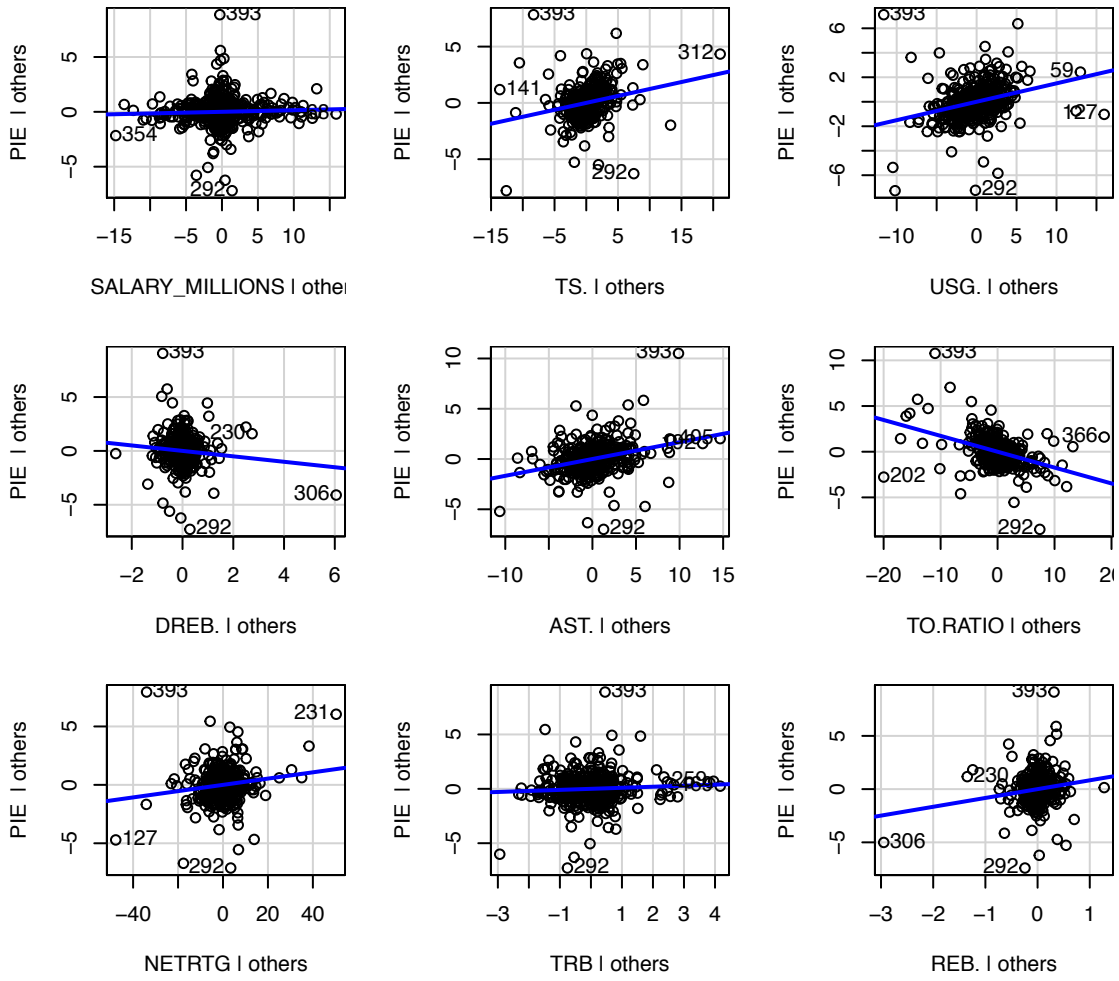


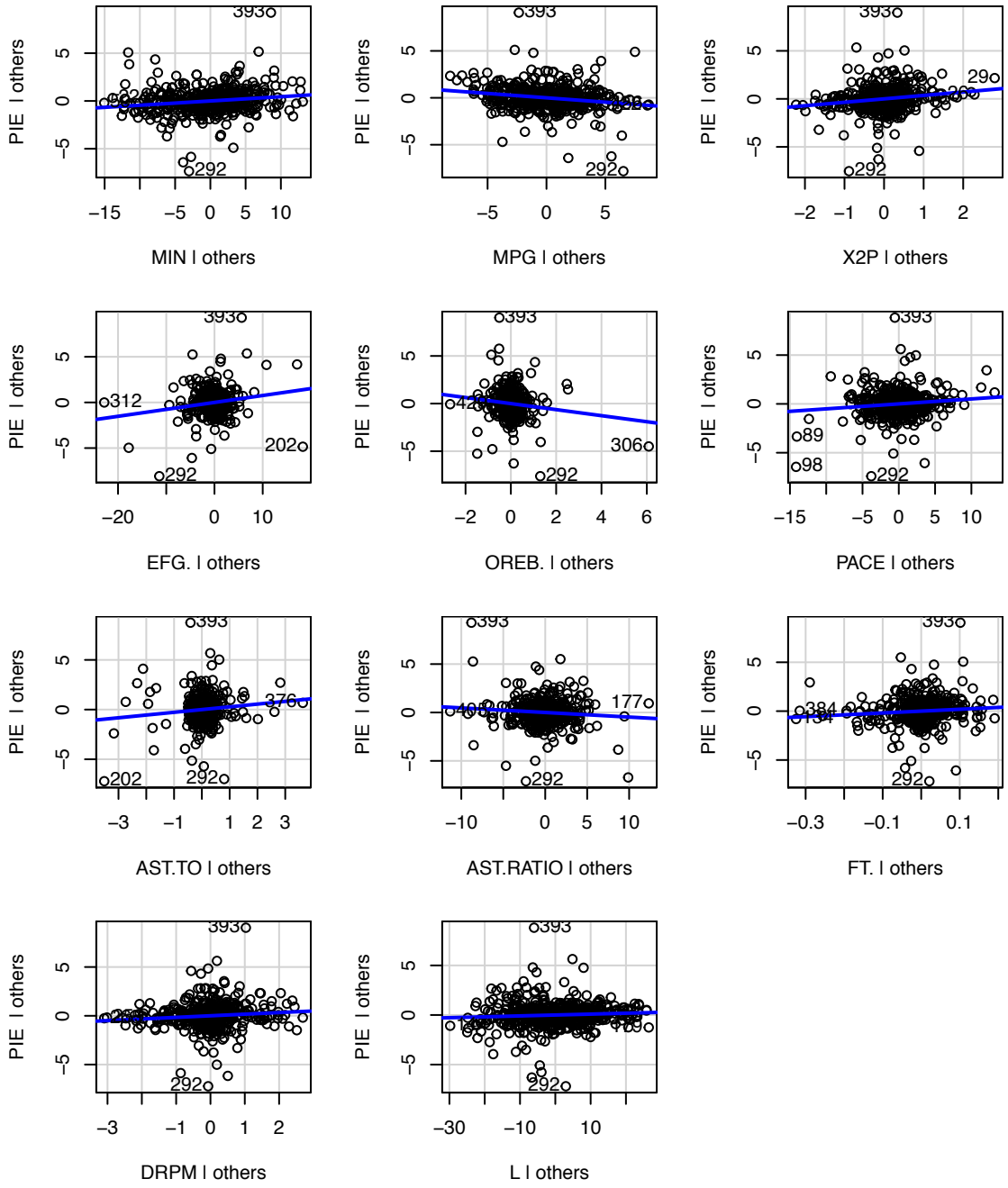
Twitter favorites have a slightly positive correlation with PIE, but retweets do not seem to have any significant correlation with PIE. Wikipedia pageviews seem to have a noticeably negative correlation with PIE. This might occur, because newer players may have more immediate pageviews in the current year, but if pageviews were counted throughout multiple years, it would likely show positive correlation, because better players (likely those with higher PIE) would be more viewed over time.

The most significant signs of correlation appears to occur negatively in offensive and defensive rebound percentage (OREB. & DREB.), and turnover ratio (TO RATIO), and appear to occur positively in Effective Field Goal Percentage (EFG.), Usage Percentage (USG.), true shooting percentage (TS.), and overall rebound percentage (REB.). The most surpringly correlations are the rebound percentages. While having a separately high offensive and defensive rebound percentage lowers yours PIE based on the avPlot, overall rebound percentage raises PIE. This might occur, because some players may be strong in only one of the two categories, but weaker in the other, lowering their PIE value. It seems that a player with well rounded skills (such as turnover ratio, effective field goal percentage, and true shooting percentage) that benefit both offense and defense make a player's PIE increase.

Salary Logistic Regression

```
formula2 <- as.formula(three_str[2])
glm2 <- glm(formula2, data=nba_pt)

par(mfrow=c(1,3), mai=c(0.76,0.76,0,0))
avPlots(glm2, layout=NA)
```





For the salary-based logistic regression formula, SALARY_MILLIONS (player salary) seems to have no correlation with Performance Impact Estimate (PIE). Instead, the variables with the greatest correlation to PIE are:

Usage Percentage (USG.), true shooting percentage (TS.), and Assist Percetntage (AST.) for positive correlation.

Offensive Rebound Percentage (OREB.), and Turnover Ratio (TO RATIO) for negative correlation.

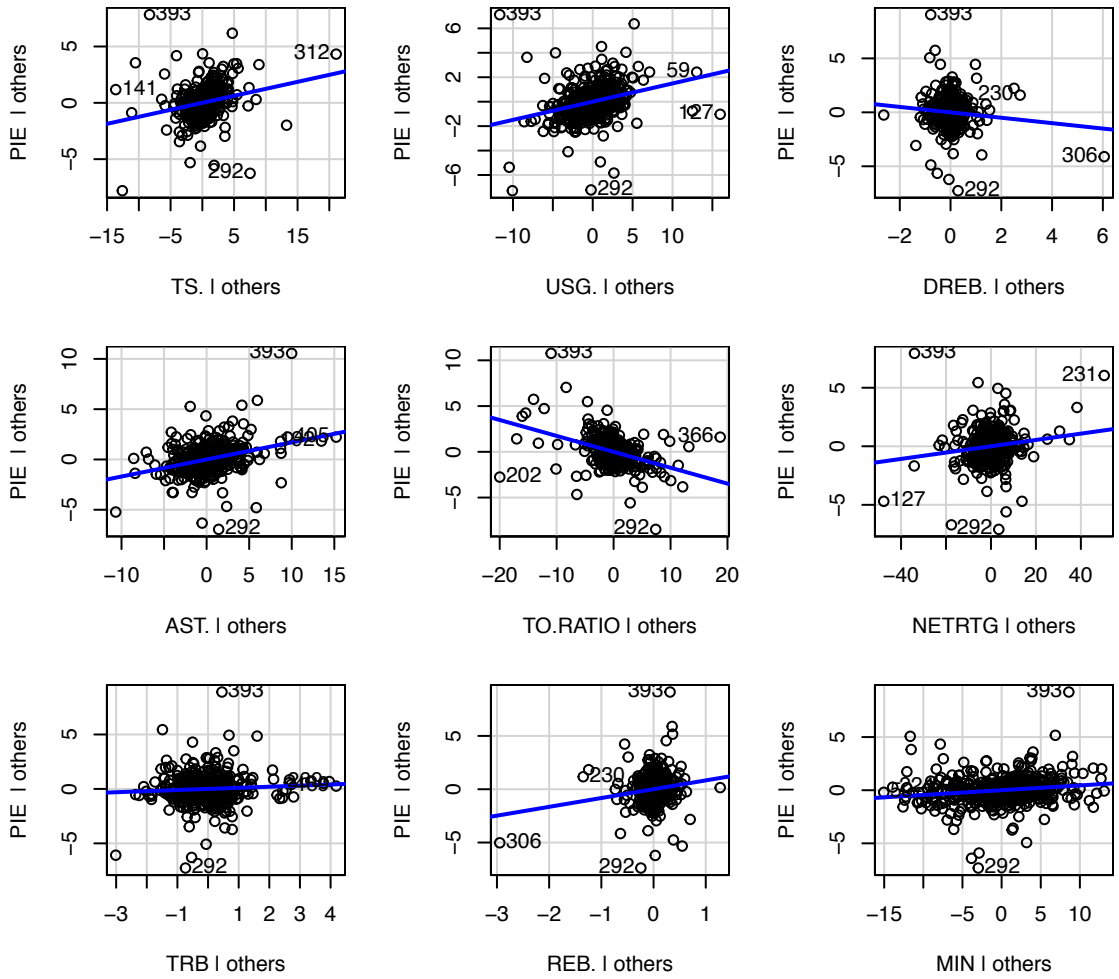
Since PIE measures overall impact and skill, it makes sense that players with more usage would have higher PIE. If a player is doing well, their coach will put them in more to help the team win. True shooting and assist percentages are also good indicators for PIE, because players that can both score themselves and lead to their teammates scoring will be more dynamic in terms of strategy and success in

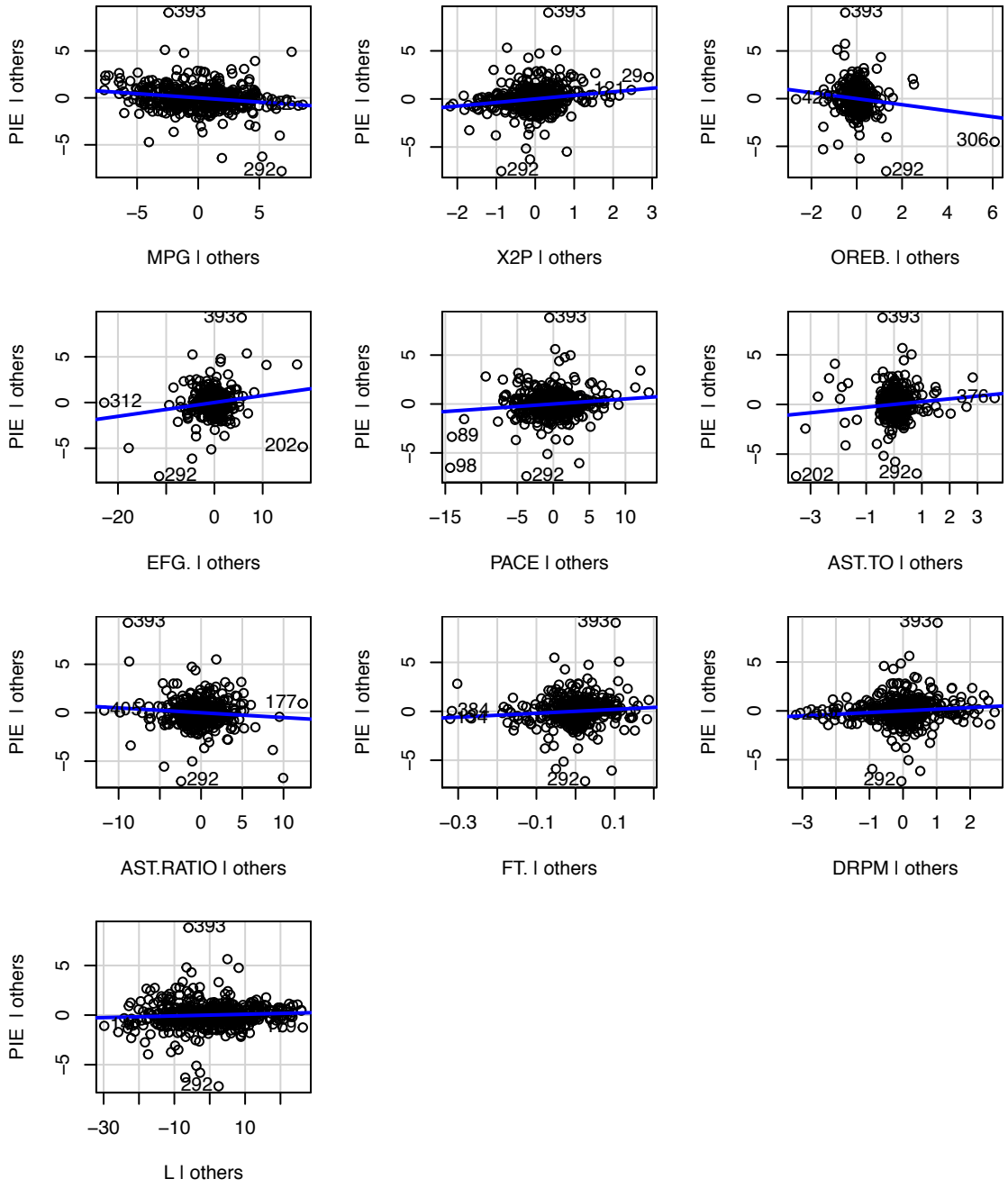
terms of earning points. This, in turn, will make a player's overall impact on the court greater, because they can perform both individually and as a team player.

Turnover Ratio measures how many times you turn the ball over to the other team, an adverse action to do for your team. Players with higher performance will tend to turnover the ball less, resulting in a lower TO RATIO. However, offensive rebound percentage is the odd one out in terms of negative correlation. Normally, having a high offensive rebound percentage leads to a scoring more points. There are no apparent adverse effects with offensive rebounds, but based on this model, it appears that players with high PIE have lower offensive rebounds. This might occur, because high performing players may tend to shoot more instead of rebound for their team.

Logistic Regression with No Additional Initial Variables

```
formula3 <- as.formula(three_str[3])
glm3 <- glm(formula3, data=nba_pt)

par(mfrow=c(1,3), mai=c(0.76,0.76,0,0))
avPlots(glm3, layout=NA)
```





Since we have no initial variables used for the forward step function, we do not need to address the correlation of said variables. However, there appears to be fewers variables in this regression with significant correlation. For example, Usage Percentage (USG.), true shooting percentage (TS.) have a much higher correlation with PIE, and are the only two variables with a strong upward pattern compared to the other variables. Both variables being positive correlations make sense (based on the explanation for the last model), but when no variables are initially added, they appear to have a much higher impact on PIE than the rest of their variable counterparts. For negative correlation, turnover ratio (TO RATIO) is the only variable with a significant negative correlation with PIE. Since turning over the ball to the other team makes TO RATIO a bigger value, it makes sense that players with the highest PIE would have a very low turnover ratio.

Decision Tree, Random Forest Method

Decision Tree and Random Forest Classification

Decision Tree

Including Libraries

```
## Warning: package 'ggplot2' was built under R version 3.6.3

## -- Attaching packages ---------------------------------------------------------------------- tidyve

## v tibble  2.1.3     v dplyr   0.8.3
## v tidyr   1.0.0     v stringr 1.4.0
## v readr   1.3.1     v forcats 0.4.0
## v purrr   0.3.3

## -- Conflicts ------------------------------------------------------------------------ tidyverse_co
## x dplyr::filter() masks stats::filter()
## x dplyr::lag()    masks stats::lag()

## Warning: package 'tree' was built under R version 3.6.3

## Registered S3 method overwritten by 'tree':
##   method     from
##   print.tree cli

## Warning: package 'maptree' was built under R version 3.6.3

## Loading required package: cluster

## Loading required package: rpart

## Warning: package 'rpart' was built under R version 3.6.3

##
## ************************************************************

## Note: As of version 1.0.0, cowplot does not change the

##   default ggplot2 theme anymore. To recover the previous

##   behavior, execute:
##   theme_set(theme_cowplot())

## ************************************************************

## Warning: package 'randomForest' was built under R version 3.6.3

## randomForest 4.6-14
```

```
## Type rfNews() to see new features/changes/bug fixes.

##
## Attaching package: 'randomForest'

## The following object is masked from 'package:dplyr':
##
##     combine

## The following object is masked from 'package:ggplot2':
##
##     margin

## Warning: package 'gridExtra' was built under R version 3.6.3

##
## Attaching package: 'gridExtra'

## The following object is masked from 'package:randomForest':
##
##     combine

## The following object is masked from 'package:dplyr':
##
##     combine
```

Importing Data

```
nba_stats<-read.csv("nba_2017_players_stats_combined.csv")
nba_twitter <- read.csv('nba_2017_players_with_salary_wiki_twitter.csv')
#glimpse(nba_twitter)
```

Only 239 NBA players have twitter accounts, leading to a smaller dataset.

Cleaning the data, and creating a classification variable PIE_HIGH

```
summary(nba_stats$PIE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -6.100   6.825   8.500   8.845  10.700  23.000
```

```
summary(nba_twitter$PIE)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##  -0.400   7.250   8.900   9.559  11.700  23.000
```

```
#the median PIE for nba players with twitter accounts is 0.4 higher than without

PIE_HIGH<-ifelse(nba_twitter$PIE<=8.9, "No", "Yes")
nba_twitter<-data.frame(nba_twitter,PIE_HIGH)
nba_twitter<-nba_twitter%>%select(-X,-Rk,-MP)
nba_socialmedia<-nba_twitter%>%select(PIE_HIGH,SALARY_MILLIONS,PAGEVIEWS,TWITTER_FAVORITE_COUNT,TWITT
```

Splitting the data into test and training sets.

```
set.seed(2)
dim(nba_socialmedia)
```

```
## [1] 239   5
```

```
nba_train<- sample(1:nrow(nba_socialmedia), 0.75*dim(nba_socialmedia)[1])
nba_test<-nba_socialmedia[-nba_train,]
nba_train<-nba_socialmedia[nba_train,]
#nba_test
nba_test_high<-nba_test$PIE_HIGH
```

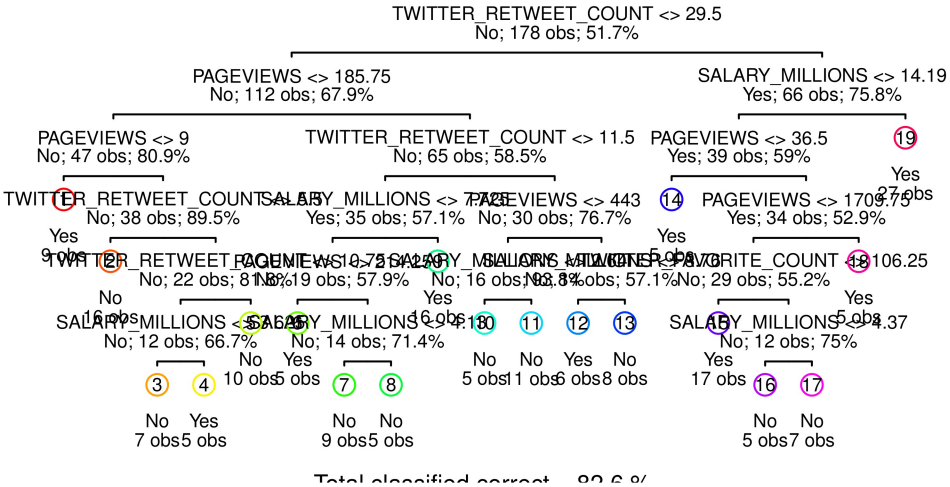
Making the decision tree

```
nba_twitter_tree<- tree(PIE_HIGH~.,data=nba_train)
summary(nba_twitter_tree)
```

```
##
## Classification tree:
## tree(formula = PIE_HIGH ~ ., data = nba_train)
## Number of terminal nodes:  19
## Residual mean deviance:  0.7197 = 114.4 / 159
## Misclassification error rate: 0.1742 = 31 / 178
```

```
draw.tree(nba_twitter_tree, nodeinfo=TRUE, cex=0.6)
title("Classification Tree of NBA Player Performance on Training Set")
```

Classification Tree of NBA Player Performance on Training Set



Here in the training set, twitter retweet count is a better indicator of performance than salary.

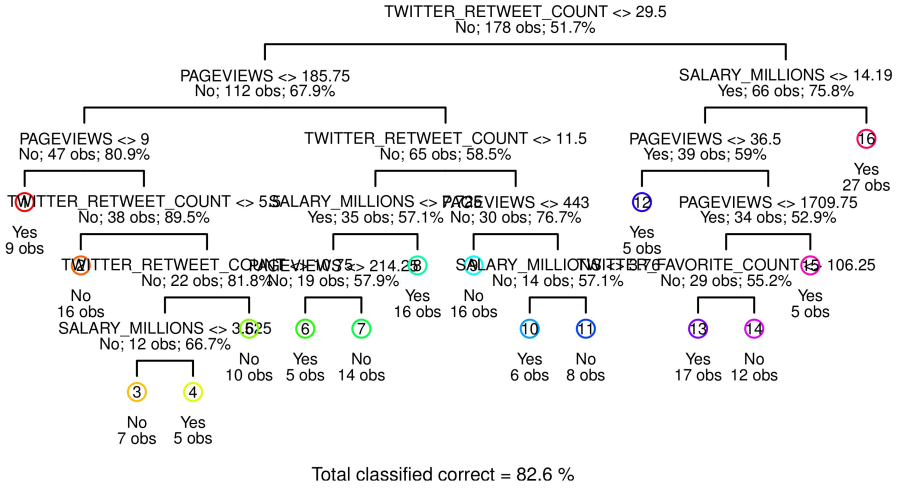
Finding the best size

```
prune<- prune.tree(nba_twitter_tree, k = 0:20, method = "misclass")
best.prune<- prune$size[which.min(prune$dev)]
best.prune
```

```
## [1] 16
```

```
pt.prune <- prune.misclass(nba_twitter_tree, best=best.prune)
draw.tree(pt.prune,nodeinfo=TRUE,cex=0.5)
title("Pruned Tree")
```

Pruned Tree



Performing a cross validation and finding the best size for it

```
cv_tree<-cv.tree(nba_twitter_tree,FUN=prune.misclass,K=5)
cv_tree
```

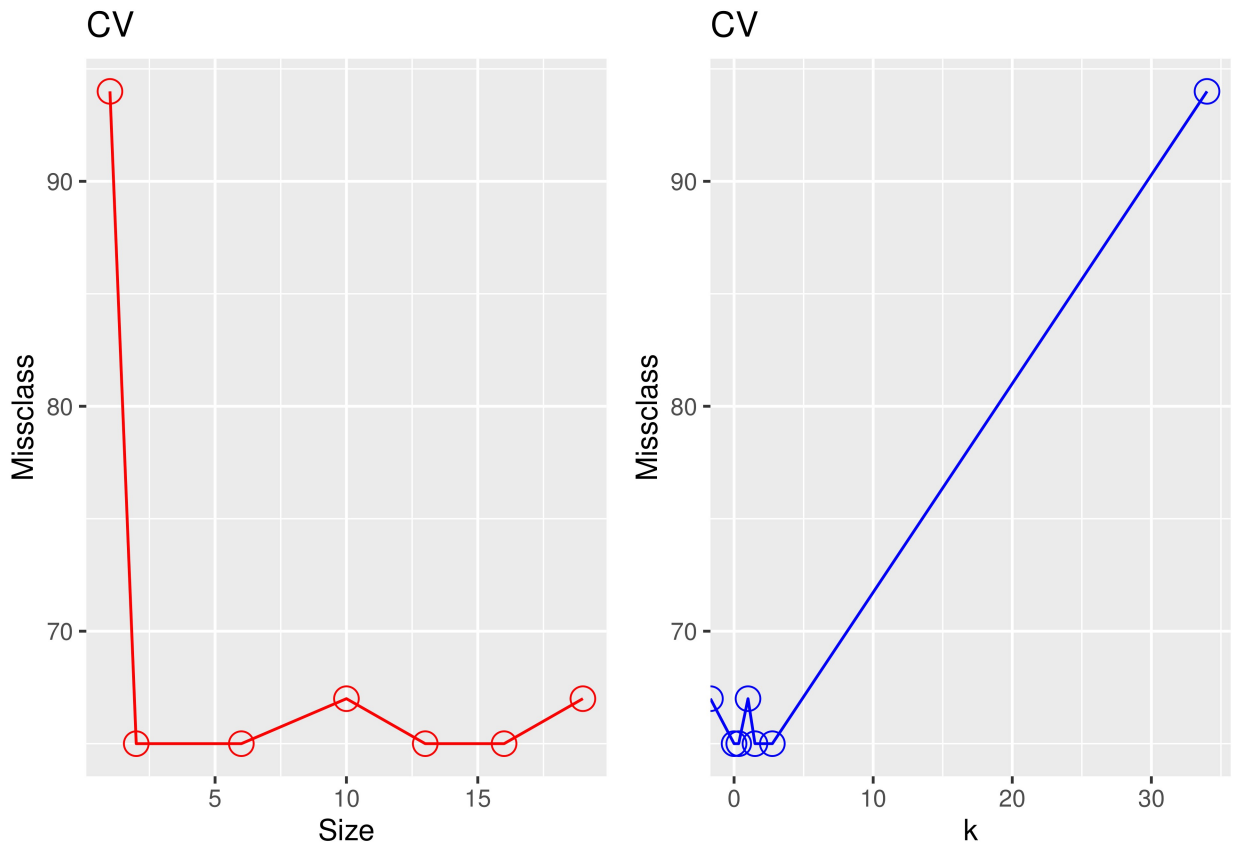
```
## $size
## [1] 19 16 13 10  6  2  1
##
## $dev
## [1] 67 65 65 67 65 65 94
##
## $k
## [1]       -Inf  0.0000000  0.3333333  1.0000000  1.5000000  2.7500000 34.0000000
##
## $method
## [1] "misclass"
##
## attr(,"class")
## [1] "prune"         "tree.sequence"
```

```
best.cv = cv_tree$size[which.min(cv_tree$dev)]
best.cv
```

```
## [1] 16
```

Graphing the CV misclassification error

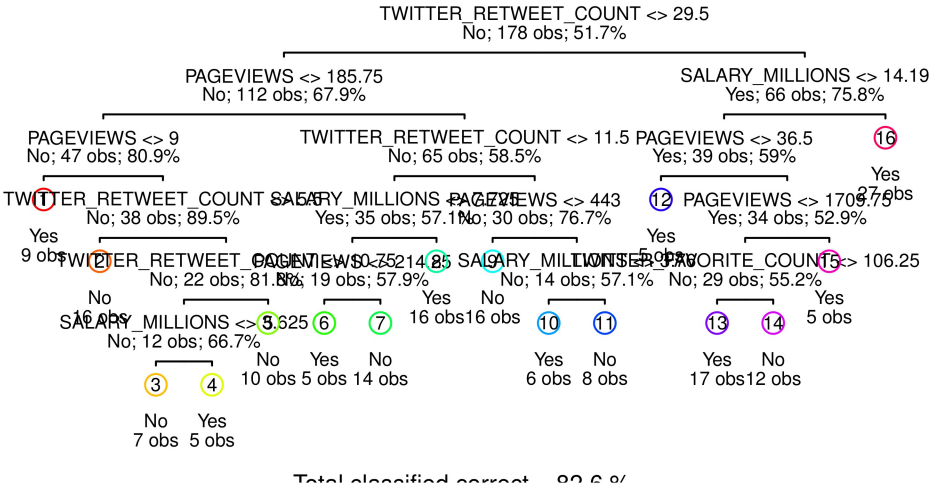
```
#Making a dataframe for ggplot
cv_graph<-data.frame(Size = cv_tree$size, Missclass= cv_tree$dev, k=cv_tree$k)
p<-ggplot(cv_graph,mapping=aes(x=Size, y=Missclass)) + geom_point(color="red",shape=1, size=4)+geom_l
q<-ggplot(cv_graph,mapping=aes(x=k, y=Missclass)) + geom_point(color="blue",shape=1, size=4)+geom_lin
grid.arrange(p,q,ncol=2)
```



Plotting the CV tree

```
pt.cv <-prune.misclass(nba_twitter_tree, best=best.cv)
draw.tree(pt.cv,nodeinfo=TRUE,cex=0.6)
title("CV tree")
```

CV tree



Calculating the Test Error error Rates for the Pruned and CV models

```
#Predicting on test set
pred.pt.prune<-predict(pt.prune, nba_test, type="class")
# Obtain confusion matrix
err.pt.prune<-table(pred.pt.prune, nba_test_high)
err.pt.prune
```

```
##              nba_test_high
## pred.pt.prune No Yes
##           No  18  13
##           Yes 10  19
```

```
print("Classification Error for Pruned Model:")
```

```
## [1] "Classification Error for Pruned Model:"
```

```
1-sum(diag(err.pt.prune))/sum(err.pt.prune)
```

```
## [1] 0.3833333
```

```
# Predict on test set
pred.pt.cv = predict(pt.cv, nba_test, type="class")
# Obtain confusion matrix
err.pt.cv = table(pred.pt.cv, nba_test_high)
err.pt.cv
```

```
##           nba_test_high
## pred.pt.cv No Yes
##        No  18  13
##        Yes 10  19
```

```
print("Classification Error for CV Model:")
```

```
## [1] "Classification Error for CV Model:"
```

```
1-sum(diag(err.pt.cv))/sum(err.pt.cv)
```

```
## [1] 0.3833333
```

Since the number of leaves were the same, we have the same error rate for both.

Random Forest

Making the Forest

```
set.seed(69)
nba_forest<-randomForest(PIE_HIGH~.,data=nba_socialmedia,proximity=TRUE,na.action=na.roughfix)
nba_forest
```

```
##
## Call:
##  randomForest(formula = PIE_HIGH ~ ., data = nba_socialmedia,      proximity = TRUE, na.action = n
##                Type of random forest: classification
##                      Number of trees: 500
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 39.33%
## Confusion matrix:
##     No Yes class.error
## No  76  45   0.3719008
## Yes 49  69   0.4152542
```

Graphing the Classification Error Rates

```
oob.error.data <- data.frame(
  Trees=rep(1:nrow(nba_forest$err.rate), times=3),
  Type=rep(c("OOB", "No", "Yes"), each=nrow(nba_forest$err.rate)),
  Error=c(nba_forest$err.rate[,"OOB"],
          nba_forest$err.rate[,"No"],
          nba_forest$err.rate[,"Yes"]))

ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))
```

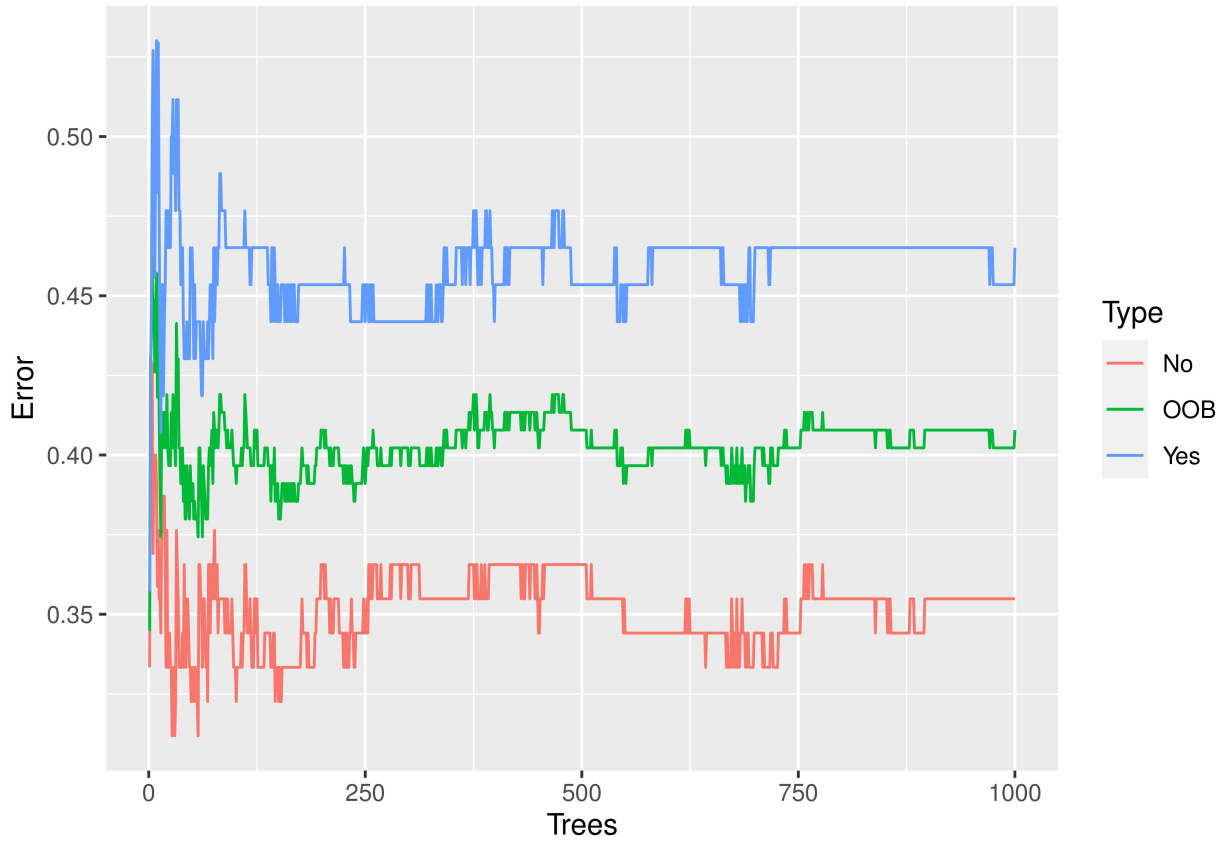
Trying twice as many trees

```
set.seed(69)
nba_forest<-randomForest(PIE_HIGH~.,data=nba_train,proximity=TRUE,na.action=na.roughfix,ntree=1000)
nba_forest
```

```
##
## Call:
##  randomForest(formula = PIE_HIGH ~ ., data = nba_train, proximity = TRUE,      ntree = 1000, na.ac
##                Type of random forest: classification
##                      Number of trees: 1000
## No. of variables tried at each split: 2
##
##         OOB estimate of  error rate: 40.78%
## Confusion matrix:
##     No Yes class.error
## No  60  33   0.3548387
## Yes 40  46   0.4651163
```

```
oob.error.data <- data.frame(
  Trees=rep(1:nrow(nba_forest$err.rate), times=3),
  Type=rep(c("OOB", "No", "Yes"), each=nrow(nba_forest$err.rate)),
  Error=c(nba_forest$err.rate[,"OOB"],
          nba_forest$err.rate[,"No"],
          nba_forest$err.rate[,"Yes"]))
```

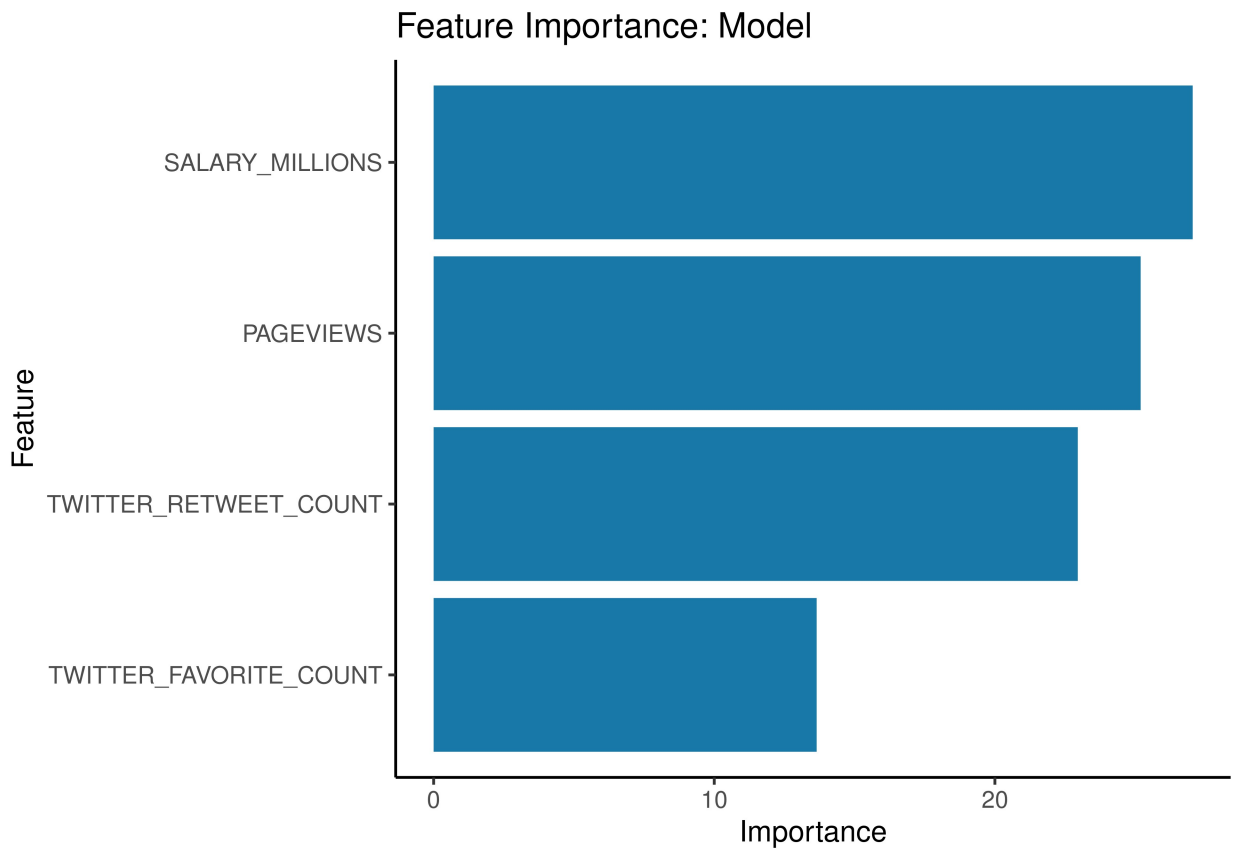
```
ggplot(data=oob.error.data, aes(x=Trees, y=Error)) +
  geom_line(aes(color=Type))
```



It levels out around 750 so we have a good sized forest there.

Measuring Variable Significance

```
feat_imp_df <- importance(nba_forest) %>%
  data.frame() %>%
  mutate(feature = row.names(.))

# plot dataframe
ggplot(feat_imp_df, aes(x = reorder(feature, MeanDecreaseGini),
                        y = MeanDecreaseGini)) +
  geom_bar(stat='identity',fill="#1979a9") +
  coord_flip() + theme_classic() +
  labs(
    x     = "Feature",
    y     = "Importance",
    title = "Feature Importance: Model"
  )
```



Using Gini importance mechanism, we see that salary is the best indicator for an NBA players performance.

KNN Method with Salary

Data Preprocessing

We are going to make a new data frame called salary, which only contains numeric variables. We are also going to make a new response variable that represents the player impact estimation (PIE). The binary response variable "high" measures whether or not the PIE is high or not, and is determined by the following:

$$\text{High} = \begin{cases} Yes\ (salary > median(salary)) \\ No\ (salary \leq median(salary)) \end{cases}$$

```
# check which columns contain NA or missing values
colnames(nba_pie_twitter)[colSums(is.na(nba_pie_twitter)) > 0]
```

```
## [1] "X3P."                    "FT."                     "TWITTER_FAVORITE_COUNT"
## [4] "TWITTER_RETWEET_COUNT"
```

```
# create a new dataframe with the binary response variable 'high'
#remove the variables that are not relevant or contain missing values
salary = nba_pie_twitter %>%
  mutate(high=as.factor(ifelse(SALARY_MILLIONS<=median(SALARY_MILLIONS),"Low","High")))%>%
  select(-PLAYER,-TEAM.x,-POSITION,-PAGEVIEWS,-TWITTER_FAVORITE_COUNT,
         -TWITTER_RETWEET_COUNT,-X,-eFG.,-ORPM,-DRPM,
         -OFFRTG,-DEFRTG,-X3P.,-FT.)

head(salary$high)
```

```
## [1] Low  Low  Low  High High High
## Levels: High Low
```

Now we have added our binary response variable *high* to the data frame. As said above, the *high* is a measure of player impact estimation, and denoted as "low" if the salary of a player is lower than the median salary across all players, and vice versa.

```
salary_millions <- salary$SALARY_MILLIONS
salary_high <- salary$high
high_count <- length(salary_high[salary_high=='High'])
low_count <- length(salary_high[salary_high=='Low'])

print(paste0('The median salary in millions = ',median(nba_pie_twitter$SALARY_MILLIONS)))
```

```
## [1] "The median salary in millions = 5.37"
```

```
par(mfrow=c(1,2))
plot(salary_high,width=0.5,col=c('blue','red'))
#legend("bottomright",pch=15,fill,col=c('blue','red'),
       #legend=c('High=118','Low=120'))
plot(nba_pie_twitter$SALARY_MILLIONS,nba_pie_twitter$PIE.x,
     col=ifelse(salary_high=='High','blue','red'))
mtext("High Salary versus Low Salary",side=3,line=-1,outer=TRUE)
```

As expected, the split between the number of NBA players who have a salary considered high and the players who have a salary that is considered low is about even. This is likely because we determined high and low with the median salary, so it makes sense that half will be high and half will be low. Looking at the scatter plot of the salary data, we can see that the two classes (high,low) are actually split fairly linearly. However, the salaries that are considered "high" are a lot more spread out than the salaries considered as "low"; the low salaries fall in between 0-5 million dollars, while the high ones range from 6-31 million. Note: the player who is paid the salary of approximately 31 million is LEBRON JAMES.

We want to investigate the relationship between player impact estimation and high/low salary.

Training/Test Split (salary)

We are going to split the data into roughly 50% training data, and 50% testing data. Because the dataset contains information on 239 players, the first 119 will be used for training data and the remaining 118 will be for testing data, and we will omit one row from the training data. This is because later on when we plot the KNN clusters, the input for our data frames need to have the same number of rows.

```
set.seed(69)

salary= salary[-239,]

s.train=sample(1:nrow(salary),119)
salary.train = salary[s.train,]
salary.test = salary[-s.train,]

s.Ytrain = salary.train$high
```

```
s.Xtrain = salary.train %>% select(-high)

s.Ytest = salary.test$high
s.Xtest = salary.test %>% select(-high)

#head(str(s.Ytrain))
#head(str(s.Xtrain))
```

Before we find the training and testing error when $k = 5$, we will make a table to keep track of it.

```
k.5table <- matrix(nc=2,nr=4)
k.5table <- as.table(k.5table)
```

Finding the training error rate with k=5 chunks (salary)

Because we do not know the best value for k, we are going to arbitrarily assume that $k = 5$ will produce the lowest error rates. Later on, we will use Leave One Out Cross Validation (LOOCV) to determine the actual best number of neighbors.

```
library(class)
set.seed(69)

salary.predicted.Ytrain=knn(train=s.Xtrain,test=s.Xtrain,cl=s.Ytrain,k=5)
salary.confusion.matrix.train = table(Predicted=salary.predicted.Ytrain,observed=s.Ytrain)
salary.confusion.matrix.train
```

```
##           observed
## Predicted High Low
##      High   43  11
##      Low    17  48
```

```
salary.accuracy.train = sum(diag(salary.confusion.matrix.train)/sum(salary.confusion.matrix.train))
salary.errorrate.train = 1-salary.accuracy.train

print(paste0('Training accuracy rate = ', round(salary.accuracy.train,3)))
```

```
## [1] "Training accuracy rate = 0.765"
```

```
print(paste0('Training error rate = ', round(salary.errorrate.train,3)))
```

```
## [1] "Training error rate = 0.235"
```

```
k.5table[1,1] = "Training Accuracy Rate"
k.5table[1,2] = round(salary.accuracy.train,3)
k.5table[3,1] = "Training Error Rate"
k.5table[3,2] = round(salary.errorrate.train,3)
```

Finding the testing error rate with k=5 chunks (salary)

```r
set.seed(69)

salary.predicted.Ytest=knn(train=s.Xtrain,test=s.Xtest,cl=s.Ytrain,k=5)
salary.confusion.matrix.test = table(predicted=salary.predicted.Ytest,observed=s.Ytest)
salary.confusion.matrix.test
```

```
##          observed
## predicted High Low
##      High   40  18
##      Low    18  43
```

```r
salary.accuracy.test = sum(diag(salary.confusion.matrix.test)/sum(salary.confusion.matrix.test))
salary.errorrate.test = 1-salary.accuracy.test

print(paste0('Testing accuracy rate = ', round(salary.accuracy.test,3)))
```

```
## [1] "Testing accuracy rate = 0.697"
```

```r
print(paste0('Testing error rate = ', round(salary.errorrate.test,3)))
```

```
## [1] "Testing error rate = 0.303"
```

```r
k.5table[2,1] = "Testing Accuracy Rate"
k.5table[2,2] = round(salary.accuracy.test,3)
k.5table[4,1] = "Testing Error Rate"
k.5table[4,2] = round(salary.errorrate.test,3)
```

```r
k.5table
```

```
##   A                       B
## A Training Accuracy Rate 0.765
## B Testing Accuracy Rate  0.697
## C Training Error Rate    0.235
## D Testing Error Rate     0.303
```

Our error rate and accuracy rates are not bad, however we will use LOOCV to determine the optimal number of neighbors.

Leave One Out Cross Validation to Determine Best K (salary)

```r
set.seed(69)
library(class)
salary.valerror <- NULL
salary.allk <- 1:100

for(x in salary.allk){
  salary.predYval <- knn.cv(train=s.Xtrain,cl=s.Ytrain,k=x)
  salary.valerror = c(salary.valerror,mean(salary.predYval!=s.Ytrain))
```

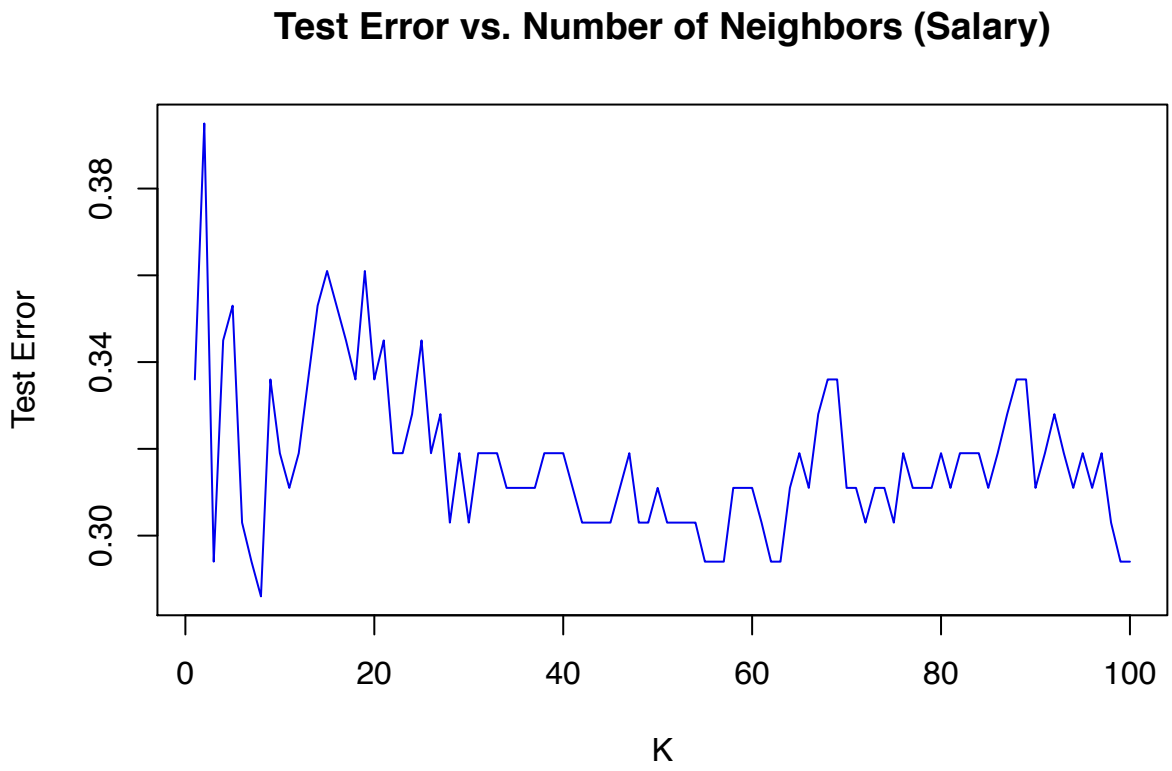
```
}

salary.valerror = round(salary.valerror,digits=3)
salary.valerror
```

```
##   [1] 0.336 0.395 0.294 0.345 0.353 0.303 0.294 0.286 0.336 0.319 0.311 0.319
##  [13] 0.336 0.353 0.361 0.353 0.345 0.336 0.361 0.336 0.345 0.319 0.319 0.328
##  [25] 0.345 0.319 0.328 0.303 0.319 0.303 0.319 0.319 0.319 0.311 0.311 0.311
##  [37] 0.311 0.319 0.319 0.319 0.311 0.303 0.303 0.303 0.303 0.311 0.319 0.303
##  [49] 0.303 0.311 0.303 0.303 0.303 0.303 0.294 0.294 0.294 0.311 0.311 0.311
##  [61] 0.303 0.294 0.294 0.311 0.319 0.311 0.328 0.336 0.336 0.311 0.311 0.303
##  [73] 0.311 0.311 0.303 0.319 0.311 0.311 0.311 0.319 0.311 0.319 0.319 0.319
##  [85] 0.311 0.319 0.328 0.336 0.336 0.311 0.319 0.328 0.319 0.311 0.319 0.311
##  [97] 0.319 0.303 0.294 0.294
```

```
salary.numberofneighbors <- max(salary.allk[salary.valerror==min(salary.valerror)])

print(paste0('Best number of neighbors using LOOCV = ',salary.numberofneighbors))
```

```
## [1] "Best number of neighbors using LOOCV = 8"
```

```
plot(x=salary.allk,y=salary.valerror,type='l',ylab="Test Error",xlab="K",col='blue2',
     main="Test Error vs. Number of Neighbors (Salary)")
```



According to the LOOCV algorithm, the best number of neighbors is $k = 8$. That means that we should

be using 8 to determine the class of our binary variable, *high*. In this case, the class of *high* will either be "yes" or "no" depending on player salary.

Train a KNN Classifier Using the Best Number of Neighbors Calculated with LOOCV (salary)

```
set.seed(69)
salary.predYtest <- knn(train=s.Xtrain,test=s.Xtest,cl=s.Ytrain,k=salary.numberofneighbors)
salary.confusionmatrix.bestk = table(predicted=salary.predYtest,true=s.Ytest)
salary.confusionmatrix.bestk
```

```
##          true
## predicted High Low
##      High   40  24
##      Low    18  37
```

Calculate the Testing Error Rate with LOOCV (salary)

```
salary.accuracy.bestk = sum(diag(salary.confusionmatrix.bestk)/sum(salary.confusionmatrix.bestk))
salary.errorrate.bestk = 1-salary.accuracy.bestk

print(paste0('Testing accuracy rate = ', round(salary.accuracy.bestk,3)))
```

```
## [1] "Testing accuracy rate = 0.647"
```

```
print(paste0('Testing error rate = ', round(salary.errorrate.bestk,3)))
```

```
## [1] "Testing error rate = 0.353"
```

As expected, after performing LOOCV and using that number for the number of neighbors, our accuracy rate is higher and our error rates are lower.

```
library(plyr)
library(dplyr)

plotdf1 <- data.frame(s.Xtrain,s.Xtest,predicted=salary.predYtest)
plotdf2 <- data.frame(x=plotdf1$SALARY_MILLIONS,y=plotdf1$PIE.x,predicted=plotdf1$predicted)
findhull <- function(df) df[chull(df$x,df$y),]

boundary2 <- ddply(plotdf2,.variables='predicted', .fun=findhull)

ggplot(plotdf1,aes(SALARY_MILLIONS,PIE.x,color=predicted,fill=predicted)) +
  geom_point(size=3)+
  geom_polygon(data=boundary2,aes(x,y),alpha=0)+
  ggtitle("KNN Cluster Boundaries After LOOCV")
```

Unfortunetly, it appears that the plot of the cluster boundaries does not offer us much useful information. We can see how the high and low boundaries are marked on the plot, but since both the clusters are overlapping, it's difficult to make any conclusions about using salary to classify PIE. Overall, this plot may be an indicator that salary is not a great way to classify whether PIE is high or low.

KNN Method with Social Media

Data Preprocessing

We are going to make a new data frame called social media, which is comprised of the number of Twitter likes and Twitter retweets. Once again, we are going to make a new response variable that represents the player impact estimation (PIE). The binary response variable "high" measures whether or not the PIE is high or not. This time, it is based on social_media, instead of salary.

Social Media = TWITTER_FAVORITE_COUNT + TWITTER_RETWEET_COUNT

$$\text{High} = \begin{cases} Yes\ (social\ media > med(social\ media)) \\ No\ (social\ media \leq med(social\ media)) \end{cases}$$

```
# create new column in original dataframe for social media
nba_pie_twitter$social <- as.numeric(nba_pie_twitter$TWITTER_FAVORITE_COUNT +
                                       nba_pie_twitter$TWITTER_RETWEET_COUNT + nba_pie_twitter$PAGEVI

# replace missing social values with the av

# create new data frame with the binary response variable 'high'
```

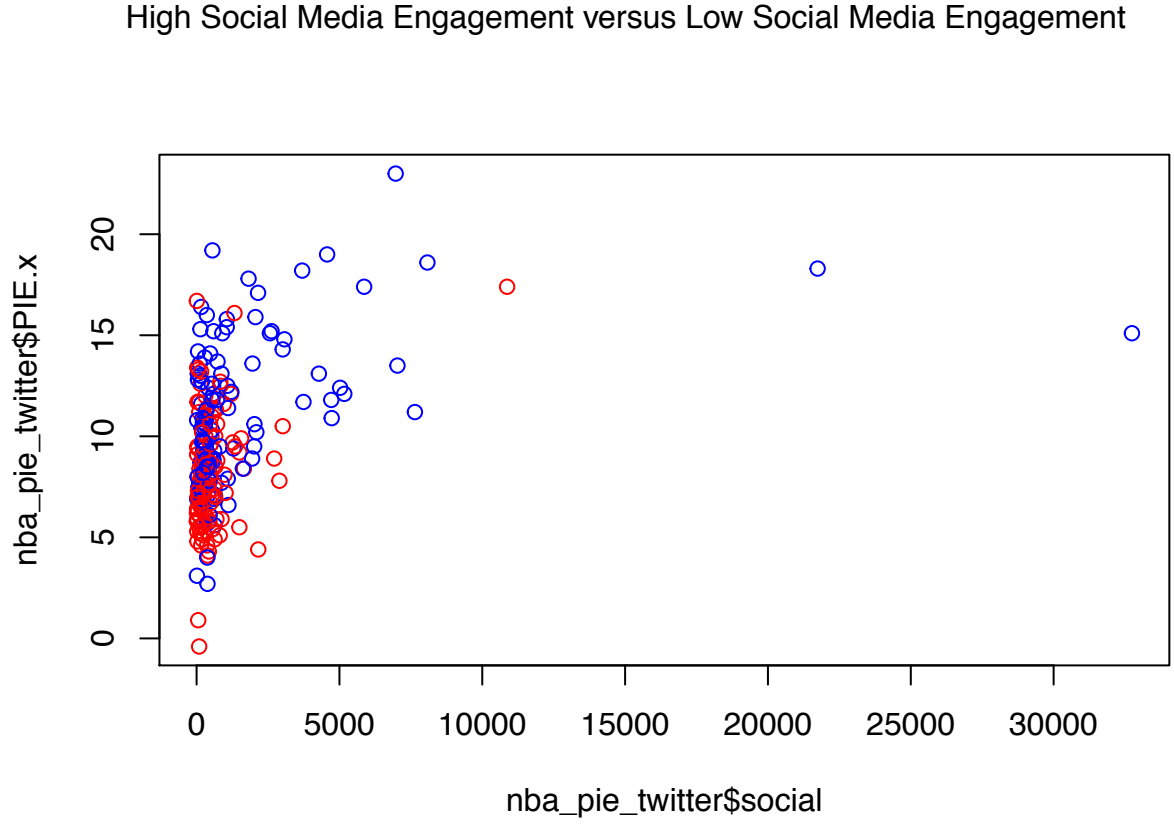
```
social = nba_pie_twitter %>%
  mutate(high=as.factor(ifelse(social<=median(social,na.rm=TRUE),"Low","High")))%>%
  select(-PLAYER,-TEAM.x,-POSITION,-PAGEVIEWS,-TWITTER_FAVORITE_COUNT,
         -TWITTER_RETWEET_COUNT,-X,-eFG.,-ORPM,-DRPM,
         -OFFRTG,-DEFRTG,-social,-X3P.,-FT.)



head(social$high)
```

```
## [1] Low  High Low  High Low  Low
## Levels: High Low
```

```
socialmedia_high <- social$high
social_high_count <- length(socialmedia_high[socialmedia_high=='High'])
social_low_count <- length(socialmedia_high[socialmedia_high=='Low'])

print(paste0('The median social media engagement = ',median(nba_pie_twitter$social,na.rm=TRUE)))
```

```
## [1] "The median social media engagement = 374.25"
```

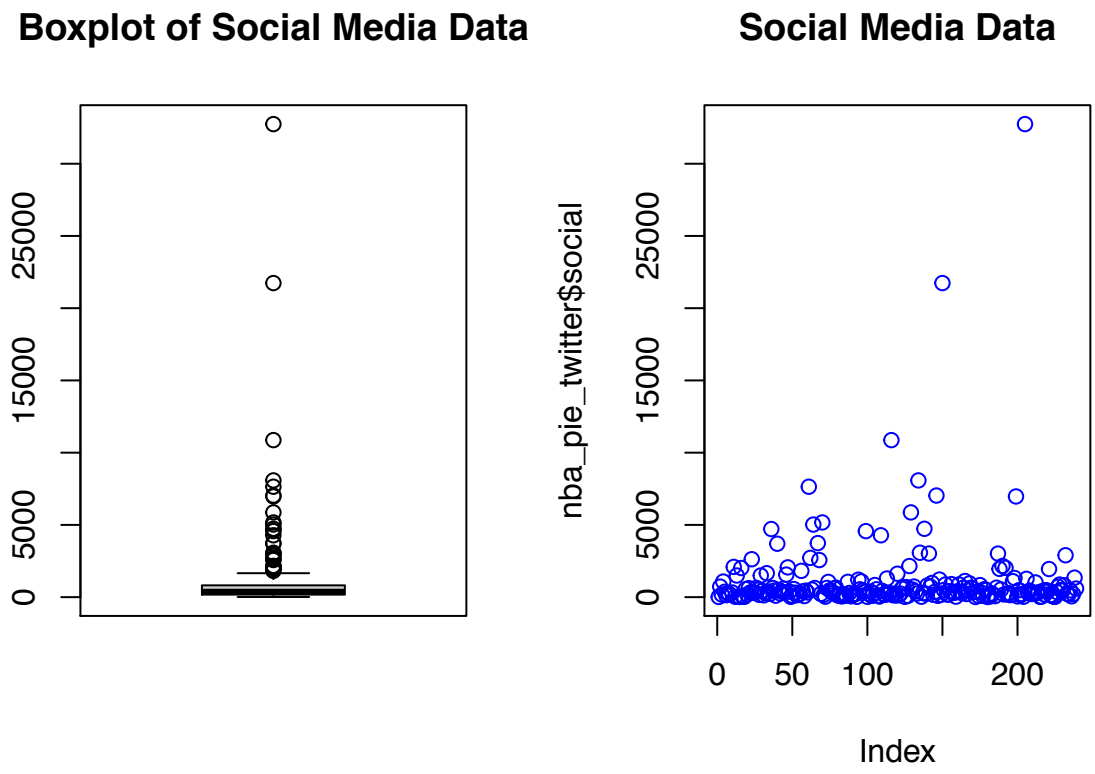
```
plot(nba_pie_twitter$social,nba_pie_twitter$PIE.x,col=ifelse(salary_high=='High','blue','red'))
mtext("High Social Media Engagement versus Low Social Media Engagement",side=3,line=-1,outer=TRUE)
```



An initial graph looks like there may be some outliers in social media usage. We may have to remove these potential outliers so the boundaries are more defined when we perform K Nearest Neighbors.

Visualizing the Social Media Outliers

```
par(mfrow=c(1,2))
outvalue <- boxplot(nba_pie_twitter$social,main='Boxplot of Social Media Data')$out
plot(nba_pie_twitter$social,col='blue',main='Social Media Data')
```



It appears like there are quite a few outliers in the dataset, as indicated by the boxplot and the scatterplot. To further illustrate this, we'll list out the points that are considered outliers. It's important that we handle the outliers carefully, because KNN uses Euclidean Distance to calculate how close points are to one another and ultimately classify our response variable. In this case, we are first going to try to perform KNN while leaving the outliers present, and train our KNN algorithm with them included in the dataset, and see what kind of error rates we get.

```
which(nba_pie_twitter$social %in% outvalue)
```

```
##  [1]  11  16  23  36  40  47  56  61  62  64  67  68  70  99 109 116 128 129 134
## [20] 135 138 141 146 150 187 188 190 192 199 205 221 232
```

Removing Missing Values

Since the KNN function won't work when there are missing values in our dataset, we have to remove them. First, lets see how many missing values there are in the column *high*.

```
sum(is.na(as.numeric(social$high)))
```

```
## [1] 3
```

Since there are only 3 missing values, removing them would be fine as long as we account for the smaller dataset when we make our training/test split.

```
social <- na.omit(social)
sum(as.numeric(is.na(social$high)))
```

```
## [1] 0
```

```
nrow(social)
```

```
## [1] 236
```

Now, we have 0 missing values and our dataset has 236 rows.

Training/Test Split (social media)

Since our social dataframe has 236 rows, we can split it into 50% training data and 50% testing data.

```
set.seed(69)

#social= social[-236,]

sm.train=sample(1:nrow(social),118)
sm.train=as.numeric(sm.train)
social.train = social[sm.train,]
social.test = social[-sm.train,]
```

```
sm.Ytrain = social.train$high
sm.Xtrain = social.train %>% select(-high)

sm.Ytest = social.test$high
sm.Xtest = social.test %>% select(-high)
```

Once again, we will make a table to keep track of the training error when using $k = 5$ nearest neighbors.

```
k.5table_social <- matrix(nc=2,nr=4)
k.5table_social <- as.table(k.5table_social)
```

Finding the training error rate with k=5 chunks (social media)

Because we do not know the best value for k, we are going to arbitrarily assume that $k = 5$ will produce the lowest error rates. Later on, we will use Leave One Out Cross Validation (LOOCV) to determine the actual best number of neighbors.

```r
#library(class)
set.seed(69)

social.predicted.Ytrain=knn(train=sm.Xtrain,test=sm.Xtrain,cl=sm.Ytrain,k=5)
social.confusion.matrix.train = table(Predicted=social.predicted.Ytrain,observed=sm.Ytrain)
social.confusion.matrix.train
```

```
##          observed
## Predicted High Low
##      High   44  13
##      Low    20  41
```

```r
social.accuracy.train = sum(diag(social.confusion.matrix.train)/sum(social.confusion.matrix.train))
social.errorrate.train = 1-social.accuracy.train

print(paste0('Training accuracy rate = ', round(social.accuracy.train,3)))
```

```
## [1] "Training accuracy rate = 0.72"
```

```r
print(paste0('Training error rate = ', round(social.errorrate.train,3)))
```

```
## [1] "Training error rate = 0.28"
```

```r
k.5table_social[1,1] = "Training Accuracy Rate"
k.5table_social[1,2] = round(social.accuracy.train,3)
k.5table_social[3,1] = "Training Error Rate"
k.5table_social[3,2] = round(social.errorrate.train,3)
```

Finding the testing error rate with k=5 chunks (social media)

```r
set.seed(69)

social.predicted.Ytest=knn(train=sm.Xtrain,test=sm.Xtest,cl=sm.Ytrain,k=5)
social.confusion.matrix.test = table(predicted=social.predicted.Ytest,observed=sm.Ytest)
social.confusion.matrix.test
```

```
##          observed
## predicted High Low
##      High   25  26
##      Low    29  38
```

```r
social.accuracy.test = sum(diag(social.confusion.matrix.test)/sum(social.confusion.matrix.test))
social.errorrate.test = 1-social.accuracy.test

print(paste0('Testing accuracy rate = ', round(social.accuracy.test,3)))
```

```
## [1] "Testing accuracy rate = 0.534"
```

```
print(paste0('Testing error rate = ', round(social.errorrate.test,3)))
```

```
## [1] "Testing error rate = 0.466"
```

```
k.5table_social[2,1] = "Testing Accuracy Rate"
k.5table_social[2,2] = round(social.accuracy.test,3)
k.5table_social[4,1] = "Testing Error Rate"
k.5table_social[4,2] = round(social.errorrate.test,3)
```

Now that we've calculated the training and testing error rate when using $k = 5$, we can print out the table.

```
k.5table_social
```

```
##   A                       B
## A Training Accuracy Rate 0.72
## B Testing Accuracy Rate  0.534
## C Training Error Rate    0.28
## D Testing Error Rate     0.466
```

Recall, this was our table when using $k = 5$ but with salary data:

```
k.5table
```

```
##   A                       B
## A Training Accuracy Rate 0.765
## B Testing Accuracy Rate  0.697
## C Training Error Rate    0.235
## D Testing Error Rate     0.303
```

Leave One Out Cross Validation to Determine Best K (social media)

```
set.seed(69)
library(class)
social.valerror <- NULL
social.allk <- 1:100

for(x in social.allk){
  social.predYval <- knn.cv(train=sm.Xtrain,cl=sm.Ytrain,k=x)
  social.valerror = c(social.valerror,mean(social.predYval!=sm.Ytrain))
}

social.valerror = round(social.valerror,digits=3)
social.valerror
```

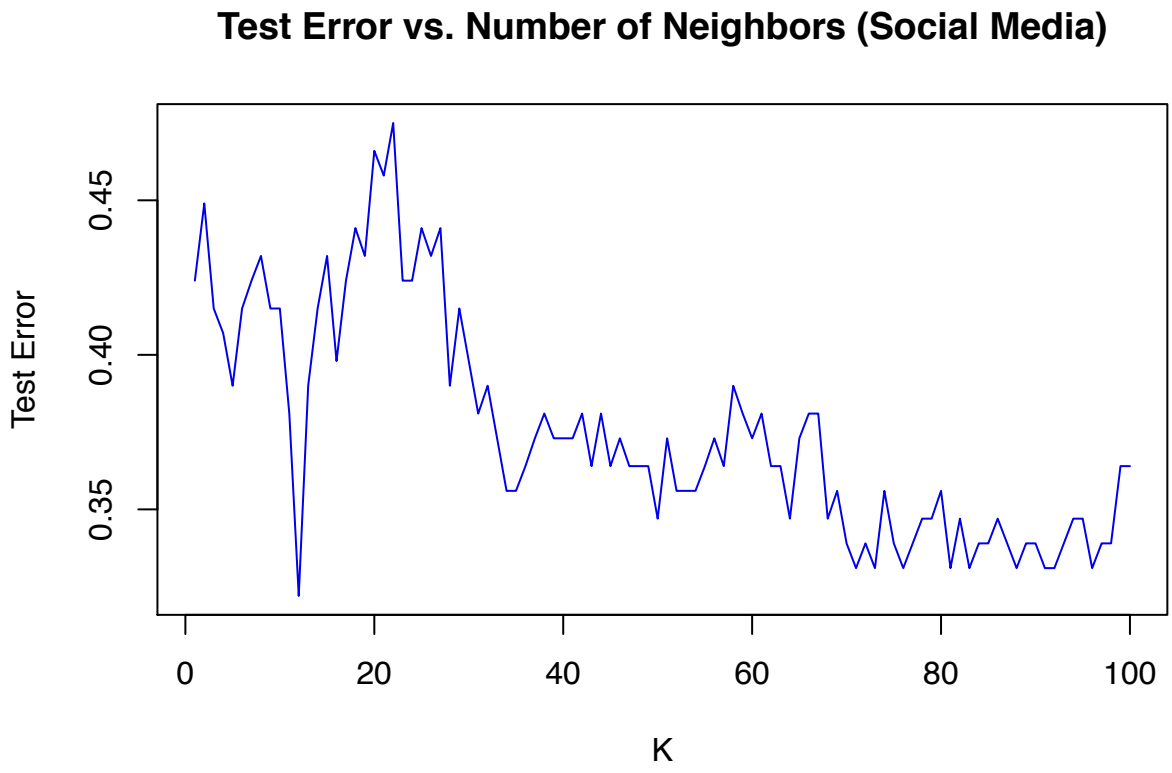
```
##   [1] 0.424 0.449 0.415 0.407 0.390 0.415 0.424 0.432 0.415 0.415 0.381 0.322
##  [13] 0.390 0.415 0.432 0.398 0.424 0.441 0.432 0.466 0.458 0.475 0.424 0.424
##  [25] 0.441 0.432 0.441 0.390 0.415 0.398 0.381 0.390 0.373 0.356 0.356 0.364
##  [37] 0.373 0.381 0.373 0.373 0.373 0.381 0.364 0.381 0.364 0.373 0.364 0.364
##  [49] 0.364 0.347 0.373 0.356 0.356 0.356 0.364 0.373 0.364 0.390 0.381 0.373
```

```
##  [61] 0.381 0.364 0.364 0.347 0.373 0.381 0.381 0.347 0.356 0.339 0.331 0.339
##  [73] 0.331 0.356 0.339 0.331 0.339 0.347 0.347 0.356 0.331 0.347 0.331 0.339
##  [85] 0.339 0.347 0.339 0.331 0.339 0.339 0.331 0.331 0.339 0.347 0.347 0.331
##  [97] 0.339 0.339 0.364 0.364
```

```
social.numberofneighbors <- max(social.allk[social.valerror==min(social.valerror)])

print(paste0('Best number of neighbors using LOOCV = ',social.numberofneighbors))
```

```
## [1] "Best number of neighbors using LOOCV = 12"
```

```
plot(x=social.allk,y=social.valerror,type='l',ylab="Test Error",xlab="K",col='blue2',
     main="Test Error vs. Number of Neighbors (Social Media)")
```



Train a KNN Classifier Using the Best Number of Neighbors Calculated with LOOCV (social media)

```
set.seed(69)
social.predYtest <- knn(train=sm.Xtrain,test=sm.Xtest,cl=sm.Ytrain,k=social.numberofneighbors)
social.confusionmatrix.bestk = table(predicted=social.predYtest,true=sm.Ytest)
social.confusionmatrix.bestk
```

```
##           true
## predicted High Low
##      High   28  23
##      Low    26  41
```

Calculate the Testing Error Rate with LOOCV (social media)

```
social.accuracy.bestk = sum(diag(social.confusionmatrix.bestk)/sum(social.confusionmatrix.bestk))
social.errorrate.bestk = 1-social.accuracy.bestk

print(paste0('Testing accuracy rate = ', round(social.accuracy.bestk,3)))
```

```
## [1] "Testing accuracy rate = 0.585"
```

```
print(paste0('Testing error rate = ', round(social.errorrate.bestk,3)))
```

```
## [1] "Testing error rate = 0.415"
```

Once again, after performing LOOCV and using that number for the number of neighbors, our accuracy rate is higher and our error rates are lower.

Conclusion

After performing KNN and setting k equal to the best number of neighbors found via LOOCV, it seems like salary has a lower error rate when predicting whether PIE (performance impact estimation) is high or low. This seems reasonable because NBA players who perform better are likely to be on better teams and hence paid more.