NBA players performance and social media influence

by Anesha Miroslawska, Palak Maniar and Aahad Abubarker

Abstract

The NBA data we are using has been collected from www.kaggle.com/datasets. The techniques we will be using are as follow regression and matrix operations on datasets, bias/ variance tradeoffs and regularized regression to combat overfitting, multicollinearity, factor analysis and factor rotation, PCA (principal component analysis), LDA (linear discriminant analysis), Lasso regression and Gradient boosting analysis, SHAP analysis.

Introduction

The NBA has been influential on social media for years and the dataset from 2017- 2018 is used in this analysis.

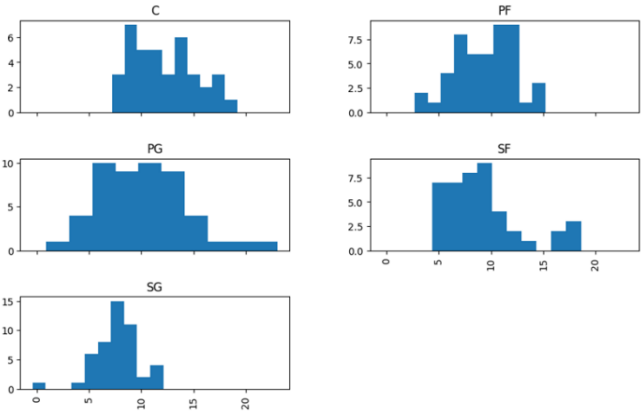
This study on NBA payer ranking and their influence on social media will help us better understand if a basketball player's score ranking an influencing factor on their social media popularity is or if their social media is influencing their performance and if is salary a factor.

The extensive dataset obtained from Kaggle, we have not just the salaries or the Twitter activities, but the team performance, the player performance, promotions, and win percentage, such factors adding up to 239 rows and 49 columns.

Since the dataset was small, but contained multiple variables, the unique approach for a machine learning algorithm that first came to mind was a tree model, or SVM, etc. Deep Learning approaches were considered but then ruled out due to 2 significant reasons.

- Lack of data
- Less interpretability of Deep Learning Algorithms like LSTMs.

Since the priority here is not just to make accurate predictions but also to be able to interpret the influencing factors and the logic behind those predictions. And it was evident that it would even be difficult to build a Deep learning model with the lack of instances and player information in the dataset. Hence after careful analysis and several trials and errors, I decided to implement the Gradient Boosting Regression model to make accurate predictions and to also interpret the coefficients that influence these predictions.



Various Multicollinearity plots are also drawn out to visualize the dependencies of our y (PIE) variable over the other variables. The captivating part of the following project is not the implementation of the Gradient Boosting regression Algorithm, but it is the SHAP analysis that was implemented on the result of this algorithm. Given the fact that I had to implement hyperparameter tuning, and various other measures to improve the accuracy of the model to a substantial extent; but the shap analysis was just as important to perfectly understand the influence of every variable in the model formulation. It scores the influence of these variables and shows the factors that are pulling down the results of supporting our final output predictions. It also helps us visualize these impacts across various instances of our computation helps us pick and choose the best one for optimal results.

Error Analysis of the model, along with outlier identification was also conducted and the observations regarding the same have also been made. Interpretable explanations for the same are also provided and the model has a lot of future potential to better analyses the performance and the behavioral patterns of several; NBA player or in other sports as well based on external factors like Salary and Social Media Influences

Another initial approach we came up with a correlation plot of the 100 NBA players and their game stats with twitter information as shown (figure 1.)

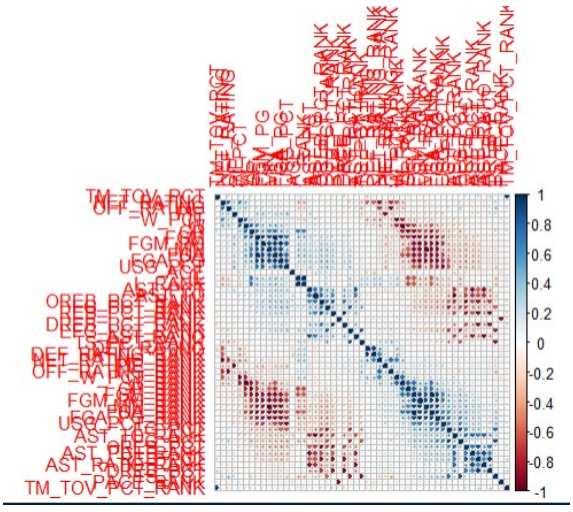


Figure: 1

Approach 1

When analyzing the initial regularized regression correlation plot of 100 NBA players I realized how big of a dataset this was, I also knew my group wanted to use 2 other csv files incorporated into this project. I wanted to take a different approach by making a new dataset using the variables I think are much more useful with finding the information of if social media plays a role in a player's popularity, performance, and salary. I combined the datasets and wrote a new csv file named "NBA1.csv." I then was able to use PCA and LDA techniques by player position as the category.

Analysis 1

At the initial stages of analysis of all the data files I made some graphs just to get different perspectives and more abstract ideas about this data.

The images below are an idea of what I saw when I was learning about the datasets provided. This is the "nba_2017_salary.csv" in different views.

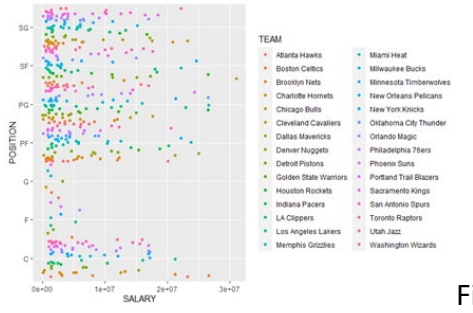


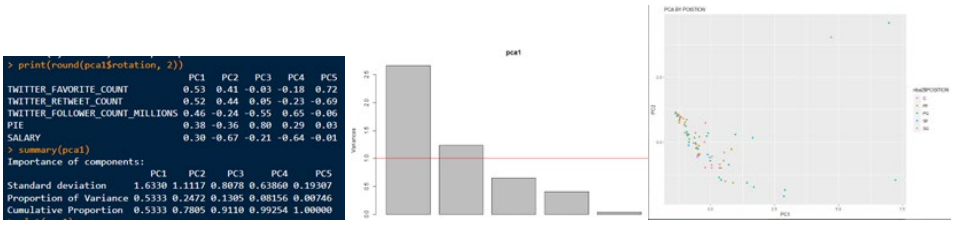
Fig: sal-1

Fig: sal-2

This gave me the idea to make a new dataset with the variables I will be using. I combined the NBA player dataset with the salary dataset and the twitter dataset. I grouped all the data by player and made sure we didn't have duplicate data and I had to fix the POSITION variables since many entries had whitespace in the data. This helped reduce the data that at one time was approximately 500 players by 90 variables to now 71 players with 8 variables.

```
> names(nba2) #71 players x 8 var
[1] "PLAYER"                    "TWITTER_FAVORITE_COUNT"
[3] "TWITTER_RETWEET_COUNT"     "TWITTER_FOLLOWER_COUNT_MILLIONS"
[5] "ACTIVE_TWITTER_LAST_YEAR"  "PIE"
[7] "SALARY"                    "POSITION"
```

I was now ready to do PCA, I first did the approach with variables in place 2: 7 and then realized that variable 4 "ACTIVE_TWITTER_LAST_YEAR" was skewing the data so I removed this variable. I made sure to make different views of the data in mind just so I can try different methods. Here are PCA results in different views in figures PCA-1:3.



Figures: PCA-1:3

These numbers and charts indicate to me that PC1 and PC2 have high importance. I also wanted to see the PCA in a different view, so I made a dimension plot that indicated position "SF" to be the most dominant categorical variable, as seen below in figure PCA-4, I was also able to make a 3D graph of PCA-5 as shown below.

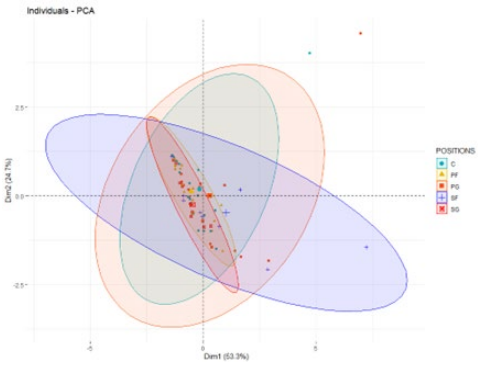


Figure: PCA-4

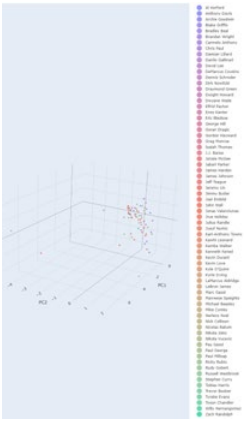
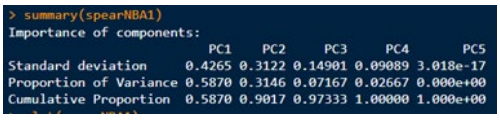
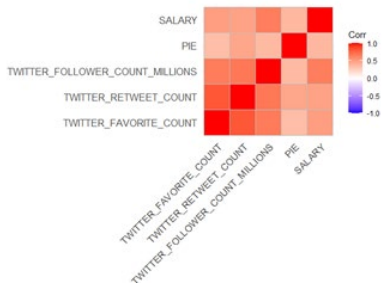


Figure: PCA-5

Next, I then made a PCA spearman and Pearson's correction as seen in figure: Spearman- 1:2 and Pearson's- 1:2 then I compared them. As you can see on the plots how they are both very correlated, but the spearman plot looks slightly more correlated however when you look at the importance of components you see they both only have PC1, but spearman has PC1 at a slightly higher importance.





Figures: Spearman- 1:2


```
> summary(pearNBA1)
Importance of components:
                          PC1    PC2     PC3     PC4       PC5
Standard deviation     0.6563 0.3278 0.21251 0.02194 1.686e-17
Proportion of Variance 0.7378 0.1840 0.07737 0.00082 0.000e+00
Cumulative Proportion  0.7378 0.9218 0.99918 1.00000 1.000e+00
```

Figures: Pearson's -1:2

Lastly, I was able to do LDA and this is the prediction of LDA player stats categorically by position.

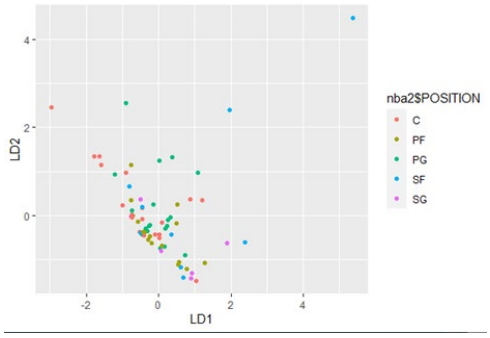


Figure: LDA-1

The next results are the accuracy prediction is 47% with user accuracy on SF at 60%. The next graph is used from the accuracy information table.

```
Confusion Matrix and Statistics

          Reference
Prediction  C PF PG SF SG
        C  18  0  0  0  1
        PF  0  7 10  0  0
        PG  0  7 13  0  0
        SF  0  3  4  2  0
        SG  0  4  1  1  0

Overall Statistics

               Accuracy : 0.5634
                 95% CI : (0.4405, 0.6809)
    No Information Rate : 0.3944
    P-Value [Acc > NIR] : 0.002909

                  Kappa : 0.4129

 Mcnemar's Test P-Value : NA

Statistics by Class:

                     Class: C Class: PF Class: PG Class: SF Class: SG
Sensitivity            1.0000   0.33333   0.4643   0.66667   0.00000
Specificity            0.9811   0.80000   0.8372   0.89706   0.91429
Pos Pred Value         0.9474   0.41176   0.6500   0.22222   0.00000
Neg Pred Value         1.0000   0.74074   0.7059   0.98387   0.98462
Prevalence             0.2535   0.29577   0.3944   0.04225   0.01408
Detection Rate         0.2535   0.09859   0.1831   0.02817   0.00000
Detection Prevalence   0.2676   0.23944   0.2817   0.12676   0.08451
Balanced Accuracy      0.9906   0.56667   0.6507   0.78186   0.45714
```

Figure: LDA-2

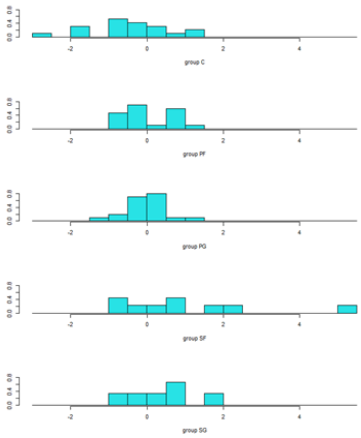


Figure: LDA-3

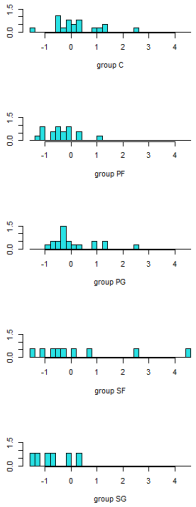


Figure: LDA-4

I then tried a different approach and got a LDA that gives more precise information regarding the player stats. I used the NBA1.csv file I made earlier and got better results in the LDA. As shown in figure LDA-6 the accuracy is much higher and in figure LDA-6 you can see how “PG” has the best rating.

```
Accuracy (PCC): 98.7654320987654%

Cohen's Kappa: 0.9843

Users accuracy:
    C     G    PF    PG    SF    SG
100.0 100.0  94.4 100.0 100.0 100.0


Producers accuracy:
    C     G    PF    PG    SF    SG
 95.7 100.0 100.0 100.0 100.0 100.0


Confusion matrix
    y
x     C  G PF PG SF SG
  C  22  0  1  0  0  0
  G   0  2  0  0  0  0
  PF  0  0 17  0  0  0
  PG  0  0  0 20  0  0
  SF  0  0  0  0 11  0
  SG  0  0  0  0  0  8
```

Figure: LDA- 5

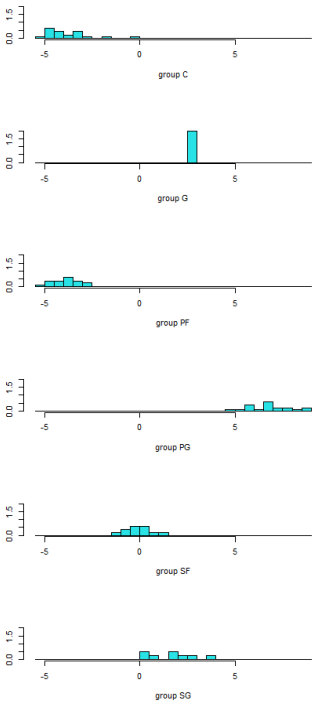


Figure: LDA-6

Approach 2

The method that I have personally implemented and worked on is the primary analysis of the variable-intensive dataset using the Gradient Boosting Regression Model. The Gradient Boosting regression model was used to interpret the accuracy of the predictions that the algorithm has to offer. Along with that, as a part of our Exploratory data analysis, we have also implemented several plots to visualize the interactions and multicollinearity between several variables. As a highlight of this method, a SHAP Analysis was also introduced and implemented to understand the impact of the various variables on the final prediction since there were plenty in the dataset. Majorly we observed whether factors like individual Salaries, Twitter follower counts, promotions, and social media activity show some kind of correlation with the player’s performance, measured in terms of (PIE) Player Impact Estimate.

Analysis 2

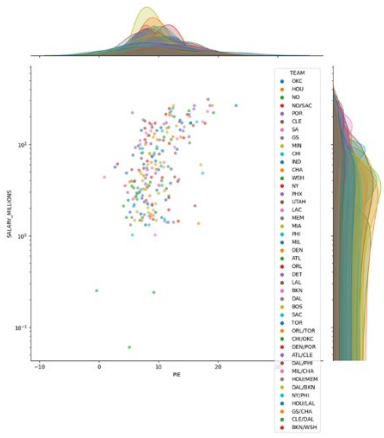
Exploratory Data Analysis

From the zip file of the various datasets provided, I have gone with the "nba_2017_players_with_salary_wiki_twitter" dataset. Just because the following dataset had the least amount of clutter and still managed to provide the most information in the form of relevant variables to the algorithm. The dataset was a right mix between several types of variables including integers for wins, Games played, etc; float values for PIE, and other such indexes; and object variables for variables like TEAM name, Player names, Position, etc. Hence conducting necessary EDA, some of these variables were dropped considering their irrelevance to the contribution of the model formulation. But I assumed the Position variable might play an important role in the performance and the salary of the player. And hence I decided to opt for one-hot encoding(dummy) variable creation for this particular variable.

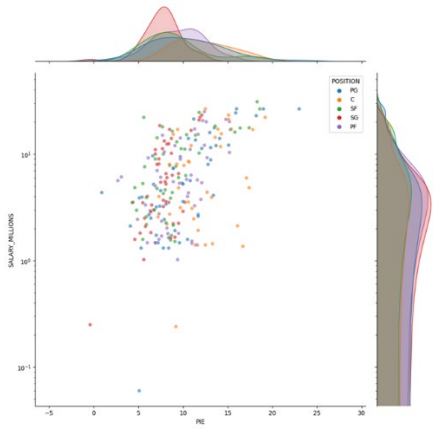
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 239 entries, 0 to 238
Data columns (total 42 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   Unnamed: 0      239 non-null    int64
 1   Rk              239 non-null    int64
 2   PLAYER          239 non-null    object
 3   POSITION        239 non-null    object
 4   AGE             239 non-null    int64
 5   MP              239 non-null    float64
 6   FG              239 non-null    float64
 7   FGA             239 non-null    float64
 8   FG%             239 non-null    float64
 9   3P              239 non-null    float64
 10  3PA             239 non-null    float64
 11  3P%             232 non-null    float64
 12  2P              239 non-null    float64
 13  2PA             239 non-null    float64
 14  2P%             239 non-null    float64
 15  eFG%            239 non-null    float64
 16  FT              239 non-null    float64
 17  FTA             239 non-null    float64
 18  FT%             237 non-null    float64
 19  ORB             239 non-null    float64
 20  DRB             239 non-null    float64
 21  TRB             239 non-null    float64
 22  AST             239 non-null    float64
```

```
 23  STL                    239 non-null    float64
 24  BLK                    239 non-null    float64
 25  TOV                    239 non-null    float64
 26  PF                     239 non-null    float64
 27  POINTS                 239 non-null    float64
 28  TEAM                   239 non-null    object
 29  GP                     239 non-null    int64
 30  MPG                    239 non-null    float64
 31  ORPM                   239 non-null    float64
 32  DRPM                   239 non-null    float64
 33  RPM                    239 non-null    float64
 34  WINS_RPM               239 non-null    float64
 35  PIE                    239 non-null    float64
 36  PACE                   239 non-null    float64
 37  W                      239 non-null    int64
 38  SALARY_MILLIONS        239 non-null    float64
 39  PAGEVIEWS              239 non-null    float64
 40  TWITTER_FAVORITE_COUNT  236 non-null    float64
 41  TWITTER_RETWEET_COUNT   236 non-null    float64
dtypes: float64(34), int64(5), object(3)
memory usage: 78.5+ KB
Dataset (df1) Dimensions = (239, 42)
```

Salary distribution Based on the Team



Salary distribution Based on the Player position



Discussions: The following two graphs shows us the direct relation between the salary of a player and his impact to the game inn to the form of PIE. The color of the scatterplot helps us interpret the position the player is playing for (in the second graph) and the team that the player is playing for (in the first graph).

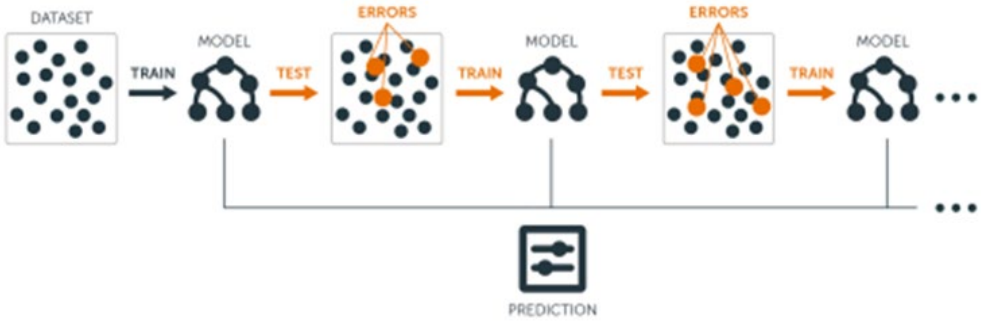
This helps us interpret that the high-performing players usually have higher salaries with some exceptions that can be observed in the graph. Along with that other observations like the SF C and PG are more of the higher paying positions which also perform well on the court.

As a part of the EDA, several Multicollinearity plots were also made to analyze the relations between these variables mathematically and prove our findings.

Application of chosen Analysis Technique

Approach not taught in class: Gradient Boosting Regression Model.

Gradient Boosting Regression is a machine learning algorithm that combines multiple weak predictive models (typically decision trees) to create a strong predictive model. It is a popular algorithm for regression tasks and is known for its effectiveness in handling complex, nonlinear relationships between features and target variables.



Given these considerations, the Gradient Boosting Regression model was deemed appropriate. It offers a balance between prediction accuracy and interpretability. The

model works by iteratively building weak learners, typically decision trees, to capture the relationships between the features and the target variable. The iterative nature of the algorithm allows it to learn from the mistakes of previous models and improve its predictions.

The application of the Gradient Boosting Regression model in this context aimed to make accurate predictions while providing insights into the influencing factors. The model's coefficients can be examined to understand the relative importance of different features in making predictions. By analyzing the coefficients, one can interpret the underlying logic behind the predictions and gain insights into the factors that contribute the most to the target variable.

The model was implemented through a process of careful analysis, trial, and error. The specific hyperparameters and settings of the Gradient Boosting Regression model were tuned to optimize its performance on the given dataset. By employing this approach, the aim was to achieve accurate predictions while also facilitating the interpretation of the model's behavior and decision-making process.

Hyperparameters (Ideal, chosen post experimentation)

- Train - Test Split: 80% - 20%
- No. of Estimators: 25
- Alpha = 0.2
- Error Metrics: Mean Squared error

- Accuracy Evaluation: R-Squared.

Residual analysis/model diagnostics

The Best Model gave the following values for its predictions.

Training Mean Squared Error (MSE): 0.852

Training R-squared (R2) Score: 0.934

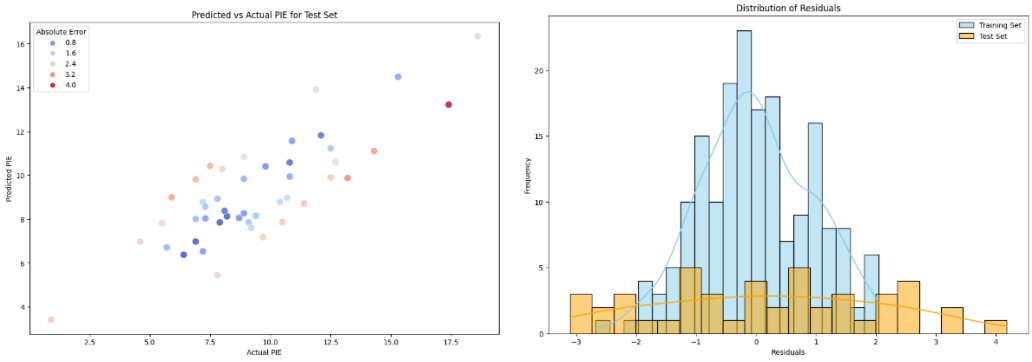
Testing Mean Squared Error (MSE): 3.558

Testing R-squared (R2) Score: 0.659

Mean Squared Error (MSE): MSE measures the average squared difference between the predicted values and the actual values in the dataset. It quantifies the overall quality of the model's predictions. A lower MSE indicates better prediction accuracy, as it means the model's predictions are closer to the actual values. In your case, the MSE for the testing set is 3.558, which suggests that, on average, the squared difference between the predicted values and the actual values is 3.558.

R-squared (R2) Score: R-squared is a statistical measure that represents the proportion of the variance in the target variable that is predictable from the independent variables (features) in the model. It indicates the goodness of fit of the model to the data. R2 score ranges from 0 to 1, where 1 indicates a perfect fit and 0 indicates that the model does not explain any of the variances in the target variable. In this case, the R2 score is 0.659, which means that approximately 65.9% of the variance in the target variable is explained by the independent variables in the model. This indicates a reasonably good fit of the model to the data.

But observing the MSE value for training data of 0.852 and the R-squared value of 0.934 (93.4%) for the same data, it means that even after trying out various training and testing splits and several hyperparameter tuning techniques the model is still overfitting up to some extent. A major remedy to this problem, which is also a limitation for the following Project, is increasing the size of the dataset. With more training data, regularization, feature selection and feature augmentation the overfitting problems can be tackled.



The following plots can also be utilized for the error calculation and Accuracy Interpretation of the model.

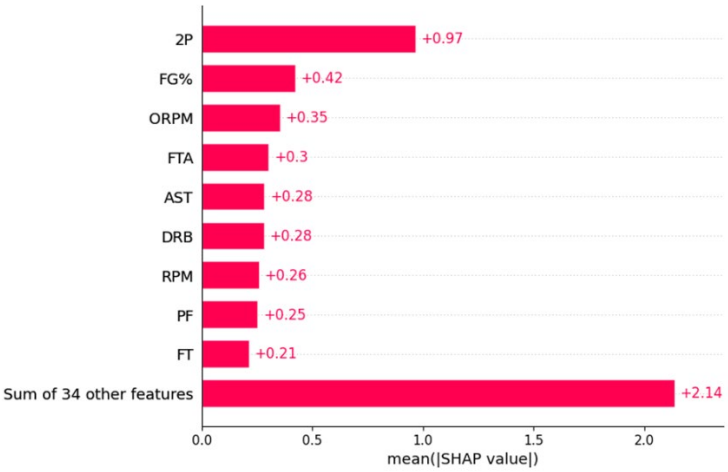
SHAP Analysis

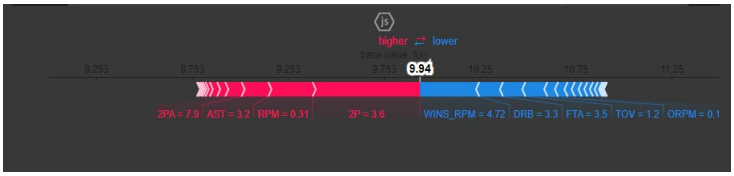
SHAP (SHapley Additive exPlanations) analysis is a method used to explain the predictions of machine learning models by assigning importance or contributions to each input feature. It provides insights into the impact of different features on the model's predictions and helps understand the underlying logic behind the model's decisions.

The main idea behind SHAP analysis is based on the concept of Shapley values from cooperative game theory. Shapley values determine the contribution of each feature by considering different permutations and combinations of features and measuring their impact on the model's predictions.

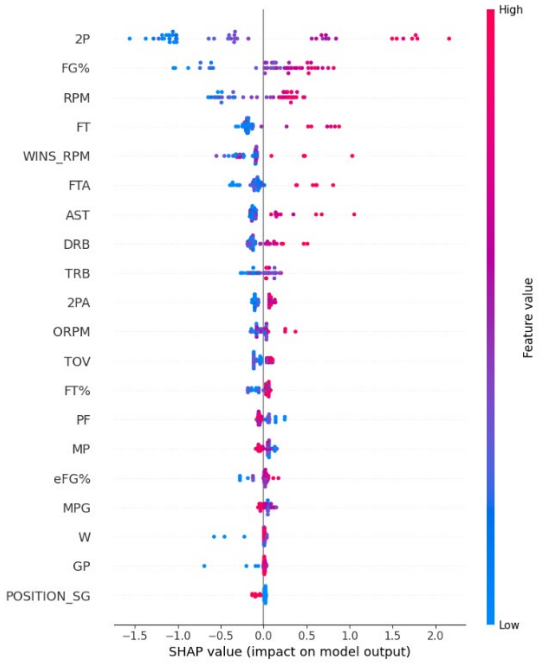
Hence to get an idea of the Interpretability of the impact of various variables on our prediction we have the following Shap analysis implemented on the features of our model.

Its results can be observed as follows.





This helps us visualize the variables for instance values and their contributions to the results of our model.



Hence the model Importance of each variable is seen in the following graph. But from these which are contributing positively to the model, and which are contributing negatively is not to be seen. For that, the second plot is plotted as such.

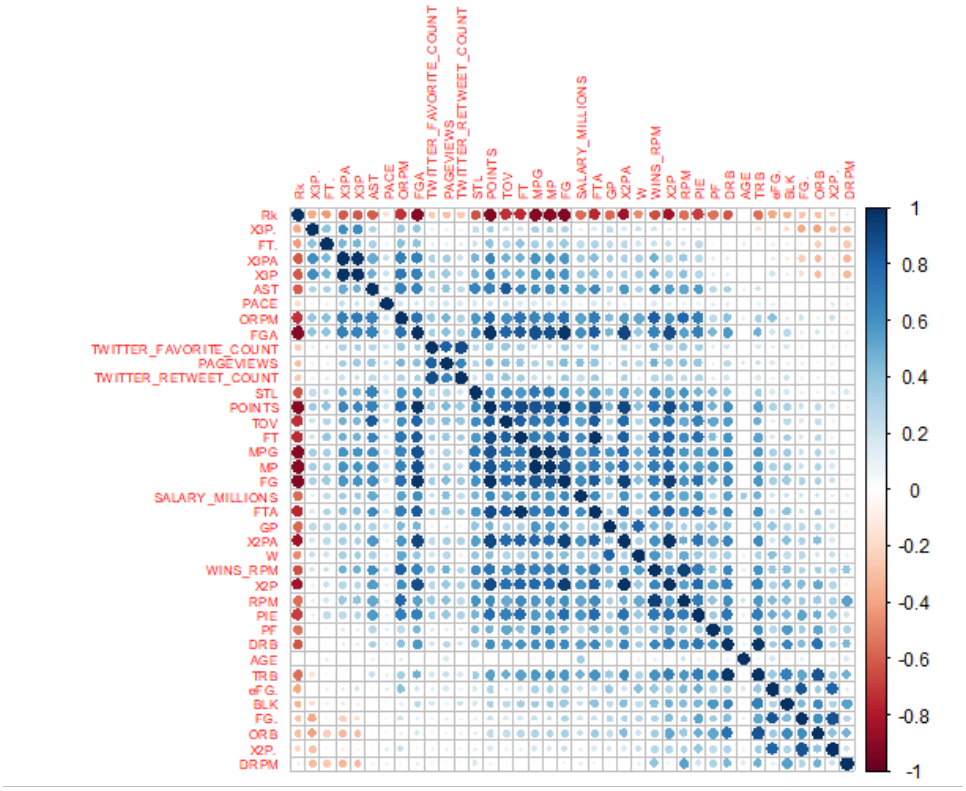
Approach 3

In my section of analysis on the NBA dataset, I will attempt to run a regularized regression on the Twitter+Salary dataset of ~250 players. I used Lasso Regression for this dataset because currently there are about 40 predictors, causing interpretability of regression models to be difficult. With Lasso's ability to select predictors while adjusting for overfitting, the model it creates will be much more parsimonious than an ordinary least squares model or a ridge regression which only handles overfitting. In this analysis, I split the dataset into a training and test set to understand the performance of Lasso.

Analysis 3

In the initial dataset for NBA players with twitter information, there were some empty values in some predictors. For example, Draymond Green did not have a twitter account in the 2017-2018 NBA season, so we had to set values pertaining to Twitter account engagement to 0. We went with 0 instead of a mean or median value insert because we did not want artificial values to be created in order to predict salary as it can become complex very fast if we set a mean value for a player who has a NA for 3-point percentage. 3-point percentage is very important to the strategy of the modern NBA.

Before beginning with the initial regression, I produced a correlation plot order of eigenvectors to easily see the correlation between variables. A visual plot like this allows an initial view of the data. In this analysis, we are looking to predict the salary, in millions, of an NBA player from predictors such as age, on court performance and twitter stats.



There are some things to note from the correlation plot. Here, we can see that twitter statistics, such as Twitter_favorite_count and twitter_retweet_count, are weakly correlated with other variables. Looking at the variables correlated with age, it may seem that salary may be correlated with age however, using knowledge of the NBA, note that players who are playing in their later ages are usually important pieces to a team so they would be paid substantially more.

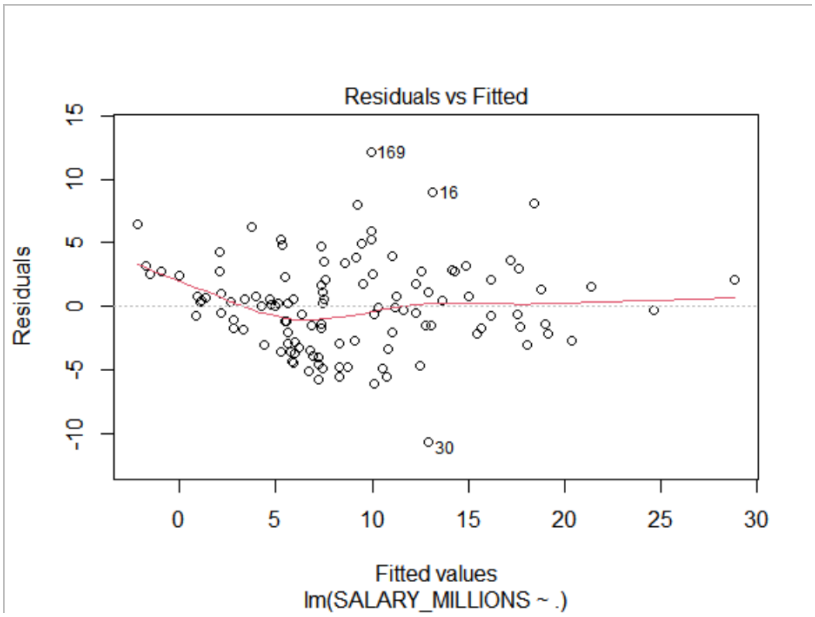
In the regularized regression analysis, I first created a model using ordinary least squares method to baseline the performance of Lasso regression. In our training and test sets, I split the data by separating half of the NBA dataset into training and the other half into test. Using OLS with training set, first we can see that our R^2 is 0.723 with an RMSE of 3.53. The RMSE for olsTest was 5.85, we can see that the RMSE nearly doubles, which means that there is overfitting in the model.

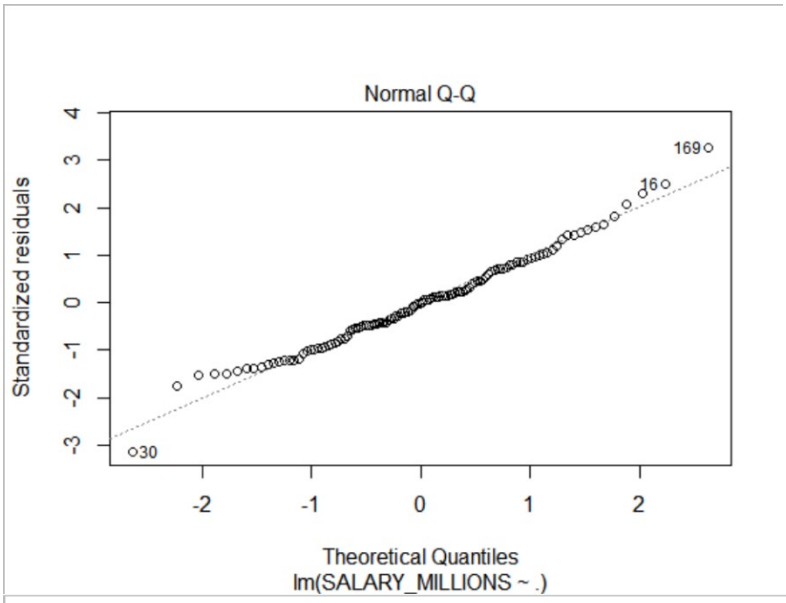
```
PACE                      -2.761e-01  2.332e-01  -1.184   0.2399
W                          2.365e-02  7.268e-02   0.325   0.7457
PAGEVIEWS                  8.182e-04  5.999e-04   1.364   0.1763
TWITTER_FAVORITE_COUNT     1.409e-04  3.651e-03   0.039   0.9693
TWITTER_RETWEET_COUNT     -1.099e-04  8.987e-03  -0.012   0.9903
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 4.252 on 82 degrees of freedom
Multiple R-squared:  0.723,     Adjusted R-squared:  0.6014
F-statistic: 5.946 on 36 and 82 DF,  p-value: 1.298e-11
```

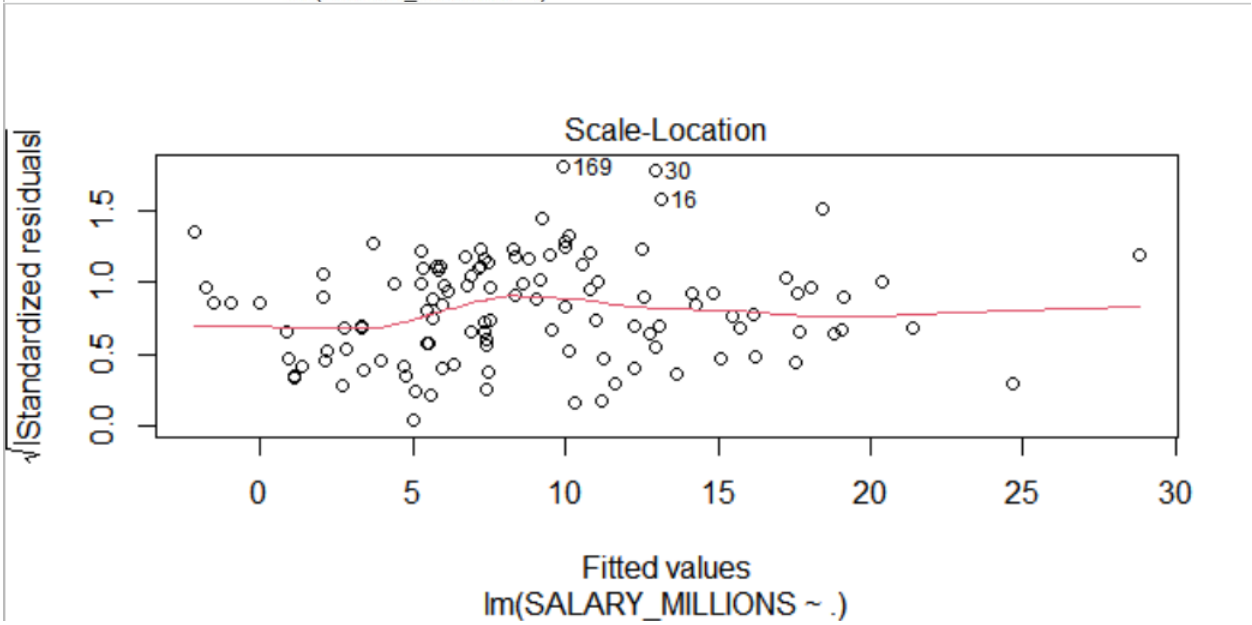
```
> rmseOlsTrain
[1] 3.529862
```

```
> rmseOlsTest
[1] 5.853848
```

Also, to determine the adequacy of the model, I looked at the residuals for the OLS model.



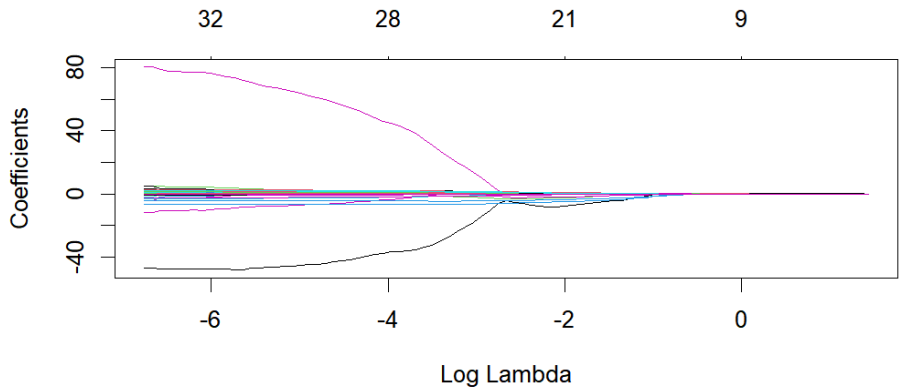


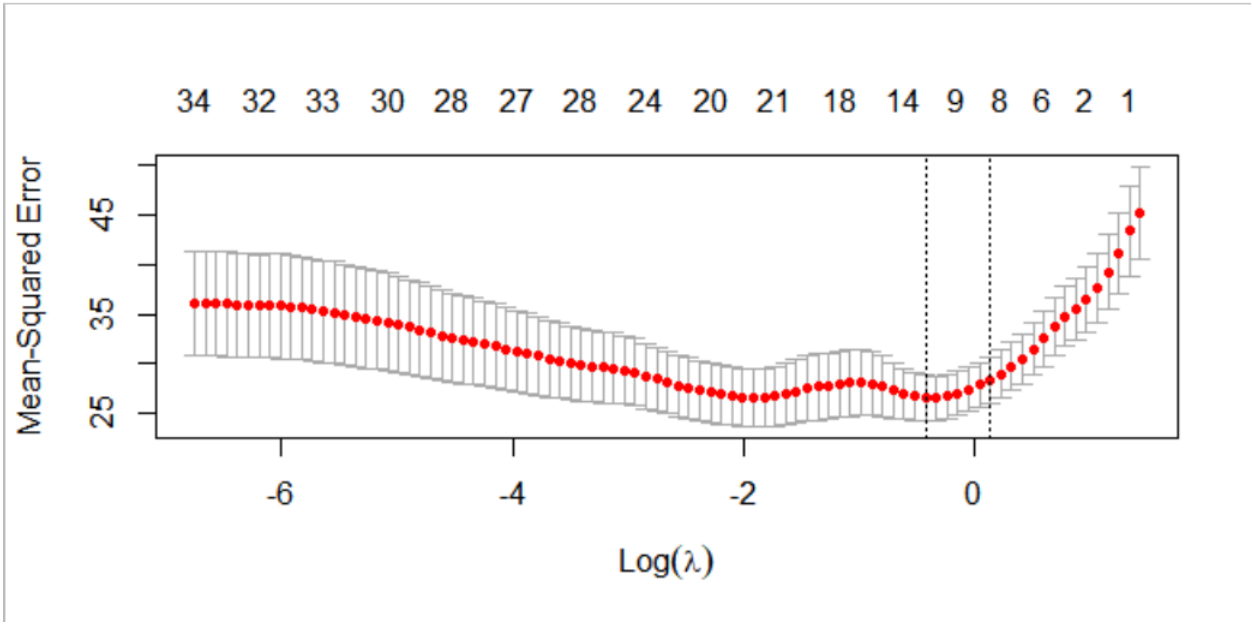


Generally, we see some randomness to the spread and independence for the severity of the spread but there are some clusters of points that we are seeing. These can perhaps be fixed by variable transformation in the subsequent model. However, evidently there is a large increase in RMSE values from training set to test set indicating overfitting.

Therefore, we must use regularized regression in our analysis, and Lasso's ability to select influential predictors and reduce overfitting is the regression technique that will help our model the most.

To create a Lasso regression, I first separated the rows and columns and placed them into their respective matrices, training, and test. To choose Lasso regression in the glmnet library, I used alpha = 1, for solely Lasso, not relaxed Lasso. To visualize the impact of Lasso, I created a plot of coefficients and lambda, showing how lambda increases to -3 and starts to level out. We also look at the mean squared error plot to select our lambda.





In the coefficients vs lambda plot, it shows that as lambda reaches a certain value between -4 and -2, the coefficients converge towards 0. However, in the case of the MSE vs log(lambda) plot, something much more beneficial is displayed. Both lambda.1se and lambda.min are visualized against MSE, so we can select which lambda value to use for our model. Although they are both relatively low, we can see that after lambda.min the MSE starts to go up at a rapid pace. Since we are looking to reduce the error in our model, lambda.1se looks to be more advantageous.

Next we used lambda selection in cross validated glmnet to find lamda.1se and lamda.min to use in our Lasso fitted model.

	Lambda	Index	Measure	SE	Nonzero
min	0.147	37	23.89	3.621	21
1se	1.037	16	27.38	4.430	8

Lambda.1se was 1.037 and Lambda.min was 0.147. Since we wanted a more parsimonious model to help with interpretability and generalization of the data, I chose to use lambda.1se, which is the largest value of lambda within one standard error or the minimum cross validated error.

When we look at the model concerning each lambda value and the amount of variance and number of predictors selected we get a dramatic difference in each lambda.

```
61 28 69.53 0.0158
62 28 69.63 0.0144 16  8 49.54 1.0370
```

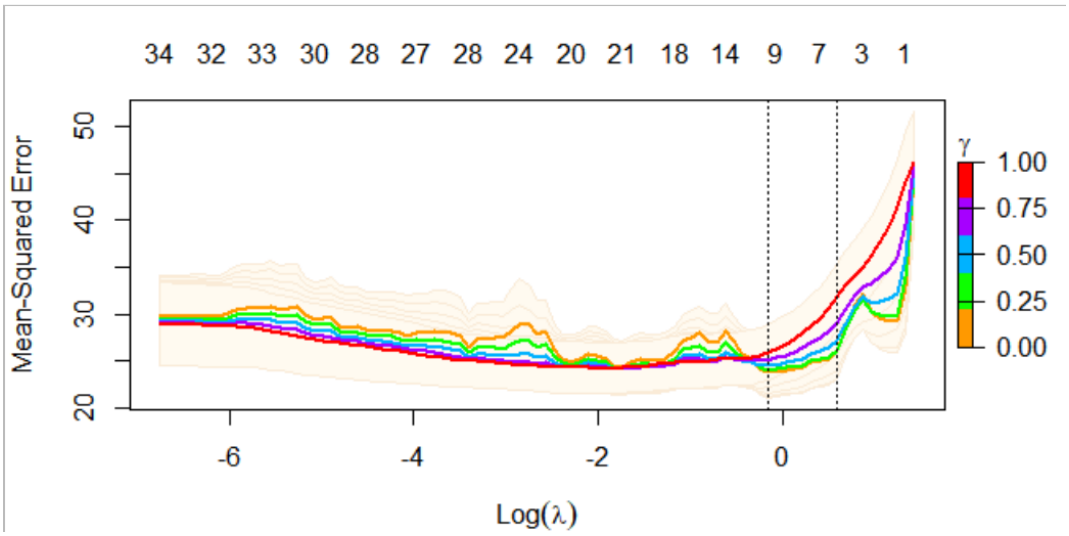
In Lambda.min, we get an R^2 of 0.70, which isn't too large to indicate any evidence of overfitting on the model; however, it only removed about 12 predictors from the full model. In a dataset with near 40 independent variables, and the goal of predicting the salary of a player, 28 predictors will be too many to adequately interpret. In the case of Lambda.1se, we may get a lower R^2 value of 0.50, but the major benefit is that it reduces the number of predictors to only 8 which helps our goal of creating a parsimonious model.

Still, to compare the performance of both, I compared the RMSELasso of "1se" and "min", 5.08 and 5.20 respectively.

```
> rmseLasso1se
[1] 5.079689
> rmseLassoMin
[1] 5.196814
>
```

We can see that there was still overfitting in using Lasso as the training set using OLS had an RMSE of 3.53 and Lasso's RMSEs are slightly higher. However, one benefit of using Lasso was that it decreased the amount of overfitting from the test set compared to OLS as OLS' RMSE was nearly double that of the training set, and using Lasso decreased the RMSE.

Another aspect of Lasso regression is using relaxed Lasso, which is an in between Ridge and Lasso regularized regression. To create a model using our dataset of NBA players, the prediction of salary using relaxed Lasso, we first visualized the improvements in its root mean squared error and overfitting reduction.



The visualized plot of MSE vs Log(lambda) was very similar to the vanilla Lasso reguralized model where we see a sharp increase in error after Lambda.min. This can be an indication that relaxation may not have a meaningful impact in our analysis. Since this is a new model, we must select our lambda values again.

```
     Gamma Index Lambda Index Measure    SE Nonzero
min   0.25     2  0.147    37   24.45 4.193      21
1se   0.50     3  1.989     9   28.33 4.139       4
> 
31 20 61.49  64.85 0.2569
24 13 55.65  59.73 0.4927
```

In the case of relaxation on our model, we see that there are actually differences in the number of predictors selected and the variance captured. When compared to the vanilla Lasso model, the lambda.1se relaxed Lasso model chooses 5 more predictors in order to capture only 5% more variance. In the case of our analysis, such a small increase in variance is not worth adding 5 predictors as to decrease complexity and interpretability, However, Lambda.min looks to be more appealing as it decreases the number of predictors from the vanilla Lasso from 28 to 20, while only losing 5% variance. Relaxed Lasso is much more beneficial for our analysis if our goal was to capture the most amount of variance possible.

Our findings after exploring OLS regression and the performance of vanilla Lasso regression and relaxed Lasso were that Lasso regression reduces the number of predictors in the model to 8 using Lambda.1se making the model more parsimonious. It also reduces the amount of overfitting as we see a decrease in RMSE on the test set using Lasso, in both Lambda.min and Lambda.1se. In the end, our model becomes more parsimonious and helps interpretability.

Conclusion

We studied the NBA player's performance and their popularity on social media. From our findings, it is apparent that factors like Salary and Player position do have an influence on the player's impact and performance. Although some of the findings here point out the low influence of social media on these performance indexes there are also new patterns that emerge. Factors like 2 Field Goal (2P) and Offensive Real Plus-Minus (ORPM) are observed to be highly influential to the model performance and output. As a part of the project outcome, we now also have a Gradient Boosting Regression Model that computes feature importance and predicts the PIE (Player Impact Estimate) based on all the other variables present in the dataset.

Approach 1:

- PCA shows players in poison "SF" (small forward) are the highest paying and the most popular on social media.
- LDA showed players stats in position "PG" to be the highest rated.

Approach 2:

- A Gradient Boosting regression model can be utilized to make accurate predictions for the PIE of a player.
- The factors that are the most influential in making these predictions.

Approach 3:

- Reduction in model predictors to make model more parsimonious and reduce overfitting in regularized regression for predicting salary

References

The NBA players performance and social media influence in 2017- 2018
Source: https://www.kaggle.com/datasets/noahgift/social-power-nba
CSV files used:
nba_2016_2017_100
nba_2017_twitter_players
nba_2017_salary
nba_2017_players_with_salary_wiki_twitter.csv

Appendix A

This appendix consists of information from the NBA social power dataset. Our research shows factors that influence the players performance and their popularity on social media.

Approach 1 DATA

```
nba <- read.csv("nba_2016_2017_100.csv")
cor.nba= cor(nba[5:55])
Figure 1 :
corrplot(cor.nba, order="AOE")
nbaSalary <- read.csv("nba_2017_salary.csv")

# CHECK THE DATA
names(nbaSalary)
any(duplicated(nbaSalary)) # True
#checking for missing values
sum(is.null(nbaSalary)) # 0
nbaSalary[duplicated(nbaSalary$NAME),]

# Impute DOUBLE VALUES of SALARY with mean(SALARY) - (some of the salary for
#players had multi salary for players, got the mean for the salary)
nbaSalary <- nbaSalary %>%
  group_by(NAME, POSITION, TEAM) %>%
  mutate(SALARY = mean(SALARY, na.rm =  T)) %>%
  mutate(POSITION = trimws(POSITION)) # takes away white spaces (this was done due
to csv having same positions  )
plot(nbaSalary)
plot(nbaSalary[2:4])
Figure: sal -1
### PLOT POS VS SAL
ggplot(nbaSalary, aes (x=SALARY, y= TEAM, color=POSITION)) + geom_point(position =
position_dodge(width = .5))
nbaSalary
Figure: sal-2
# TEAM and SALARY BY POSITION
ggplot(nbaSalary, aes (x=SALARY, y=POSITION , color=TEAM)) + geom_point(position =
position_dodge(width = 1))

nba1 <- merge (nba, nbaSalary[,-3], by=intersect(nba$PLAYER_NAME,
nbaSalary$NAME), by.x="PLAYER_NAME", by.y="NAME" , sort= TRUE  )
nba1= nba1[,-56]
nba1
# CHECK FOR DOUBLES
any(duplicated(nba1)) # FALSE
```

```
nba1

# write as csv file
write.csv(nba1, "nba1.csv", row.names=T)
nba1 = read.csv("nba1.csv")
twitter <- read.csv("nba_2017_twitter_players.csv")
nba2 <- merge (twitter , nba1[,c('PLAYER_NAME' ,
'TWITTER_FOLLOWER_COUNT_MILLIONS', 'ACTIVE_TWITTER_LAST_YEAR', 'PIE',
'SALARY', 'POSITION' )], by.x='PLAYER',by.y="PLAYER_NAME")
names(nba2) #71 players x 8 var
str(nba2)

# PCA
Figure: PCA-1:3
pca1= prcomp(selectedNBA, scale= T) # selectedNBA
str(pca1)
print(round(pca1$rotation, 2))
summary(pca1)
plot(pca1)
abline(1,0, col="red") #Plot line
p_pca= pca1$x %>% as.data.frame() %>%
  ggplot(aes(x= PC1, y=PC2,  color= nba2$POSITION))+ geom_point()+ labs(subtitle =
"PCA BY POISTION")
p_pca
library("factoextra")
fviz_pca_var(pca1, col.var= "#00AFBB")
fviz_pca_ind(pca1)

Figure: PCA-4
fviz_pca_ind(pca1,
         geom.ind = "point", # show points only (nbut not "text")
         col.ind = nba2$POSITION, # color by groups
         palette = c("#00AFBB", "#E7B800", "#FC4E07", "blue", "red"),
         addEllipses = TRUE, # Concentration ellipses
         legend.title = "POSITIONS")

# CORRELATION - SPEARMAN
Figures: Spearman- 1:2
spearNBA= cor(selectedNBA, method="spearman")
dist(spearNBA)
summary(spearNBA)
ggcorrplot(spearNBA)
spearNBA1= prcomp(spearNBA)
summary(spearNBA1)
```

```
plot(spearNBA1)

# pearson
```
Figure: Pearson's- 1:2
```
pearNBA= cor(selectedNBA)
pearNBA
dist(pearNBA)
ggcorrplot(pearNBA)

pearNBA1= prcomp(pearNBA)
summary(pearNBA1)
plot(pearNBA1)
```
Figure: PCA- 5
```
components <- pca1[["x"]]
components <- data.frame(components)
components$PC2 <- -components$PC2
components$PC3 <- -components$PC3
components = cbind(components, nba2$PLAYER)
tot_explained_variance_ratio <- summary(pca1)[["importance"]]['Proportion of
Variance',]
tot_explained_variance_ratio
tot_explained_variance_ratio <- 100 * sum(tot_explained_variance_ratio)
tot_explained_variance_ratio
tit = 'Total Explained Variance = 100.001'
fig <- plot_ly(components, x = ~PC1, y = ~PC2, z = ~PC3, color =
~components$`nba2$PLAYER`, colors = c('#636EFA','#EF553B','#00CC96') ) %>%
  add_markers(size = 2)
fig <- fig %>% layout(title = tit,scene = list(bgcolor = "#e5ecf6"))
fig
```
Figure: LDA-1
```
# LDA categorical
nbaLDA= lda(POSITION ~ . - PLAYER - ACTIVE_TWITTER_LAST_YEAR ,  data = nba2,
output = "Scatterplot")
nbaLDA
coef(nbaLDA)
#LDA PREDICTION
predLDA= predict(nbaLDA, newdata= nba2) # How does it do on the training set?
names(predLDA)
# training
table(predLDA$class, nba2$POSITION)
# accuracy measures
```
Figure: LDA- 2
```
accuracy(nba2$POSITION, predLDA$class)
```

```
plot(predLDA$x, col= predLDA$class)
predLDA$class
p_lda= predLDA$x %>% as.data.frame() %>% ggplot(aes(x= LD1, y=LD2,  color=
nba2$POSITION))+
  geom_point()
p_lda
```
Figure: LDA-3/ LDA-4
```
library(MASS)
ldahist(predLDA$x[,1], g= nba2$POSITION)
ldahist(predLDA2$x[,2], g= nba2$POSITION)

# FIXED nba1- LDA
names(nba1)
nbaLDA1= lda(POSITION ~ . -PLAYER_ID  - PLAYER_NAME - TEAM_ABBREVIATION -
CFPARAMS -TWITTER_FOLLOWER_COUNT_MILLIONS -WIKIPEDIA_HANDLE -
TWITTER_HANDLE,data = nba1, output = "Scatterplot")
nbaLDA1
coef(nbaLDA1)
#LDA PREDICTION
predLDA1= predict(nbaLDA1, newdata= nba1) # How does it do on the training set?
names(predLDA1)
# training
table(predLDA1$class, nba1$POSITION)
```
Figure: LDA-5
```
# accuracy measures
accuracy(nba1$POSITION, predLDA1$class)
plot(predLDA1$x, col= predLDA1$class)
predLDA1$class
p_lda1= predLDA1$x %>% as.data.frame() %>% ggplot(aes(x= LD1, y=LD2,  color=
nba1$POSITION))+
  geom_point()
p_lda1
```
Figure: LDA-6
```
ldahist(predLDA1$x[,1], g= nba1$POSITION)
ldahist(predLDA1$x[,2], g= nba1$POSITION)
```

Approach 2 DATA (Techniques not covered in class)

```
df1 = nba_2017_players_with_salary_wiki_twitter.csv
import pandas as pd
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.ensemble import RandomForestRegressor
```

```
from sklearn.metrics import mean_squared_error, r2_score
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

df1 = pd.read_csv('nba_2017_players_with_salary_wiki_twitter.csv')
df1.info()
df1.shape

df1.isna()
df1 = df1.dropna()
categorical_vars = ['POSITION']

# Perform one-hot encoding
df1_encoded = pd.get_dummies(df1, columns=categorical_vars)

#Try dropping PIE_Rank since it is directly highly correlated with PIE and hence skews the feature importance of the model and it's results.

features1 = df1_encoded.drop(columns = ['PLAYER','PIE','TEAM'])
target1 = df1_encoded['PIE']


sns.jointplot(data=df1.reset_index(), y='SALARY_MILLIONS',
          x='PIE', hue='TEAM', height=10, palette='tab10',
          alpha=0.6)
plt.yscale('log')


sns.jointplot(data=df1.reset_index(), y='SALARY_MILLIONS',
          x='PIE', hue='POSITION', height=10, palette='tab10',
          alpha=0.6)
plt.yscale('log')

df1[['POSITION', 'PIE']]\
   .reset_index(drop=True)\
   .hist(by='POSITION',figsize=(11,7), sharex=True)
plt.show()

df1.pivot_table(
   values='SALARY_MILLIONS',
   index='PIE',
   columns='POSITION',
   aggfunc='count')\
```

```
      .style.background_gradient(cmap='Blues',axis=None)

df1.corr()\
   .style.background_gradient(
      cmap='RdBu',
      axis=None)

from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import GradientBoostingRegressor

X_train1, X_test1, y_train1, y_test1 = train_test_split(features1, target1, test_size=0.2,
random_state=42)

# Gradient Boosting Regression model
gb_model1 = GradientBoostingRegressor(n_estimators=25, random_state=42, alpha =
0.2)

# Train the model
gb_model1.fit(X_train1, y_train1)

#Predictions on Training set
y_train_pred1 = gb_model1.predict(X_train1)

# Evaluation Metrics
mse_train1 = mean_squared_error(y_train1, y_train_pred1)
r2_train1 = r2_score(y_train1, y_train_pred1)


print("Training Mean Squared Error (MSE):", mse_train1)
print("Training R-squared (R2) Score:", r2_train1, "\n")

# Make predictions on the test set
y_gb1 = gb_model1.predict(X_test1)

# Evaluation Metrics
mse1 = mean_squared_error(y_test1, y_gb1)
r21 = r2_score(y_test1, y_gb1)

print("Testing Mean Squared Error (MSE):", mse1)
print("Testing R-squared (R2) Score:", r21)

# Visualize predicted vs actual values on the test set
plt.figure(figsize=(12, 8))
```

```
sns.scatterplot(x=y_test1, y=y_gb1, hue=np.abs(y_test1 - y_gb1), s=100, alpha=0.8,
palette='coolwarm')
plt.xlabel('Actual PIE')
plt.ylabel('Predicted PIE')
plt.title('Predicted vs Actual PIE for Test Set')
plt.legend(loc='upper left', title='Absolute Error')
plt.show()

# Calculate and visualize the residuals
residuals_train1 = y_train1 - y_train_pred1
residuals_test1 = y_test1 - y_gb1

plt.figure(figsize=(12, 8))
sns.histplot(data=residuals_train1, bins=20, kde=True, color='skyblue', label='Training
Set')
sns.histplot(data=residuals_test1, bins=20, kde=True, color='orange', label='Test Set')
plt.xlabel('Residuals')
plt.ylabel('Frequency')
plt.title('Distribution of Residuals')
plt.legend()
plt.show()

import matplotlib.pyplot as plt
import seaborn as sns
import numpy as np
plt.figure(figsize=(12, 8))
sns.scatterplot(x=y_test1, y=y_gb1, hue=np.abs(y_test1 - y_gb1), s=100, alpha=0.8,
palette='coolwarm')
plt.xlabel('Actual PIE')
plt.ylabel('Predicted PIE')
plt.title('Predicted vs Actual PIE for Test Set')
plt.legend(loc='upper left', title='Absolute Error')
plt.show()

abs_error = np.abs(y_test1 - y_gb1)
plt.figure(figsize=(12, 8))
sns.histplot(data=abs_error, bins=20, kde=True, color='skyblue')
plt.xlabel('Absolute Error')
plt.ylabel('Frequency')
plt.title('Distribution of Absolute Error for Test Set')
plt.show()

!pip install shap
```

```
import shap
shap.initjs()

# SHAP analysis
explainer = shap.Explainer(gb_model1)
shap_values = explainer.shap_values(X_test1)

# Summary plot
shap.summary_plot(shap_values, X_test1, feature_names=X_test1.columns)

# Force plot for a specific instance
instance_index = 0 # Change this to the index of the instance you want to visualize
shap.force_plot(explainer.expected_value, shap_values[instance_index],
X_test1.iloc[instance_index])

# Show the plots
plt.show()

shap.initjs()
instance_index = 1 # Change this to the index of the instance you want to visualize
shap.plots.force(explainer.expected_value, shap_values[instance_index],
X_test1.iloc[instance_index])

# Create an explainer object
explainer = shap.Explainer(gb_model1)

# Calculate SHAP values for the test set
shap_values = explainer(X_test1)

# Plot the bar plot of SHAP values
shap.plots.bar(shap_values)
```

Approach 3 DATA

```
allStats <- read.csv("nba_2016_2017_100.csv")
teamValue <- read.csv("nba_2017_team_valuations.csv")
twitter <- read.csv("nba_2017_twitter_players.csv")
salary  <- read.csv("nba_2017_salary.csv")

head(allStats)
allStats[,30]
corrplot(cor(allStats[,5:55]), order="AOE")
```

```
allSFilter <- allStats[,5:55]
model1 = lm(W ~ . -L, GP, W_PCT, data = allSFilter)
summary(model1)


salaryTwitter100 <- left_join(allStats, twitter, by = c("PLAYER_NAME" = "PLAYER"))
head(salaryTwitter100)

playerPrf <- c("FGM",
           "OFF_RATING", "DEF_RATING",
           "NET_RATING", "MIN", "AGE",
           "PTS", "W", "SALARY_MILLIONS"
          )

subset <- salaryTwitter100[playerPrf]

ggplot(subset, aes(x = PTS, y = W)) +
  geom_point() +
  xlab("PPG") +
  ylab("WINS")

corrplot(cor(subset), order="AOE")


library(dplyr)
#REGRESSION ON SALARY
salTwitter = read.csv("nba_2017_players_with_salary_wiki_twitter.csv")
head(salTwitter)
sum(is.na(salTwitter)) #Since the players that do not have twitter are notable players
                  # we will replace NA with 0 so as not remove player from analysis
salTwitter$TWITTER_FAVORITE_COUNT[is.na(salTwitter$TWITTER_FAVORITE_COUNT)]
<- 0
salTwitter$TWITTER_RETWEET_COUNT[is.na(salTwitter$TWITTER_RETWEET_COUNT)]
<- 0

na_values <- is.na(salTwitter)
na_counts <- colSums(na_values)
print(na_counts) #X3P has NA values because not 3 pointers were taken for threshold
(low sample)
#check type of each column
salTwitter <- replace(salTwitter, is.na(salTwitter), 0)
# make new df with only num/int columns so remove "Name, Team" and X since just
obs number
sal_num <- salTwitter %>% select_if(~ !is.character(.))
```

```
sal_num <- sal_num[,-1]
str(sal_num)

library(corrplot)
library(ggplot2)
library(GGally)

#ggpairs(sal_num)
corrplot(cor(sal_num), order="AOE", tl.cex = 0.6)

#Regularized Regression
# Grab test and training sets
n = nrow(sal_num)
s = sample(n, n/2)
salTrain = sal_num[s, ]
salTest = sal_num[-s, ]


olsFit = lm(SALARY_MILLIONS ~ ., data=salTrain)
summary(olsFit)
plot(olsFit)


rmseOlsTrain = sqrt(mean(olsFit$residuals^2))
rmseOlsTrain

# Predict on the test set and compute error
olsPred = predict(olsFit, salTest)
rmseOlsTest = sqrt(mean((olsPred - salTest$SALARY_MILLIONS)^2))
rmseOlsTest  # A lot higher, almost double

#LASSO REGRESSION
library(glmnet)
# Separate the X's and Y's as matrices
xTrain = as.matrix(salTrain[, -35])   # Take out "SALARY_MILLIONS”
yTrain = as.matrix(salTrain[, 35])    # Take only "SALARY_MILLIONS”

xTest = as.matrix(salTest[, -35])   # Take out "SALARY_MILLIONS”
yTest = as.matrix(salTest[, 35])    # Take only  "SALARY_MILLIONS”

fitLasso = glmnet(xTrain, yTrain, alpha=1)
plot(fitLasso, xvar="lambda")
fitLasso = cv.glmnet(xTrain, yTrain, alpha=1)
fitLasso
```

```
lassoPred1se = predict(fitLasso, xTest, s="lambda.1se")
lassoPredMin = predict(fitLasso, xTest, s="lambda.min")

rmseLasso1se = sqrt(mean((lassoPred1se - yTest)^2))
rmseLassoMin = sqrt(mean((lassoPredMin - yTest)^2))
rmseLasso1se
rmseLassoMin
# LASSO did a bit better on this set
#plot for MSE
#plot(olsFit)
plot(fitLasso)


#explore relaxed lasso
fitR_Lasso = glmnet(xTrain, yTrain, relax = T)
plot(fitR_Lasso, xvar="lambda")
plot(fitR_Lasso)
fitR_Lasso

fitR_Lasso_cv = cv.glmnet(xTrain, yTrain, relax = T)
fitR_Lasso_cv
plot(fitR_Lasso)
RlassoPred1se = predict(fitRLasso, xTest, s="lambda.1se")
RlassoPredMin = predict(fitRLasso, xTest, s="lambda.min")
```