Executive Summary

The National Basketball Association (NBA) is a men's professional basketball league comprised of 30 teams. Each of these teams have multi-million dollar budgets that they use in order to build a talented team that will hopefully win enough games to bring home the NBA title. To do this, the teams build their rosters based on talent and budget. While team payrolls are large, they are not unlimited and strategic decisions must be made to develop the most talented team while maintaining budget constraints. The assumption of most teams, and players as well, is that the more money a particular player demands, the better the player. This is not always the case, however. There are numerous instances where players have been drastically overpaid and dramatically underperformed.[1-2] Conversely, there have been players that have performed well above their higher-paid counter-parts.[3] Ideally, there would be a mechanism where player salary could be determined by consistent player performance. It is reasonable to assume that NBA front offices do indeed subscribe to this premise, but some player salaries would suggest otherwise. The purpose behind this course project is to use predictive analytics to predict what a player's salary should be by looking at player statistics.

Over the course of this project, our team has developed a program to predict which players are currently being overpaid and which players are underpaid. We used several factors that were commonly found in higher paid players such as age, efficiency rating, minutes played, turnovers, and several others.

We were able to use these factors, in conjunction with several data science techniques, to determine the accuracy of our predictions.

The end result was an output of the top 10 overpaid and underpaid players.

Basketball teams could use what we've built to help manage payrolls and keep player performance tied to player salary. Additionally, players and their agents could benefit from this product when it came time for contract renegotiations. Players could clearly show that based on their performance statistics, they should receive a higher compensation, or coaches could use it to determine the next star player from the next draft. However, if the owners of the basketball teams came across this product, then they might use it to determine who needs a pay cut.

Introduction

With ever increasing team payrolls, owners can usually afford to pay an underachieving player. The real issue is in determining the opportunity cost of this: What could you have gotten for that money instead? How else could you have improved your roster? It should come as no surprise then that teams are looking for alternative methods for determining appropriate pay levels for their athletes. This is where predictive analytics can hopefully shed some needed light on this very expensive problem as laid out below:

- What should an NBA player be paid based on his performance statistics?
- Which players are drastically overpaid?
- Which players are drastically underpaid?

For this project, we will be utilizing the NBA player statistics dataset that was obtained from Kaggle[4]. The website lists the statistics as originating from 2017-2018 team rosters, but upon further verification, the rosters are more in line with 2016-2017 data[5]. The downloaded data contained 88 missing salary values. All but two of these were obtained from HoopsHype.com5 website of player salaries.

The dataset contains 52 different variables. Not all of these are relevant to predicting player salaries and a detailed description of the variables used will be included in the technical approach section below.

Methods

Three distinct methods are used to generate our predictive model. These are broken down into particular phases.

- Phase I: This consists of conducting an exploratory data analysis (EDA) on the NBA dataset. Our dataset contains 52 different variables that could contribute to our predictive model. We realize this is simply too many, so we have developed a process by which we will select the variables of interest. The first step was to look at the summary statistics for each of the variables. The goal of this task was to see if any variables contained missing values, values that were possible outliers, or incorrect values.

  The next phase of our EDA was to look at the histograms for each variable. According to our text, *"Applied Predictive Analytics"*[6], many predictive algorithms assume the model variables follow a normal distribution. There are inherent advantages to using normally distributed variables, so our approach will focus on columns that closely follow this distribution. Additionally, we will use the histograms to help determine if any variable has outlier values.

- Phase II: This portion will focus on feature/variable selection from the 52 variables within the dataset. Our research led us to an article[7] that provided a more definitive method for feature selection that is based on various methods, such as a random forest classifier, a recursive feature elimination, an extra trees classifier, a chi squared analysis, and a Lasso regression. These methods generated DataFrames of the top features and a scoring system was added that would keep track of the highest-ranking features for each method. The overall feature score was then determined that provided the final feature rankings.

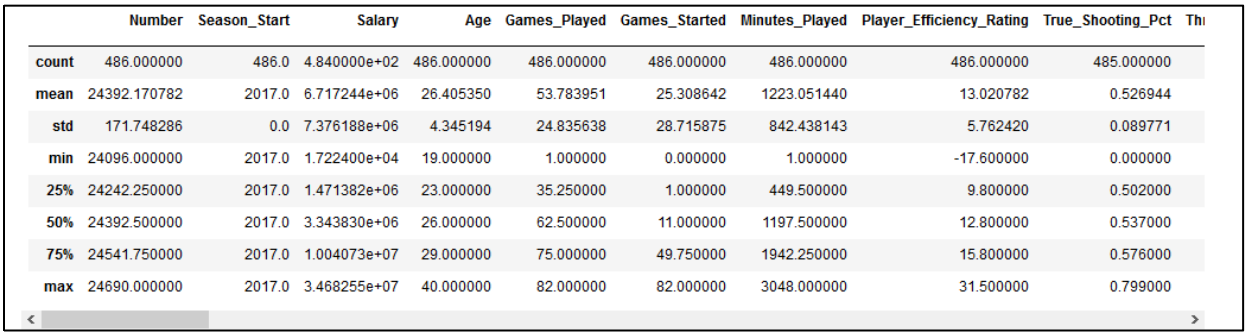
  Some features ended up with the same final score and so a correlation matrix was built to further reduce the features that were considered redundant.

- Phase III: This phase takes the results from Phases I and II and provides the final features that will be used in our initial predictive model build. The model will then be run and the summary of results and a path forward will be discussed.

Results

Phase I: Exploratory Data Analysis (EDA)

Statistical Summary: A sample output of this analysis is shown in Figure 1. We took particular note of the **count** values to help verify missing data, as well as the **max** and **min** values to track incorrect or outlier values. The result of these summary statistics showed that there were indeed some missing values that would require further investigation but the max and min numbers appeared to be within the expected ranges of possible values.



```
             Number  Season_Start        Salary         Age  Games_Played  Games_Started  Minutes_Played  Player_Efficiency_Rating  True_Shooting_Pct  Th
count    486.000000         486.0  4.840000e+02  486.000000    486.000000     486.000000      486.000000                486.000000         485.000000
mean   24392.170782        2017.0  6.717244e+06   26.405350     53.783951      25.308642     1223.051440                 13.020782           0.526944
std      171.748286           0.0  7.376188e+06    4.345194     24.835638      28.715875      842.438143                  5.762420           0.089771
min    24096.000000        2017.0  1.722400e+04   19.000000      1.000000       0.000000        1.000000                -17.600000           0.000000
25%    24242.250000        2017.0  1.471382e+06   23.000000     35.250000       1.000000      449.500000                  9.800000           0.502000
50%    24392.500000        2017.0  3.343830e+06   26.000000     62.500000      11.000000     1197.500000                 12.800000           0.537000
75%    24541.750000        2017.0  1.004073e+07   29.000000     75.000000      49.750000     1942.250000                 15.800000           0.576000
max    24690.000000        2017.0  3.468255e+07   40.000000     82.000000      82.000000     3048.000000                 31.500000           0.799000
```

Figure 1. Sample of summary statistics for NBA dataset.

Histograms: The output from the histogram plots are shown in Figure 2.

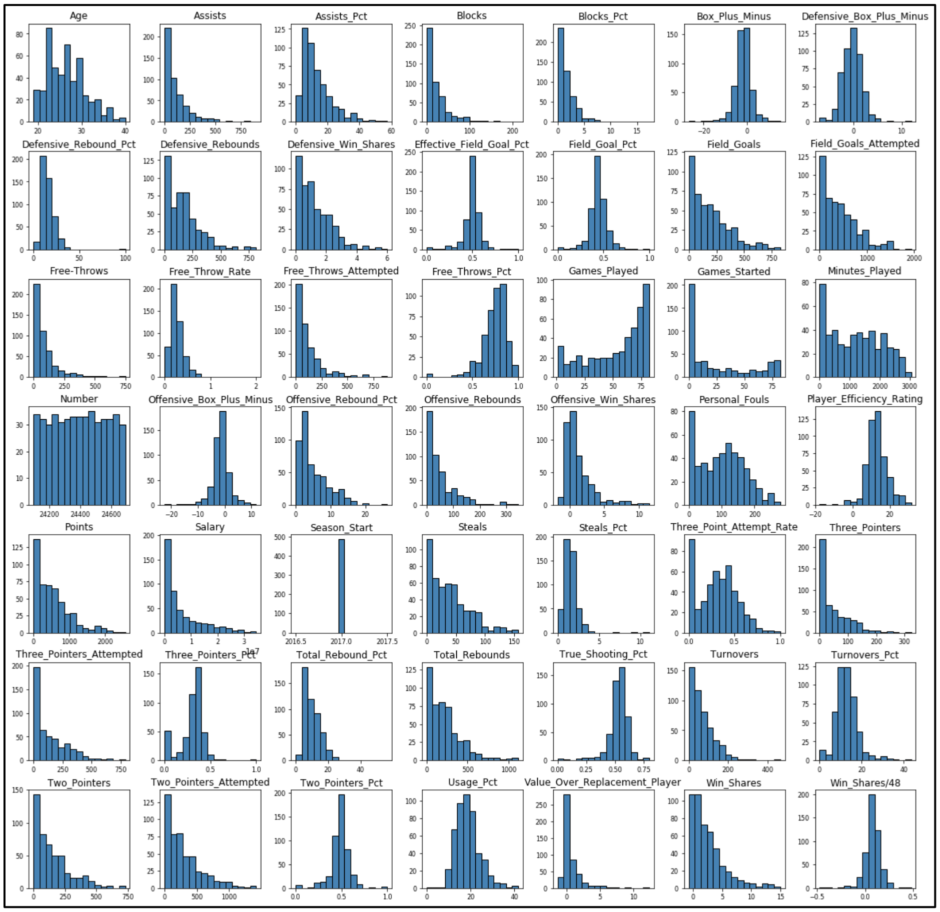


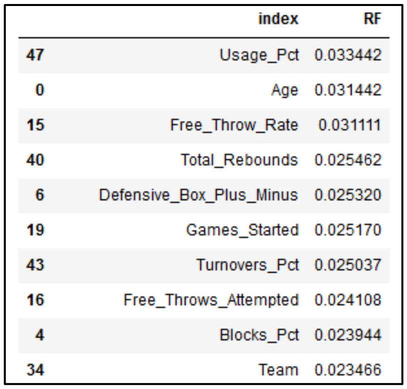
Figure 2. Histogram plots of all NBA dataset variables.

The histogram plots do indeed indicate those variables with normal distributions, such as Player_Efficiency_Rating. The plots also show that there do not appear to be any outlying data points that we need to be concerned about.

While both of these two techniques helped us get a better view of our variable distributions, we still were not comfortable selecting our model variables from these two EDA techniques alone.

Phase II: Feature Selection

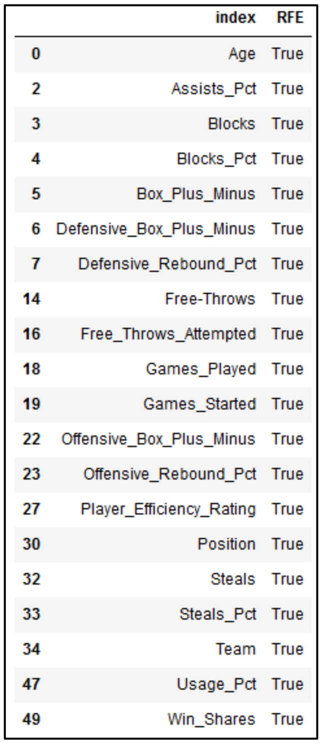
- **Variable Importance: Random Forest Classifier**—this method utilized the RandomForestClassifier from the sklearn library. Random forests are one the most popular machine learning algorithms. They are so successful because they provide in general a good predictive performance, low overfitting, and easy interpretability. This interpretability is given by the fact that it is straightforward to derive the importance of each variable on the tree decision. In other words, it is easy to compute how much each variable is contributing to the decision.[8] Each of our variables were subjected to this algorithm and then ranked on importance to the model. The output of this process is shown in Figure 3. The first column is the DataFrame column number, the **index** column is the name of the dataset variable, and the **RF** column is the Random Forest value. Due to the length of the output, only the first 10 values are shown.



	index	RF
47	Usage_Pct	0.033442
0	Age	0.031442
15	Free_Throw_Rate	0.031111
40	Total_Rebounds	0.025462
6	Defensive_Box_Plus_Minus	0.025320
19	Games_Started	0.025170
43	Turnovers_Pct	0.025037
16	Free_Throws_Attempted	0.024108
4	Blocks_Pct	0.023944
34	Team	0.023466

Figure 3. Output from Random Forest variable importance calculation.

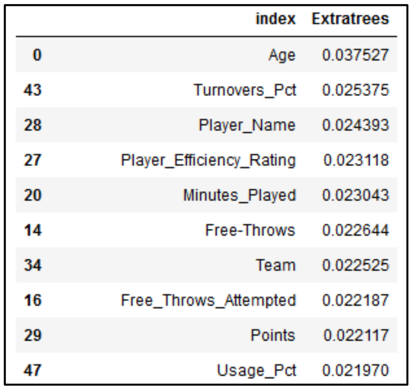
- **Recursive Feature Elimination (RFE)**—this technique is also from the sklearn library. RFE is basically a backward selection of the predictors. This technique begins by building a model on the entire set of predictors and computing an importance score for each predictor. The least important predictor(s) are then removed, the model is re-built, and importance scores are computed again, hence the recursive nature of the process.[9] The output from this process is shown in Figure 4. Like before, the first column is the DataFrame column number, the **index** column is the name of the dataset variable, and the **RFE** column is the Recursive Feature Elimination result. The values of the RFE column are shown as True because they have not been eliminated in the RFE selection process. In other words, these remaining variables are considered important to the overall predictive model. The output below (in alphabetical order) shows all of the variables that returned a True value.



	index	RFE
0	Age	True
2	Assists_Pct	True
3	Blocks	True
4	Blocks_Pct	True
5	Box_Plus_Minus	True
6	Defensive_Box_Plus_Minus	True
7	Defensive_Rebound_Pct	True
14	Free-Throws	True
16	Free_Throws_Attempted	True
18	Games_Played	True
19	Games_Started	True
22	Offensive_Box_Plus_Minus	True
23	Offensive_Rebound_Pct	True
27	Player_Efficiency_Rating	True
30	Position	True
32	Steals	True
33	Steals_Pct	True
34	Team	True
47	Usage_Pct	True
49	Win_Shares	True

Figure 4. Output from RFE calculation.

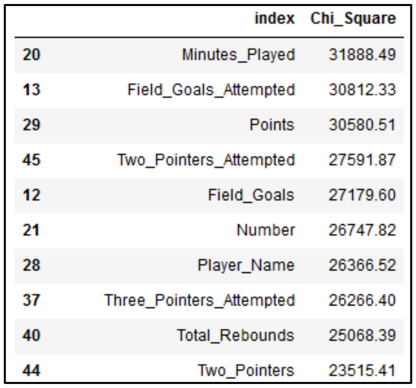
- **Variable Importance: Extra trees Classifier**—this technique uses the ExtraTreesClassifier module from the sklearn library. In concept, the Extra Trees Classifier is very similar to a Random Forest Classifier and only differs from it in the manner of construction of the decision trees in the forest. Each Decision Tree in the Extra Trees Forest is constructed from the original training sample. Then, at each test node, each tree is provided with a random sample of k features from the feature-set. From this, each decision tree must select the best feature to split the data. This random sample of features leads to the creation of multiple de-correlated decision trees.[10] The output from this process is shown in Figure 5. The first column is the DataFrame column number, the **index** column is the name of the dataset variable, and the **Extratrees** column is the algorithm result. Due to the length of the output, only the first 10 values are shown.



	index	Extratrees
0	Age	0.037527
43	Turnovers_Pct	0.025375
28	Player_Name	0.024393
27	Player_Efficiency_Rating	0.023118
20	Minutes_Played	0.023043
14	Free-Throws	0.022644
34	Team	0.022525
16	Free_Throws_Attempted	0.022187
29	Points	0.022117
47	Usage_Pct	0.021970

Figure 5. Output from Extra Trees Classifier.

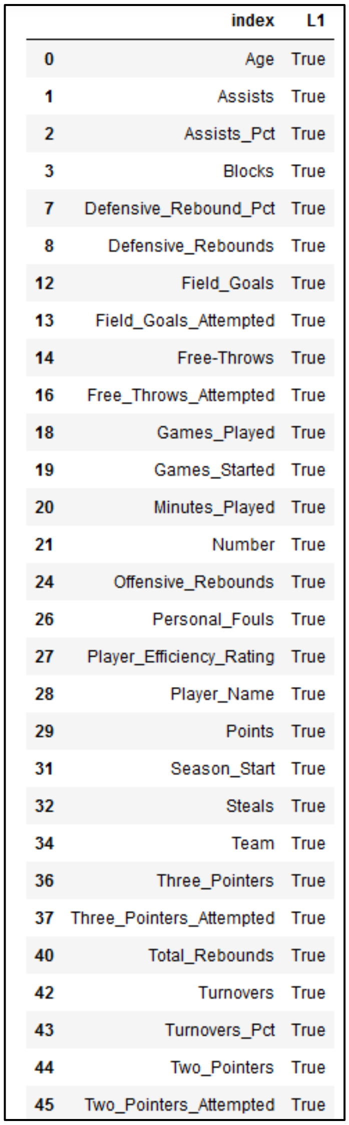
- **Chi Square**—this method is also from the sklearn library. The Chi Square Test is used in statistics to test the independence of two events. In feature selection, the two events are occurrence of the feature and occurrence of the class. In other words, we want to test whether the occurrence of a specific feature and the occurrence of a specific class are independent. When the two events are independent, the observed count is close to the expected count, thus a small chi square score. So a high chi square value indicates that the hypothesis of independence is incorrect. In other words, the higher value of the chi square score, the more likelihood the feature is correlated with the class, thus it should be selected for the model.[11] The output of the chi square test is shown in Figure 6. The first column is the DataFrame column number, the **index** column is the name of the dataset variable, and the **Chi_Square** column is the calculated chi square value. Due to the length of the output, only the first 10 values are shown.



	index	Chi_Square
20	Minutes_Played	31888.49
13	Field_Goals_Attempted	30812.33
29	Points	30580.51
45	Two_Pointers_Attempted	27591.87
12	Field_Goals	27179.60
21	Number	26747.82
28	Player_Name	26366.52
37	Three_Pointers_Attempted	26266.40
40	Total_Rebounds	25068.39
44	Two_Pointers	23515.41

Figure 6. Output from the Chi Square calculation.

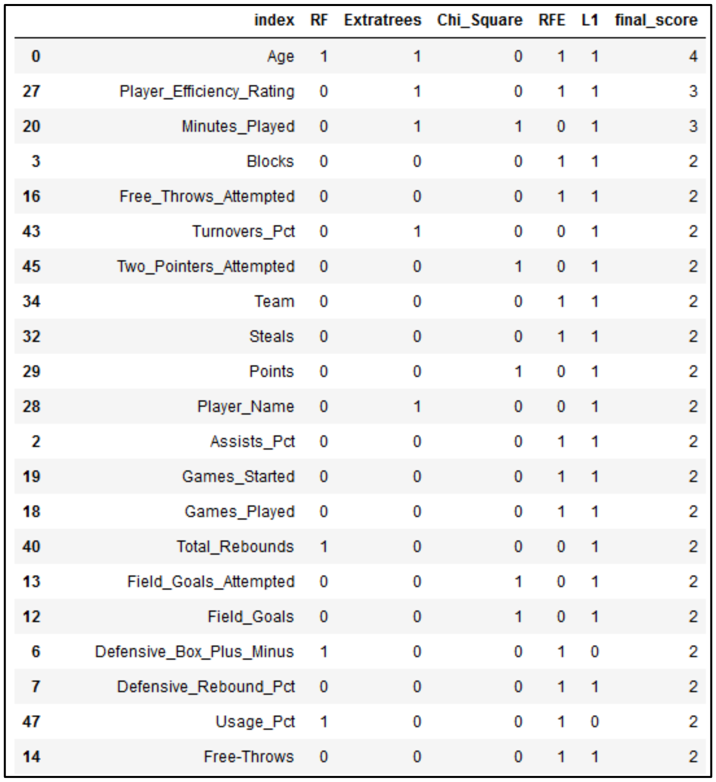
- **Lasso Regression (L1)**—this method was also used from the sklearn library. Regularisation consists in adding a penalty to the different parameters of the machine learning model to reduce the freedom of the model and in other words to avoid overfitting. In linear model regularisation, the penalty is applied over the coefficients that multiply each of the predictors. From the different types of regularisation, Lasso or L1 has the property that is able to shrink some of the coefficients to zero. Therefore, that feature can be removed from the model.[12] The output from our Lasso Regression is shown in Figure 7. The first column is the DataFrame column number, the **index** column is the name of the dataset variable, and the **L1** column is the Lasso Regression result. Like with the RFE output, the values of the L1 column are shown as True because they have not been eliminated in the Lasso Regression selection process. In other words, these remaining variables are considered important to the overall predictive model. The output below (in alphabetical order) shows all of the variables that returned a True value.



	index	L1
0	Age	True
1	Assists	True
2	Assists_Pct	True
3	Blocks	True
7	Defensive_Rebound_Pct	True
8	Defensive_Rebounds	True
12	Field_Goals	True
13	Field_Goals_Attempted	True
14	Free-Throws	True
16	Free_Throws_Attempted	True
18	Games_Played	True
19	Games_Started	True
20	Minutes_Played	True
21	Number	True
24	Offensive_Rebounds	True
26	Personal_Fouls	True
27	Player_Efficiency_Rating	True
28	Player_Name	True
29	Points	True
31	Season_Start	True
32	Steals	True
34	Team	True
36	Three_Pointers	True
37	Three_Pointers_Attempted	True
40	Total_Rebounds	True
42	Turnovers	True
43	Turnovers_Pct	True
44	Two_Pointers	True
45	Two_Pointers_Attempted	True

Figure 7. Output from Lasso Regression (L1) calculation.

- **Scoring Table of Final Results**—a scoring table was constructed that tallied all of the different feature selection methods. From this table, we were able to determine which variables should be included in our model. The scoring table, listing only values with a final_score above 2, is shown in Figure 8.



	index	RF	Extratrees	Chi_Square	RFE	L1	final_score
0	Age	1	1	0	1	1	4
27	Player_Efficiency_Rating	0	1	0	1	1	3
20	Minutes_Played	0	1	1	0	1	3
3	Blocks	0	0	0	1	1	2
16	Free_Throws_Attempted	0	0	0	1	1	2
43	Turnovers_Pct	0	1	0	0	1	2
45	Two_Pointers_Attempted	0	0	1	0	1	2
34	Team	0	0	0	1	1	2
32	Steals	0	0	0	1	1	2
29	Points	0	0	1	0	1	2
28	Player_Name	0	1	0	0	1	2
2	Assists_Pct	0	0	0	1	1	2
19	Games_Started	0	0	0	1	1	2
18	Games_Played	0	0	0	1	1	2
40	Total_Rebounds	1	0	0	0	1	2
13	Field_Goals_Attempted	0	0	1	0	1	2
12	Field_Goals	0	0	1	0	1	2
6	Defensive_Box_Plus_Minus	1	0	0	1	0	2
7	Defensive_Rebound_Pct	0	0	0	1	1	2
47	Usage_Pct	1	0	0	1	0	2
14	Free-Throws	0	0	0	1	1	2

Figure 8. Scoring table of all feature selection results.

We are obviously going to include Age and Player_Efficiency_Rating since they scored the highest. After that, however, we have many other variables that all returned a sum score of 2. We eliminated some of these features by looking at the correlation matrix.

- **Correlation Matrix**— once the features were scored a correlation matrix was used to test the features correlation. The table in Figure 9 is a heatmap showing the correlation of the features scoring at least a 2.

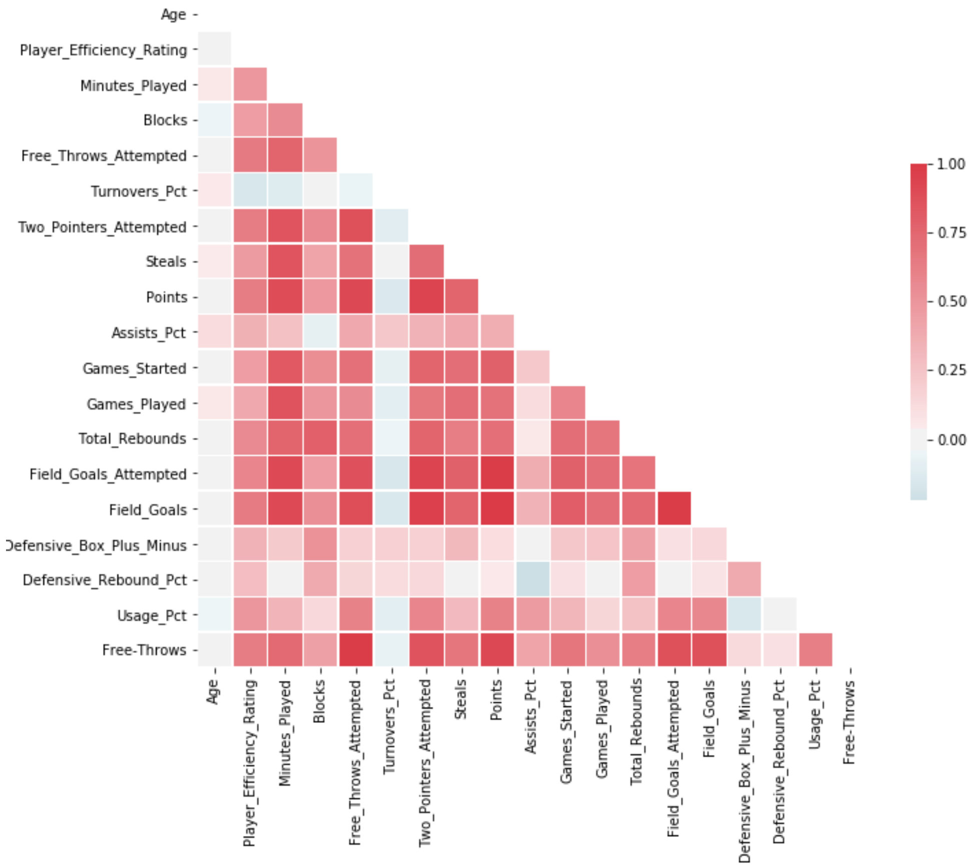


Figure 9. Output from initial Correlation Heatmap

It is clear in this heatmap that there is quite a bit of correlation between features. This isn't surprising because a good basketball player is good at everything and a bad one is bad at everything. These are all important parts of the game so if someone it great at assists they are also likely great at field goals. That being said, we still want to remove as much unnecessary correlation as we can. We first removed all features marking "attempts". They were too highly correlated with successes, so they didn't add much value. The other feature we removed was Minutes Played. This scored highly in our L1 test but it is very highly correlated with most features. That makes sense because the more time you play means the more chances you have at blocking, rebounding, taking shots, etc. Removing these features gave us a heatmap with the 14 features in Figure 10.

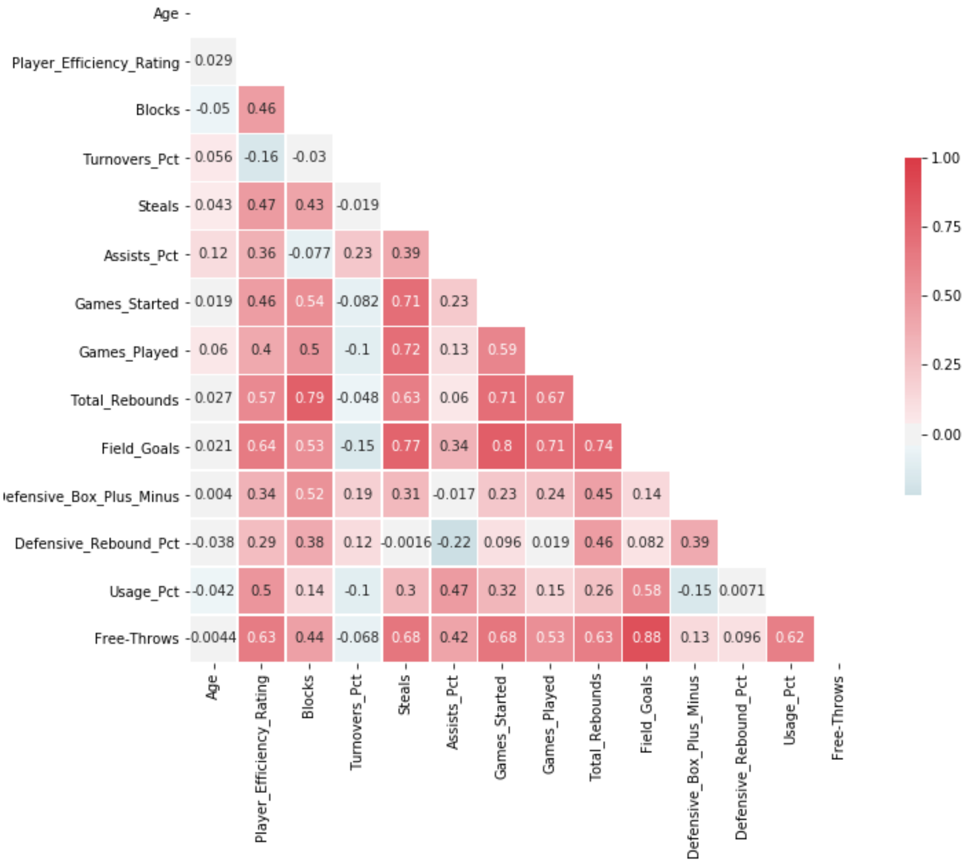


Figure 10. Output from final Correlation Heatmap

Removing those features has removed quite a few of the highly correlated points. There is still some high correlation that we have left in for now. For example, Field Goals and Free Throws are highly correlated but field goals do not include free throws so there isn't that direct relationship like we had in other features.

Phase III: Feature Finalization and Model Build

- **Feature Finalization**—a features summary is shown below in Table 1. This table summarizes each feature selection method and was used determine the final features that were used for the preliminary model run in Phase III.

Table 1. Feature Selection Summary

Feature	Summary Statistics			Histograms	Correlation Matrix	Feature Selection
	Complete Data?	Reasonable Max Value?	Reasonable Min Value?	Normal Distribution?	Uncorrelated?	Final Score
Age	Yes	Yes	Yes	No	Yes	4
Player_Efficiency_Rating	Yes	Yes	Yes	Yes	Yes	3
Minutes_Played	Yes	Yes	Yes	No	No	3
Blocks	Yes	Yes	Yes	No	Yes	2
Free_Throws_Attempted	Yes	Yes	Yes	No	No	2
Turnovers_Pct	No	Yes	Yes	Yes	Yes	2
Two_Pointers_Attempted	Yes	Yes	Yes	No	No	2
Team	Ignored. Not a measurable player metric.					2
Steals	Yes	Yes	Yes	No	Yes	2
Points	Yes	Yes	Yes	No	No	2
Player_Name	Ignored. Not a measurable player metric.					2
Assists_Pct	Yes	Yes	Yes	No	Yes	2
Games_Started	Yes	Yes	Yes	No	Yes	2
Games_Played	Yes	Yes	Yes	No	Yes	2
Total_Rebounds	Yes	Yes	Yes	No	Yes	2
Field_Goals_Attempted	Yes	Yes	Yes	No	No	2
Field_Goals	Yes	Yes	Yes	No	Yes	2
Defensive_Box_Plus_Minus	Yes	Yes	Yes	Yes	Yes	2
Defensive_Rebound_Pct	Yes	No	Yes	Yes	Yes	2
Usage_Pct	Yes	Yes	Yes	No	Yes	2
Free-Throws	Yes	Yes	Yes	No	Yes	2

Based on the table summaries, and on previous discussions in this paper, we have selected 14 variables to be used in our initial model. They are: Age, Player Efficiency Rating, Blocks, Turnover Pct, Steals, Assists Pct, Games Started, Games Played, Total Rebounds, Field Goals, Defensive Box Plus Minus, Defensive Rebound Pct, Usage Pct, and Free Throws.

- **Model Build**—we used an iterative modeling approach to determine the best model for our predictions. We ended up using five different models. We divided the data into a test and training subset before each model fit. We used a 70/30 test to train ratio for each model run. The first four models[13] were run using the sklearn Python package. The fifth model[14] was run using the xgboost Python package. A description of each model considered is discussed below:
  - **Model 1: Ordinary Least Squares (OLS)**—our reason for selecting this model is that it is one of the most commonly used models and would serve as a good baseline with which we could compare the subsequent models. Table 2 shows the results from this model.

Table 2. OLS Model Fit Results

Model	Train RMSE	Train R^2	Test RMSE	Test R^2
OLS	4398500	0.63	5024129	0.57

    - The R^2 values for the models indicate that we are able to account for about 60% of the salary prediction variability with the features we selected.
    - The root mean square error (RMSE) values will serve as a reference point for the other linear models we are going to use.

  - **Model 2: Ridge Regression (RR)**—the ridge regression model is an extension of linear regression where the loss function is modified to minimize the complexity of the model. Ridge regression includes an α variable that we can adjust to help improve the fit. The higher the α value, the more likely our model is going to suffer from under-fitting. A lower α value can lead to under-fitting. Table 3 shows the results from the RR model for two different α values.

Table 3. RR Model Fit Results

Model	Train RMSE	Train R^2	Test RMSE	Test R^2
RR, α=0.01	4398504	0.63	5024400	0.57
RR, α=10	4608385	0.59	5203120	0.54

    - The R^2 values for the models indicate that we are able to account for about 60% of the salary prediction variability with the features we selected.
    - The RMSE values for the RR method are slightly higher than the OLS values. This indicates that the OLS predictions are more accurate.

- Increasing the α value decreased the model accuracy.

- **Model 3: Lasso Regression (LR)**—Lasso regression is another modification of linear regression modeling. In Lasso, the loss function is modified to minimize the complexity of the model by limiting the sum of the absolute values of the model coefficients. We will select different α values to tune the model for the best fit. Table 4 shows the results from the LR model for two different α values.

Table 4. LR Model Fit Results

Model	Train RMSE	Train R^2	Test RMSE	Test R^2
LR, α=0.01	4398500	0.63	5024129	0.57
LR, α=100	4398502	0.63	5023978	0.57

- The R^2 values for the models indicate that we are able to account for about 60% of the salary prediction variability with the features selected.
- The RMSE values for the LR method are essentially identical to the OLS values. This indicates that the OLS and Lasso predictions are basically the same.
- Increasing the α value had little impact on the model accuracy.

- **Model 4: ElasticNet Regression (ENR)**—ElasticNet regression combines the properties of both the Ridge and Lasso regression methods. ENR also utilized an α term to help improve fit. Table 5 shows the results from the ENR model for two different α values.

Table 5. ENR Model Fit Results

Model	Train RMSE	Train R^2	Test RMSE	Test R^2
ENR, α=0.01	4432638	0.62	5070003	0.56
ENR, α=10	7010747	0.06	7488845	0.05

- The R^2 values for the models were still around 60%, but were slightly lower that what we saw with the other models.
- The RMSE values for the ENR method were the highest of all the models. This indicates the least accurate prediction model. This is likely due to the fact that we removed features that were highly correlated with one another.
- Increasing the α value had a negative impact on our prediction accuracy.

- **Model 5: Extreme Gradient Boosting Regression (XGBR)**—XGBR is an ensemble modeling technique where new models are added to correct the errors made by existing models. This method uses the gradient boosting decision tree algorithm. Table 6 shows the results from the XGBR model.

Table 6. XGBR Model Fit Results

Model	Train RMSE	Train R^2	Test RMSE	Test R^2
XGBR	1818766	0.92	4531190	0.65

  - The R^2 values for the XGBR method were the best of all models evaluated. Our training set R^2 had a value of 92%, which is excellent. The results diminished to only 65% when applied to the test set, but this about 10% better than our other models. The reason for the lower R^2 value for the test set is likely due to the lower number of data points. Remember that the test set contained only one-third of the original data.
  - The RMSE values for the XGBR method were also the lowest of all models evaluated. This indicates the most accurate prediction model.
  - Because the XGBR model provided the best overall fit, this is the model we selected as our final model.
  - Figure 11 shows the actual versus predicted salaries for the XGBR model test and training sets. The solid red line shows the best fit for each data plot. As one can see, there is good clustering near the best fit line, which indicates a good model result between the actual and predicted salaries. The test set fit is not as good as the training set and this is again due to the reduced number of data points in the test fit dataset. Note: all plots in this report were made with R. Other R-generated plots are included in the "NBA Features Selection & Models.ipynb" file that accompanies this assignment.

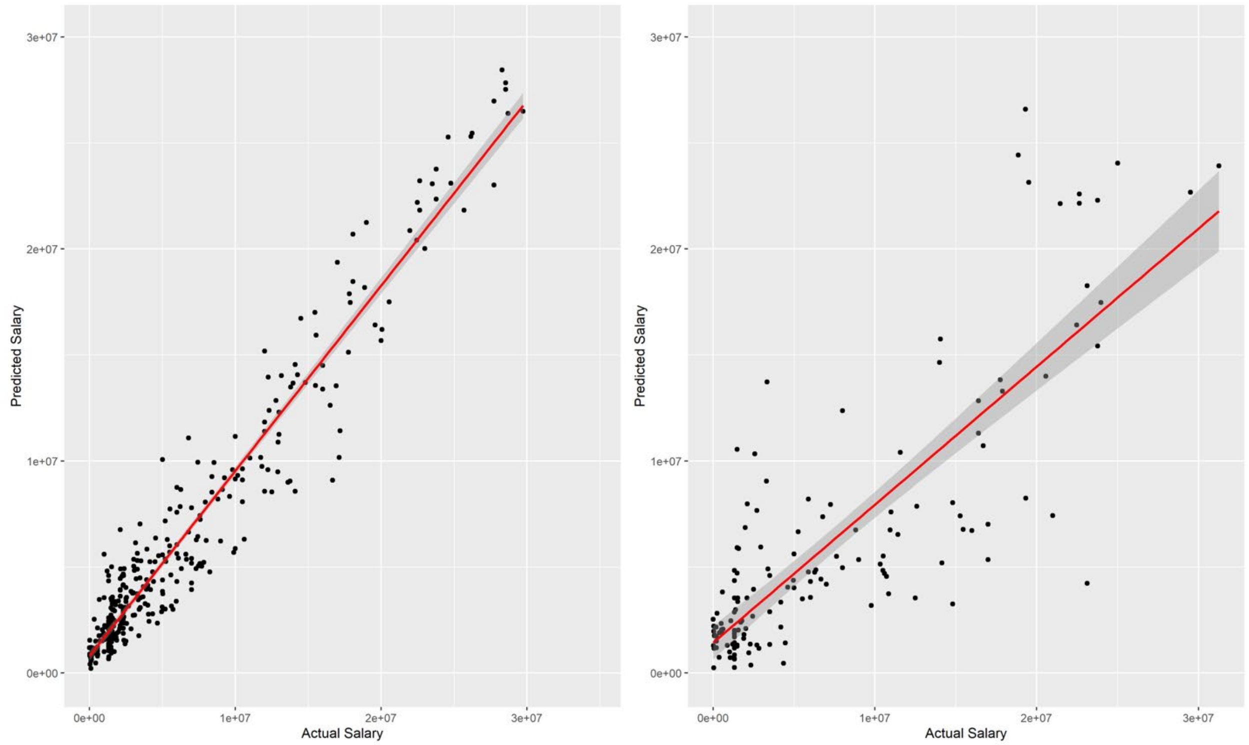


Figure 11. Comparison of Model Output: Training Set on Left and Test Set on Right

Discussion/Conclusion

In summary, exploratory data analysis was performed on the variables to help identify which ones might not be good predictors of salary. Plotting the variables as histograms helped to visualize each data point and determine which ones had a normal distribution. However, this still wasn't enough evidence to make a solid decision on which variables to use in the analysis. Several feature selection techniques were used to help with this decision, including random forest classifier, RFE, ExtraTreesClassifier, Chi Square, Lasso regression, and a correlation matrix. These techniques were helpful in feature selection and several variables were removed from the analysis if their correlation value was relatively small when compared to other variables. In the end, these efforts led to a selection of 14 features that will be used to analyze player salary.

Our initial linear model results were good, but we wanted to include other models to ensure that we were making the most accurate predictions possible. Of the five models we considered, the Extreme Gradient Boosting Regression (XGBR) proved to me the most accurate. We obtained an R^2 value of 0.92 for our training subset of data. This was about a 30% improvement over the over four models we used. Additionally, we reran the XGBR model on the complete dataset (without separating into test and training subsets) in order to correlate player names with salary predictions so we could see which athletes were overpaid and which were underpaid. When we ran this model instance, we obtained an R^2 value of 0.93 (see attached "Predictions with Player Names.ipynb" file).

The final goal of this project was to see what players are currently being overpaid as well as those players that are underpaid. Figure 12 shows the top 10 overpaid players as predicted by our model. As you can see, Chandler Parsons is the most overpaid player by over $8M.

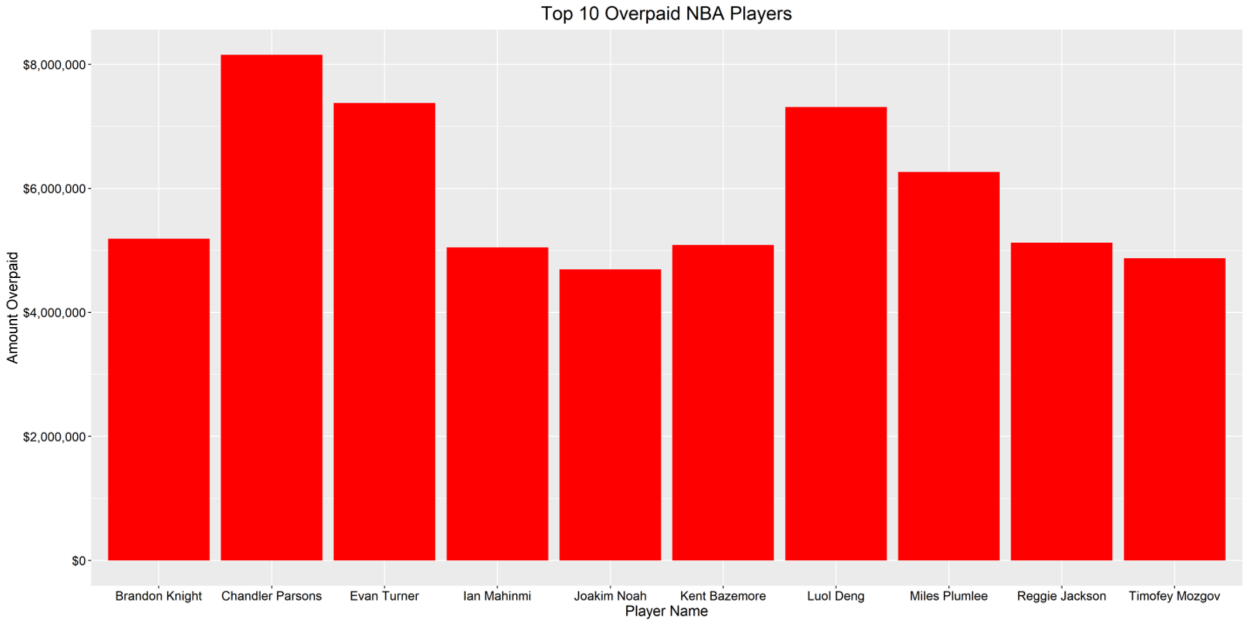


Figure 12. Top 10 Overpaid Players in the NBA

Figure 13 shows the top 10 underpaid players as predicted by our model. As you can see, Marcus Morris is the most underpaid player by over $5M.

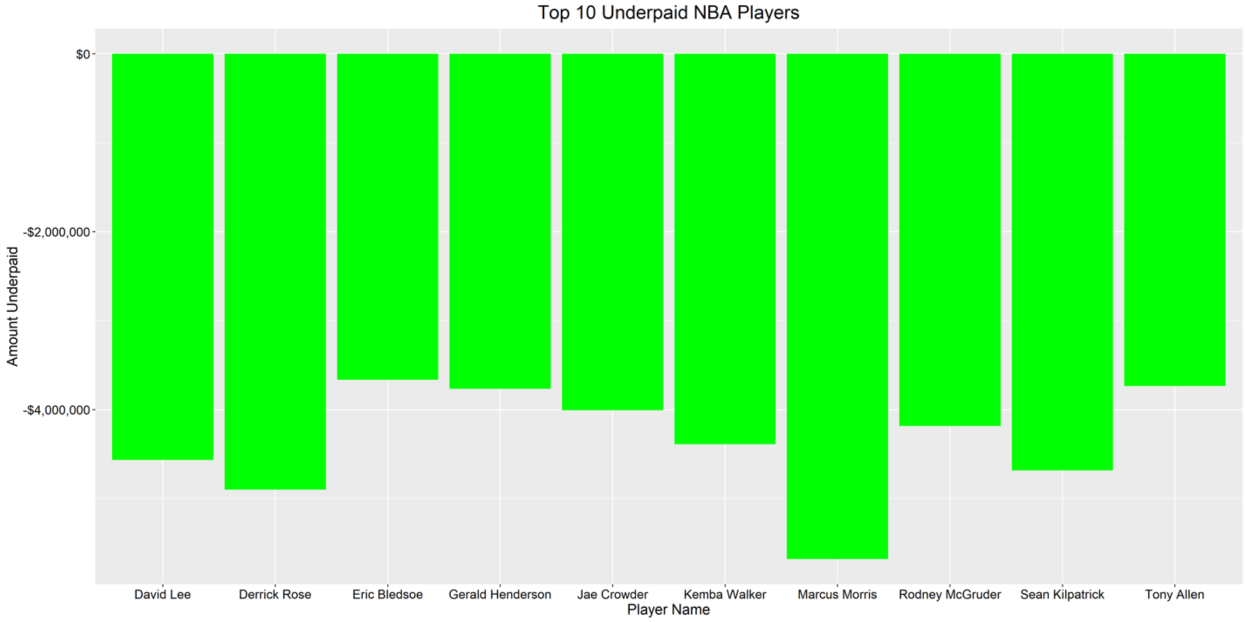


Figure 13. Top 10 Underpaid Players in the NBA

Challenges

A few challenges presented themselves over the course of this project. For starters, several of our feature selection methods were giving different responses as to which features were strongly correlated. In addition, we had lots of high correlation across many features, so additional correlation testing was required. Even still, we had a large amount of features, so we manually narrowed it down further.

Also, many of the models we used were new to us. Sometimes they were applied wrong, or they were not suited to the type of data we had. We attempted to use many more models than the ones mentioned here, including some Random Forests that performed very poorly, but in the end we believe we chose the best models for the job.

Future Work / Scalability

There are several ways this model can be improved for future use. First off, scalability testing would be required, so see if the model can accurately predict salaries from new data. Automatic processes can be put into place to blend the data without manual work. This includes data collection as well, by using methods such as web scraping to automatically acquire the data, and automatic correlation testing to determine the best variables from new data as well. Finally, the model could be improved to hook the players' names back into the data, which could generate an output of expected salary and actual salary.

Acknowledgments

We are indebted to the communities behind the multiple open-source software packages on which we depend.

References

[1] Gleeson, S. (2019, June 19). NBA draft: Ranking the five biggest busts as the No. 1 pick. Retrieved from https://www.usatoday.com/story/sports/nba/draft/2019/06/19/nba-draft-ranking-five-biggest-busts-no-1-pick/1488407001/.

[2] Davis, S. (2019, June 19). WHERE ARE THEY NOW? The biggest NBA Draft busts of all time. Retrieved from https://www.businessinsider.com/nba-draft-busts-2015-6.

[3] Ye, D. (2019, July 4). Top 10 Most Underrated NBA Players of the 2018-19 Season. Retrieved from https://howtheyplay.com/team-sports/Top-10-Most-Underrated-NBA-Players-of-the-2018-19-Season.

[4] Casalan, A. (2018, October 20). NBA player stats 2017-18. Retrieved from https://www.kaggle.com/acasalan/nba-player-stats-201718.

[5] These are the 2016/17 salaries of all NBA teams. (n.d.). Retrieved from https://hoopshype.com/salaries/2016-2017/.

[6] Abbott, D. (2014). *Applied predictive analytics: principles and techniques for the professional data analyst*. Indianapolis, IN: Wiley.

[7] Krishnan, S. (2019, December 20). Variable Selection using Python - Vote based approach. Retrieved from https://medium.com/@sundarstyles89/variable-selection-using-python-vote-based-approach-faa42da960f0.

[8] Dubey, A. (2018, December 15). Feature Selection Using Random forest. Retrieved from https://towardsdatascience.com/feature-selection-using-random-forest-26d7b747597f.

[9] Rade, D. (2019, September 2). Feature Selection in Python - Recursive Feature Elimination. Retrieved from https://towardsdatascience.com/feature-selection-in-python-recursive-feature-elimination-19f1c39b8d15.

[10] Gupta, A. (2019, July 26). ML: Extra Tree Classifier for Feature Selection. Retrieved from https://www.geeksforgeeks.org/ml-extra-tree-classifier-for-feature-selection/.

[11] Chi Square test for feature selection. (2018, July 5). Retrieved from http://www.learn4master.com/machine-learning/chi-square-test-for-feature-selection.

[12] Dubey, A. (2019, February 4). Feature Selection Using Regularisation. Retrieved from https://towardsdatascience.com/feature-selection-using-regularisation-a3678b71e499.

[13] Singh, D. (2019, May 17). Deepika Singh. Retrieved from https://www.pluralsight.com/guides/linear-lasso-ridge-regression-scikit-learn

[14] Brownlee, J. (2019, August 21). A Gentle Introduction to XGBoost for Applied Machine Learning. Retrieved from https://machinelearningmastery.com/gentle-introduction-xgboost-applied-machine-learning/