Capstone Project 1: Final Report

NBA Players' Salary Prediction

Terrence Turner

June 4, 2020

Overview

Whether you're Sam Presti, Danny Ainge, or Rob Pelinka, your job as general manager (GM) in the NBA to form the best team possible to help win the organization a championship. But the one problem all GM's face is how much is too much for a player. Of course, there are superstar players that deserve Max contracts and players on the decline that deserve the veteran minimum, but what about those in between. Maybe a player that shows promise but may not be a star yet. How would you pay him?

Problem Statement

This project studies NBA players' stats and their contracts to see how well it can predict the future salary of an NBA player.

Dataset

I initially started with three datasets that included NBA players' stats (1950-2019), the current NBA rookies' college stats (2018-19), and NBA players' contracts. My original plan was to use the rookies' college stats from their last season since they did not have any rookie stats because the season was currently going on. But when COVID-19 cut the NBA season short, I went ahead and chose to use their rookie stats and rest of the players' 2020 season stats. The datasets I am using are listed below:

1. The players' stats dataset was downloaded from Kaggle: https://www.kaggle.com/lancharro5/seasons-stats-50-19
2. The players' current salaries dataset was scraped from the website https://www.spotrac.com/nba/contracts/
3. The players' 2020 stats were scraped from https://www.basketball-reference.com/leagues/NBA_2020_per_game.html

Data Wrangling

What kind of cleaning steps did I perform?

For the contracts dataset, I dropped any null columns. Split the player column into Player, Year, and Contract Ends. I removed all parentheses, commas, percent signs, and dollar signs from the table. When splitting the player column, one of the problems I had was with suffixes in names. I was able to split the column into first name and last name. Players' Last Name column had the year of the contract as a value as well. So I split the Last name column to where I ended up with Player (first and last name value), Year (contract begins), Contract Ends columns. I got rid of accents and umlauts in player names. I made the Year column numeric, so I can increase contract Year column value by 1 to match stats year season. For example, the 2017-18 season is labeled 2018 in stats but 2017 in contracts. Also, I got rid of the accents and umlauts in the players' names. There were only three players labeled as G (guard) rather than PG (point guard) or SG (shooting guard). This is most likely because they may be considered combo guards, so I chose to make them all SG. I have 405 players in the contracts data frame.

For the stats dataset that consists of all NBA players from 1950 - 2019, I filtered to years 2014 - 2019 because the earliest contract signed was 2015. I wanted to have stats available at least a year ahead of the current season. I rename columns to show stats per game rather than in total and choose columns names that were easier to understand. I dropped total columns and reordered columns to match the 2020 stats columns into new dataFrame new_stats.

For NBA 2019-20 season (twenty_stats), there was extra value in the name column, so I split the column and dropped the extra value, and dropped the old player and rank column. Added a Year column and replaced empty values with most frequent. I concatenated new_stats and twenty_stats into df_stats. Then merged contracts and df_stats on Player and Year and named DataFrame df_final.

How did I deal with missing values, if any?

There were missing values in twenty_stats, and the plan is to fill those values with SimpleImputer. When I merged the new_stats with contracts, I had 14 players with missing values. I believe most of those players that didn't merge may have just signed a contract before the 2020 season but did not play this year because of injury, i.e., Kevin Durant and Klay Thompson.

Were there outliers, and how did I handle them?

I used IQR to find any outliers in my data. I did find some, but it's not because of data that was entered wrong. It is possible that players with a high usage rate can have a high turnover rate, or a player can have an abnormally great year in scoring. I don't think this will negatively affect the model.

Exploratory Data Analysis

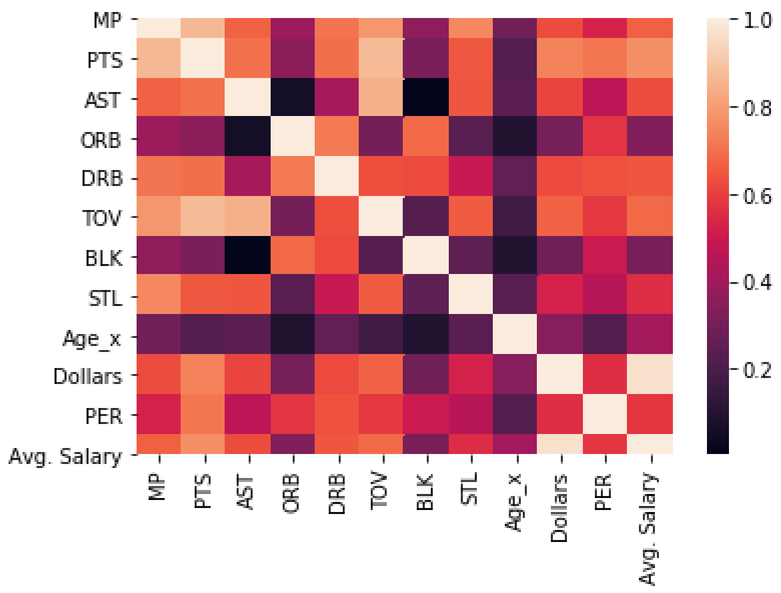
Jupyter notebooks for exploratory data analysis can be found here:

- Descriptive Analysis - https://github.com/terrenceturner/NBA-Salary-Prediction/blob/master/Data%20Storytelling.ipynb
- Statistical Analysis - https://github.com/terrenceturner/NBA-Salary-Prediction/blob/master/Statistical%20Data%20Analysis.ipynb

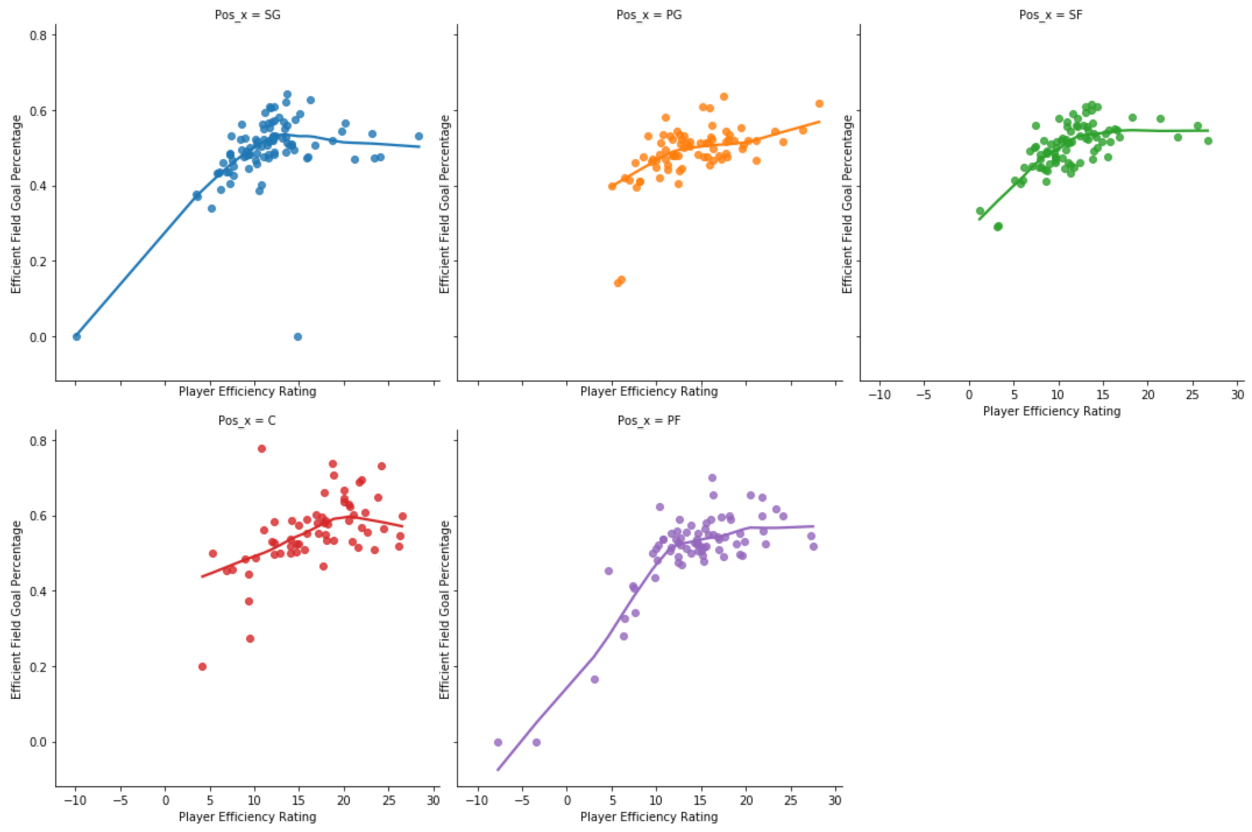
Descriptive Analysis

NBA players' stats is what validates them as a superstar depending on how well their stats are. I sorted through the stats to find the top ten players in points, player efficiency rating (PER), minutes played, and turnovers. Between those four categories, James Harden was the only player that was top three in all four of those categories. Harden is first in points, per, and turnovers. He's also top 3 in minutes played. This means James Harden mostly likely possesses the ball a lot more than other players.

The new advance statistic PER is a rating of a player's per-minute productivity. It is considered by the NBA analytics team an important stat when determining a player's worth. This was surprising when I used a heatmap and discovered that the average salary and PER were not strongly correlated. The average salary and points were strongly correlated which make sense when you think about what tickets sell for.



I also looked at what position is more prone to turnovers. Point guards are most likely to have more turnovers since they usually have the ball in their hands most of the time. Centers and power forwards came one in two with positions who average the most blocks. This is probably obvious since they are the most likely position in the paint defensively. When comparing efficient field goal percentage (eFG%) and PER proved that efficient scorers do not necessarily mean the player is more efficient due to that there is more to basketball than just scoring. The position that has more efficient players are the shooting guards.

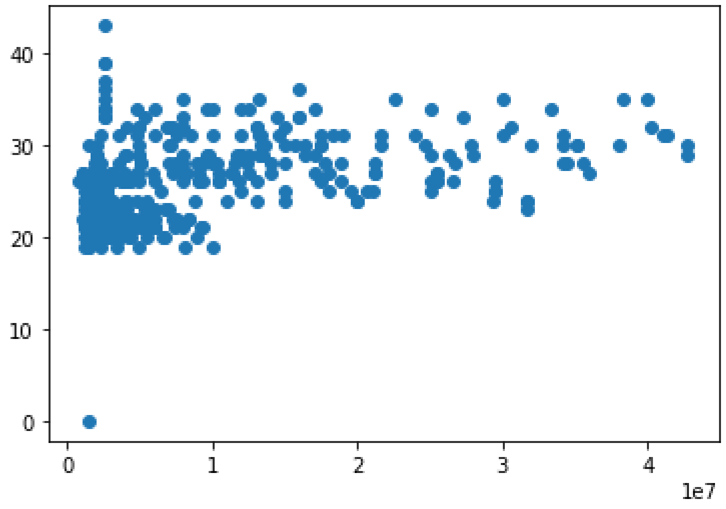


Statistical Analysis

Being an NBA player is a very lucrative job whether you're the NBA's best player or an NBA vet who's riding the bench. Based on my data the average NBA salary a year is $8,672,969.57 with $42,782,880.0 being the highest. With this project, I'm trying to predict NBA players' salaries based on their stats. What is considered to be one of the most important stats is Player Efficiency Rating (PER). Based on my data analysis using a heat map average salary (Avg. Salary) and PER are not really closely related as we would think, but Avg. Salary and points per game (PTS) are closely related. This could mean that although playing efficiently is important, scoring points is what puts fans in the seats, which puts money in the owner's pockets, which would put money in the player pockets. When I compared the top ten players in salary and top ten players in efficiency rating only James Harden and Steph Curry were in the top ten for both categories, with Harden and Curry being 1 and 2 in PER.

When it comes to Avg. Salary or even total Dollar amount there were not any strong correlations. Other stats like assist (AST) and turnovers (TOV) were strongly correlated. This is mostly due to the fast-paced nature of today's game. You push the ball down the court in hopes of scoring faster. The plan is to catch the defense off guard but sometimes the connection between the two offensive players is not there and this leads to a turnover. High risk, high reward. Minutes played (MP) and PTS were strongly correlated. I think this pretty obvious the more playing time a player gets the more opportunity the player gets to score.

I performed some statistical analysis to look at the overall distribution of Avg. Salary. I looked for players with lower and higher than the average salary. Based on my data there were zero players who earn below average salary and only 6 players with higher than the average salary. I also looked at the correlation between Avg. Salary and age using covariance, Pearson's, and Spearman's correlation. Covariance shows a negative trend and Pearson's and Spearman's correlations showed .408 and .501 respectively. These show that the Avg. Salary and Age are poorly correlated.



In-Depth Analysis

Jupyter notebook for in-depth analysis can be found here:

- In-Depth Analysis - https://github.com/terrenceturner/NBA-Salary-Prediction/blob/master/In-Depth%20Data%20Analysis.ipynb

Data Preparation for Building Machine Learning Models

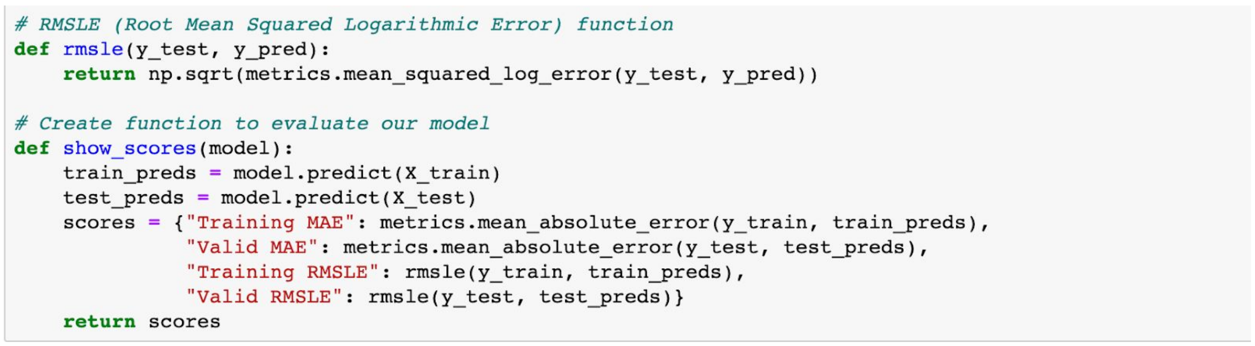
To prepare for the machine learning models I had to check to make sure all of the data was numerical and there were no missing values. Otherwise, the machine learning models will not work. Therefore I dropped categorical columns but also dropped Dollars and Guaranteed column. These columns would cause data leakage when predicting the average salary. It would be easy for the model to predict the average salary based on Dollars (total dollar amount for salary) and Guaranteed (total guaranteed money for the contract). To fill missing values I used SimpleImputer from sci-kit learn. I set up a pipeline with steps that would impute missing value with the mean, StandardScaler, and RandomForestRegressor. Then I split the data into train and test sets with an 80/20 split. 80% of data is for training the model and 20% is used for testing.

Fit Machine Learning Model

I used RandomForestRegressor and GridSearchCV for hyper tuning the model. I build a param grid with the parameters n_estimators, max_depth, min_samples_split, min_samples_leaf. I used the pipeline I created earlier as my grid search estimator and 5-fold cross-validation. The 5-fold cross-validation is when the data set is split into 5 folds. In the first iteration, the first fold is used to test the model and the rest are used to train the model. In the second iteration, 2nd fold is used as the testing set while the rest serve as the training set. This process is repeated until each fold of the 5 folds has been used as the testing set.

Build an Evaluation Function

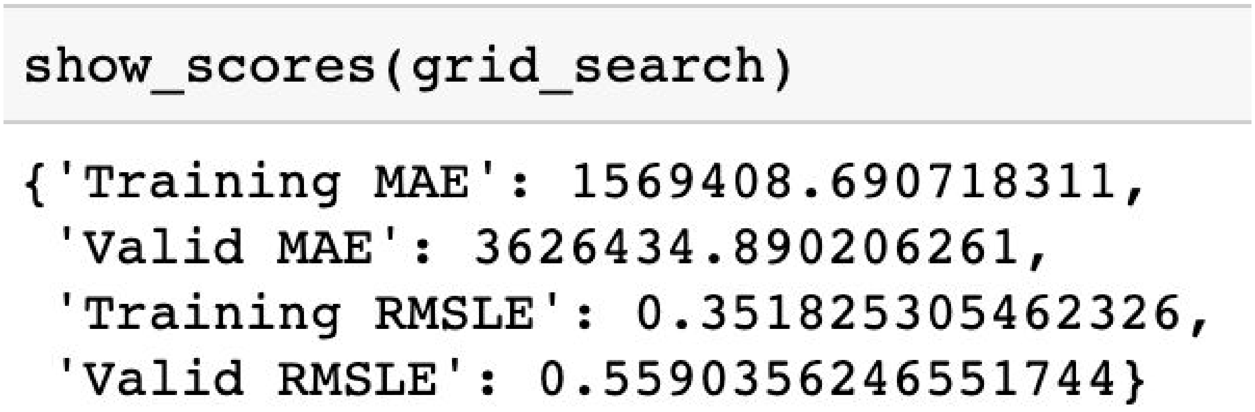
I used RMSLE to evaluate my model. The purpose of RMSLE is to find out the difference between the values predicted by the machine learning model and the actual values. So, I created an RMSLE function where I took Scikit-learn's mean_squared_log_error (MSLE). I also calculated the MAE. Then I created a function to evaluate the model.



```
# RMSLE (Root Mean Squared Logarithmic Error) function
def rmsle(y_test, y_pred):
    return np.sqrt(metrics.mean_squared_log_error(y_test, y_pred))

# Create function to evaluate our model
def show_scores(model):
    train_preds = model.predict(X_train)
    test_preds = model.predict(X_test)
    scores = {"Training MAE": metrics.mean_absolute_error(y_train, train_preds),
              "Valid MAE": metrics.mean_absolute_error(y_test, test_preds),
              "Training RMSLE": rmsle(y_train, train_preds),
              "Valid RMSLE": rmsle(y_test, test_preds)}
    return scores
```

It showed scores on MAE training and test sets, and RMSLE training and test sets. MAE gives a better indication of how far off each of your model's predictions are on average. RMSLE can be seen as a relative error between the predicted and the actual values.



```
show_scores(grid_search)
```

```
{'Training MAE': 1569408.690718311,
 'Valid MAE': 3626434.890206261,
 'Training RMSLE': 0.351825305462326,
 'Valid RMSLE': 0.559035624655174}
```

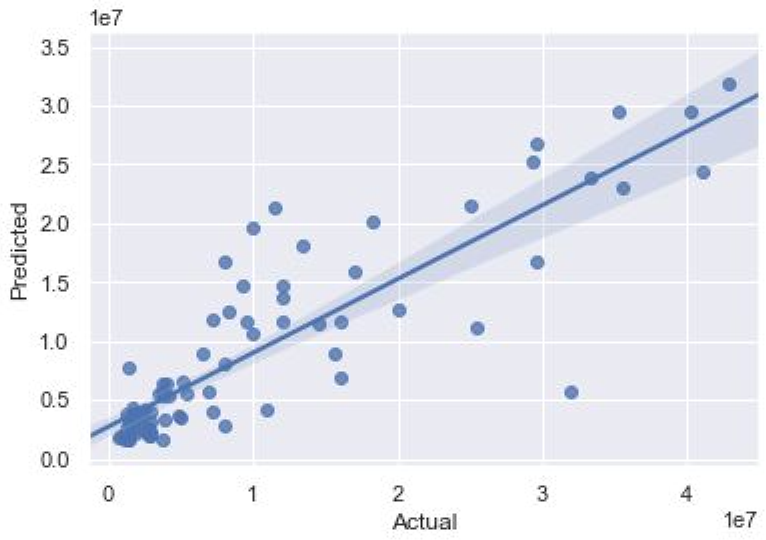
I also build a function to evaluate the model's mean absolute percentage error (MAPE). MAPE is a statistical measure of how accurate a forecast system is. It measures this accuracy as a percentage. The MAPE accuracy was 49.92% which means on average the model is off by 49.92%. Obviously that's not great, but it's not terrible for what I was trying to accomplish.

```
# Function to evaluate model's MAPE
def evaluate(model, X_test, y_test):
    errors = abs(y_pred - y_test)
    mape = 100 * np.mean(errors / y_test)
    print('Model Performance')
    print('Mean Absolute Percentage Error = {:0.2f}%.'.format(mape))

    return mape
```

Model Prediction

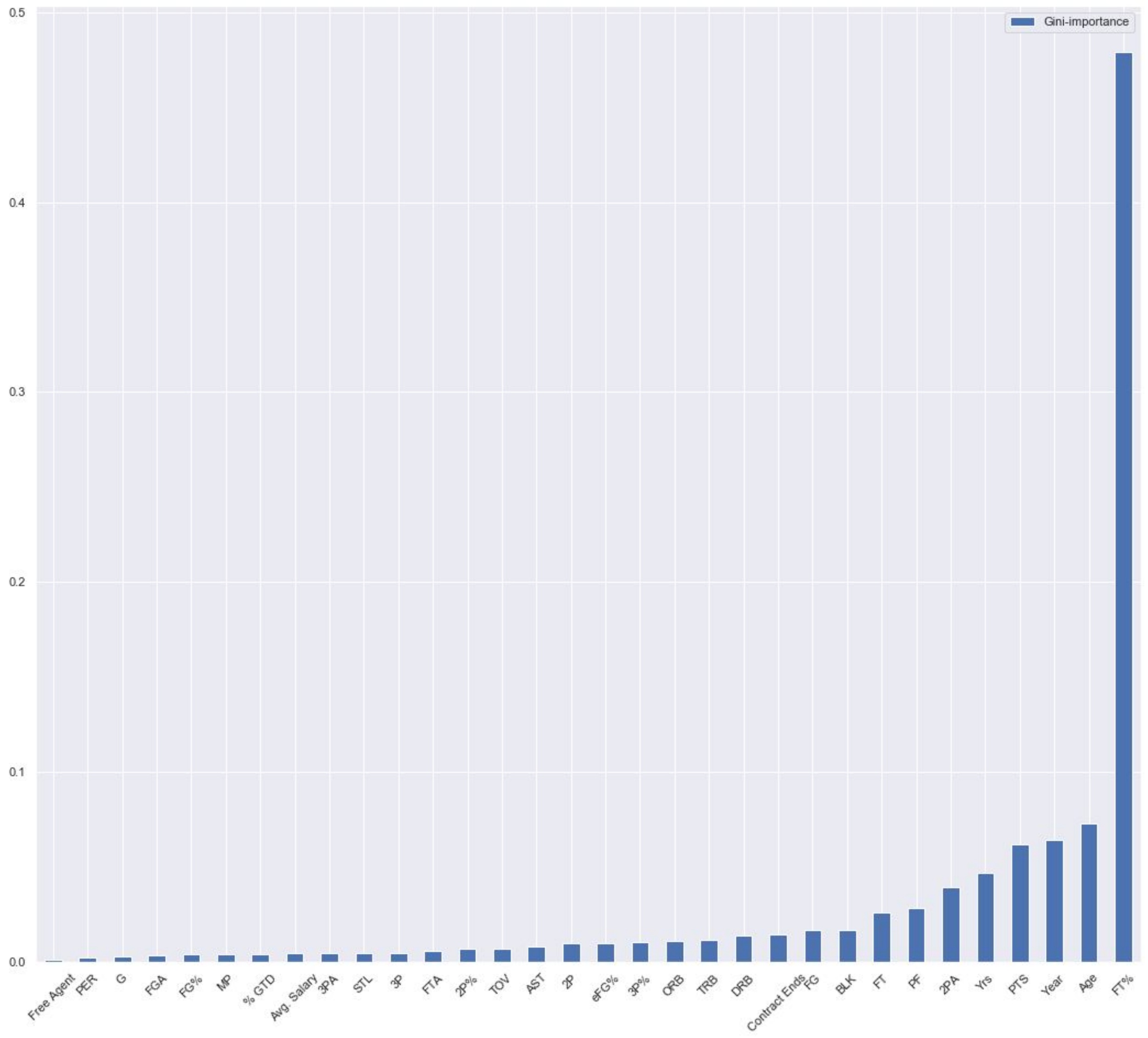
Once the model was trained, it was time to make predictions on the model using the test data. I needed to import LinearRegression class, instantiate it, and call the fit() method along with the training data. The linear regression model basically finds the best value for the intercept and slope, which results in a line that best fits the data. Then I compared the actual output values for X_test with the predicted values.



This linear line across the plot is the best available fit for the trend of the Predicted Average Salary (Avg. Salary) with respect to the Actual Avg. Salary. The data points far away from the line are the outliers. Looking at this plot, we may say that if the actual average salary is around $20,000,000, then the predicted average salary is around $15,000,000.

Feature Importance

The last thing I did was show feature importance. Feature importance seeks to figure out which different attributes of the data were most important when it comes to predicting the target variable (Avg. Salary). The results showed that the Free throw percentage (FT%) was the most important feature when it came to predicting Average Salary (Avg. Salary). Followed by points (PTS) and age (AGE).



My interpretation of this is that players that are efficient at getting to the line are making their free throws, which will also lead to more easy points. Automatically people will think of James Harden after reading that statement. Players' coaches from little league to the NBA have always preached that it is important to make free throws. Maybe they will show them this report to back the importance of making free throws.

Conclusion

Predicting an NBA player's worth is hard. You will never be able to know how well a player will play one year or how bad the next year. When predicting the average salary of NBA players our model did not do that great when you look at the numbers. The RMSLE was 35% and 55% for predicted and actual values respectively. The goal is to get those numbers as low as possible. I believe it is the best the model could have done with the data I used. I think a way to possibly improve the model could be trying to use more advanced stats as well. I am surprised PTS was not the most important feature used, but I am more surprised about FT% being number one.

The overall purpose of this project was to help NBA GMs when it came to negotiating with players on how much the players should get paid. GMs could use the model as a way to find the ideal range for a player's salary. This could keep them from offering too little or too much. GMs definitely want to avoid overpaying players and this model could act as a baseline for deciding on what to offer.

Code