Introduction

For many, watching the NBA is an exciting time of their year. They see their favorite players play the game they love and put on a show for entertainment at each stadium and viewers at home. I have watched the NBA every year for as long as I can remember. My parents are big fans of the Lakers (more so when Kobe was playing, my middle name is Kobe, after him) and would try to watch as many games as possible to try and keep up with the season.

Now because of the fans, the NBA's revenue topped over $10 Billion last year for the first time. With revenue at an all-time high, what does that mean for the players who put on the show? Some players are more interesting to watch and do better than others, but what determines how much a particular player gets paid in the NBA? Players like LeBron James and Stephen Curry are making $40+ million for this upcoming season. Do their stats from the previous year reflect that fairly? What determines that number?

In this project, I will be analyzing data about a player's first year's stats in the NBA and their salary for the following year to create a model to predict incoming NBA players' salaries based on their previous stats. Managers can use the model to see if their salary spending is valid, or the players themselves can see what they can improve on to receive a higher pay grade. It is always fun to see how much players are getting to put on a show for viewers, and my goal throughout this project is to build an effective model to accurately determine how much someone can make in the next year.

Dataset

I scraped all the data from Basketball Reference [1] and Kaggle[2]. I could find the most up-to-date salaries and numbers on Basketball Reference from the most recent and upcoming years. Kaggle provided usable data sets with every player, their stats, and wages, dating back to 1980. After combining these datasets, I joined them such that I matched each player's second contract after joining the NBA with their stats from the prior year. I reasoned that since I am using NBA stats, players would not have any for their first contract and that their performance in the NBA would strongly influence their second contract.

The final data set was extensive. It was initially overwhelming, but I could scale it to something manageable for exploratory data analysis (EDA). Here is the head of the final data set:

```
new_df.head()
```

	Player	Stats_Year	Ht	Wt	Colleges	Pos	Age	Tm	Team	G	...	PS/G	PA/G	SRS	Contract	rank	Contract_Year	Contract_Team	Salary
0	A.C. Green	1996	6-9	220.0	Oregon State	SF	32	PHO	Phoenix Suns	82	...	104.3	104.0	0.28	1	1	1997.0	Dallas Mavericks	5095088.0
1	Aaron Brooks	2008	6-0	161.0	Oregon	PG	23	HOU	Houston Rockets	51	...	96.7	92.0	4.83	1	1	2009.0	Houston Rockets	1118520.0
2	Aaron Gordon	2016	6-8	235.0	Arizona	PF	20	ORL	Orlando Magic	78	...	102.1	103.7	-1.68	1	2	2017.0	Orlando Magic	5504419.0
3	Aaron Gray	2008	7-0	270.0	Pitt	C	23	CHI	Chicago Bulls	61	...	97.3	100.4	-3.19	1	1	2009.0	Chicago Bulls	1000497.0
4	Aaron Harrison	2016	6-6	210.0	Kentucky	SG	21	CHO	Charlotte Hornets	21	...	103.4	100.7	2.36	1	1	2017.0	Charlotte Hornets	174570.0

5 rows × 51 columns

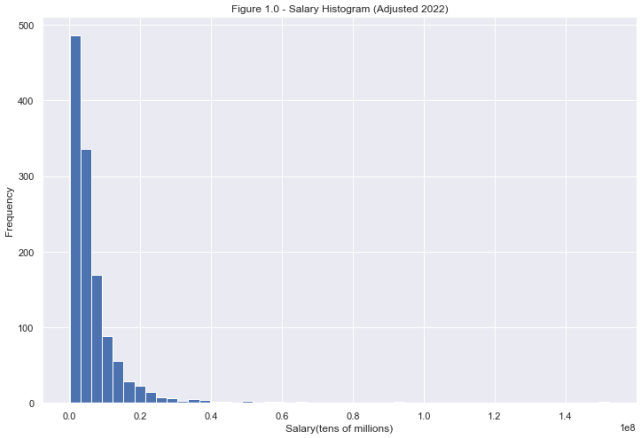
[1] Basketball-Reference http://www.basketball-reference.com/
[2] Kaggle http://www.kaggle.com/

There were a few columns cut off from the right of the data frame but noted at the bottom; there are 51 columns that are part of this data frame. That is a lot of columns, and most of them had abbreviations. I also created a few for feature engineering and EDA. To fully understand what each one is and what they mean, I have defined them and included them below:

‘Player’ — Player Name
'Stat_Year' — Year the stats were from
'Ht' — Height
'Wt' — Weight
'Colleges' — Which college they attended
‘Pos’ — Position
'Age' - Age of that year
‘Tm’ — Team (abbr.)
'Team' - Team Full Name
‘G’ — Games Played
'GS' - Games Started
‘MP’ — Minutes Played
‘FG’ — Field Goals Made
‘FGA’ — Field Goals Attempted
‘FG%’ — Field Goal Percentage
‘3P’ — 3-Pointers Made
‘3PA’ — 3-Pointers Attempted
‘3P%’ — 3-Point Percentage
‘2P’ — 2-Pointers Made
‘2PA’ — 2-Pointers Attempted
‘2P%’ — 2-Point Percentage’
‘eFG%’ —Effective Field Goal Percentage
‘FT’ — Free Throws Made
‘FTA’ — Free Throws Attempted
‘FT%’ — Free Throw Percentage
‘ORB’ — Offensive Rebounds
‘DRB’ — Defensive Rebounds
‘TRB’ — Total Rebounds
‘AST’ — Assists
‘STL’ — Steals
‘BLK’ — Blocks
‘TOV’ — Turnovers
‘PF’ — Personal Fouls
‘PTS’ — Points
'Pts Won' — Scored throughout year
'Pts Max' — If they scored high for that year
'Share' — Statistic divvying up team success for individual
'W' — Wins
'L' — Losses
'W/L%' — Win to Loss Percentage
'GB' — Games Behind/Back
'PS/G' — Points Scored per Game (by team)
'PA/G' —Points Against per Game (by team)
'SRS' — Simple Rating System
‘Contract’ — Player’s First Contract
‘Rank’ — Compared to Previous Year
‘Contract_Year’ — Year of Salary Listed
‘Contract_Team’ — Team of Salary Listed
‘Salary’ — Salary for that Year
‘Salary Cap’ — NBA Salary Cap per Team
‘Adj. Salary 2022’ — Targeted Value Created (Salary/Salary Cap

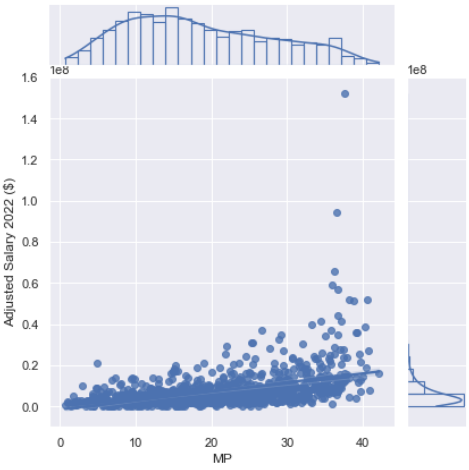
Exploratory Data Analysis (EDA)

Time to explore the relationships between the data. I wanted to visualize the data, but also try to answer some questions that I had about the data. The main questions I had were: Do the team's W/L% affect player's salaries? What are some features that determine salary? Does position affect a higher salary? On average, what is the age group that has the highest salaries? Which team has the highest total salary? The first graphs presented are seaborn joint plots. Seaborn joint plots create a scatter plot with histograms across the x and y-axes. Joint plots effectively



visualize where the data points lie, where the high and low points are, the distribution of the data between the two features, and where the outliers lie on the graph. All of the joint plots that were run can be viewed in the EDA notebook on my Github. Figure 1.0 is a quick look of the distribution among NBA player's salaries. As we can see it is significantly skewed to the left, with minimal outliers in the 0.4-0.6 range, meaning the majority of players make between $10-20 million in their second contract.

Figure 1.1
Minutes Played vs Adjusted Salary 2022



The figure on the left (Figure 1.1) is each player's minutes played (MP) and their adjusted salary for the 2022 year. I chose Adjusted Salary 2022 as my target variable and thought that MP would significantly impact a player's salary. More minutes, more opportunities, and the more important a player is since the coach often wants them on the court. However, though there are a few outliers, the robust and linear cluster of data points at the bottom of the graph and the generally normally distributed bar graph at the top of the joint plot show little correlation between the two features. With the pretty normally distributed histogram at the top and the right skewed histogram on the y-axis, the correlation coefficient is 0.22, showing there is little correlation of minutes played by a player to their salary for the next year.

Figure 1.2
W/L% vs. Adjusted Salary 2022

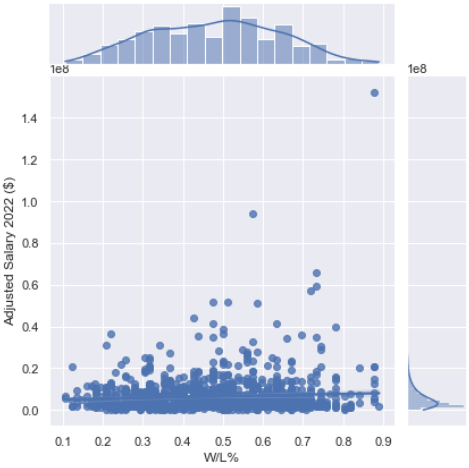


Figure 1.2 shows another feature I thought to be important: Win vs. Loss Percentage (W/L%). Interestingly, the correlation is very low, with a r-squared score of 0, meaning no correlation at all. W/L% looks normally distributed in the histogram with the distribution line, with all the data points clustered at the bottom. There are a few outliers here, and it is good to note that for both graphs thus far, Fig. 1.1 and Fig 1.2, the top right point is Michael Jordan, deemed the greatest player of all time. He will stand out in the following graphs, but it is interesting to note that most players in the NBA, regardless of their W/L% from a previous season, will have little effect on their incoming salary.

Figure 1.3
Points (PTS) vs. Adjusted Salary 2022

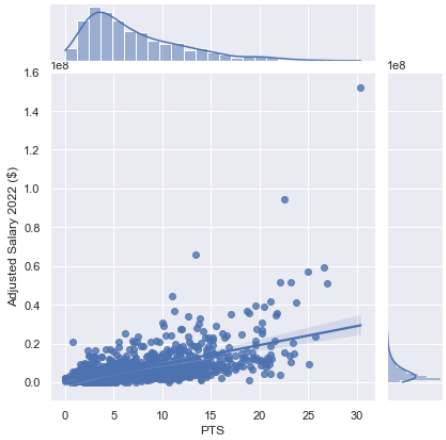
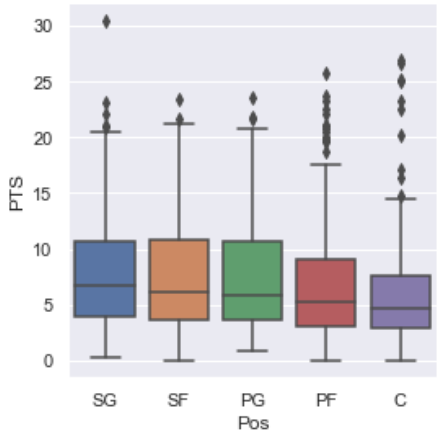


Figure 1.4
Points (PTS) vs. Position Box Plot



Position (Pos) vs. Points (PTS) vs. Adjusted Salary was a feature that seemed essential when analyzing how much a player gets paid. One would expect a player who scores much more than a teammate to earn more. Figure 1.3 shows us that there is a positive correlation between scoring more points and making more money, with a few outliers. When looking at the correlation coefficient for points concerning salary, it is the largest, with a value of 0.30. I also wanted to look at points by position; this was really to see if one position scored more on average than another. In Figure 1.4, we can infer that mostly everyone has an equal opportunity to score points and not one stands out more than another. Of course, except for Michael Jordan, extending beyond the average for shooting guards.

Figure 1.5
Blocks (BLK) vs. Adjusted Salary 2022

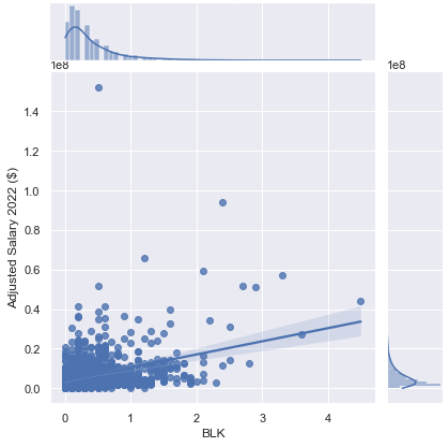
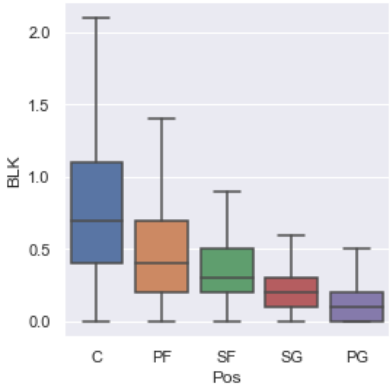


Figure 1.6
Blocks (BLK) vs. Position Box Plot



Blocks (BLK) is another feature stat that reflects a player's performance. Figure 1.6 is a fantastic swarm plot, showing that centers will have more blocks on average than the other position. Since centers are in the center, it makes logical sense to block more shots down low than a point guard or shooting guard would. Figure 1.5 shows a positive correlation between BLK and Adjusted Salary; the deviation in the histogram is clumped in with the average. Blocks

is a good stat to look at, and it shows it goes into the pool of stats to determine a salary, but it is not everything, as we will look at a little later.

Figure 1.7
Steals (STL) vs. Adjusted Salary 2022

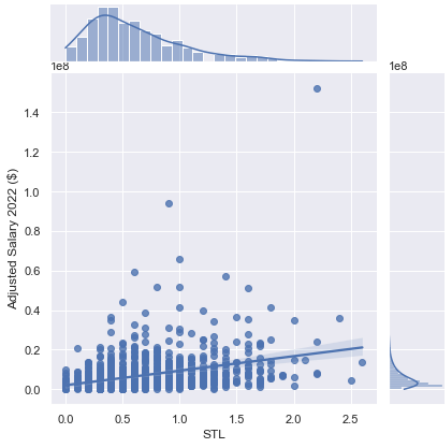


Figure 1.8
Steals (STL) vs. Position Box Plot

When watching the NBA, it is always fun to see players steal the ball and hit a fastbreak for an easy layup or a massive dunk (as long as it is not against your favorite team). Steals (STL) is another stat seemingly feeding into the weight of salary decisions. In Figure 1.7, organized data points form a positive correlation. As with blocks, the distribution is skewed to the left, but this one is slightly more distributed. Figure 1.8 shows which position gets the most steals, with point guards leading the pack, which makes sense since they guard the most ball-handling, heavy players.

Figure 1.9
Assists (AST) vs. Adjusted Salary 2022

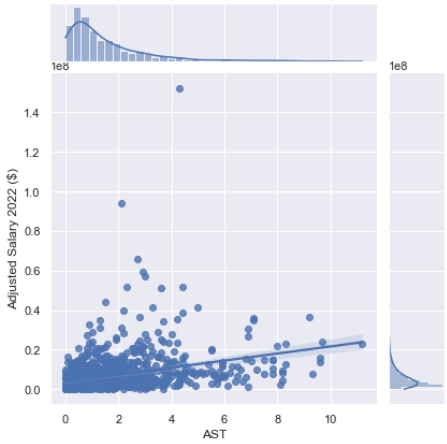
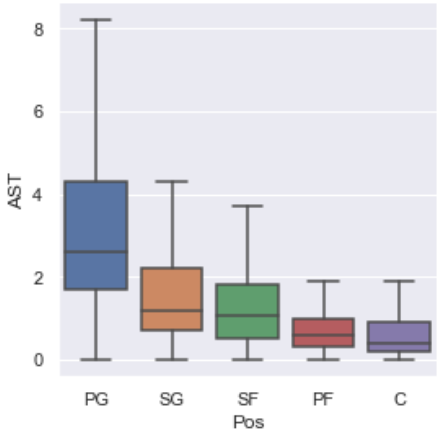


Figure 1.10
Assists (AST) vs. Position Box Plot



Lastly, we will look at assists (AST). Assists are often beautifully done, and although the scorer gets the points stats added to their totals, the credit for an assist goes to the passer. Figure 1.10 shows the plot of positions that have the most assists. On average, point guards are the ones who do the best in this category since they are the playmakers and try to find the best opening in

the first place. Figure 1.9, like the previous joint plot figures, shows that it does have a say in salary numbers, resulting in a positive correlation with assists. Also, similar to the previous graphs, the distribution is heavily skewed left on the x-axis, showing that it is not a heavy influencer on salary numbers but is still included with the rest of the stats.

I also wanted to conduct box plots based on position and age to see if that had any significant impact on salary.

Figure 1.11
Age (Binned) vs. Salary

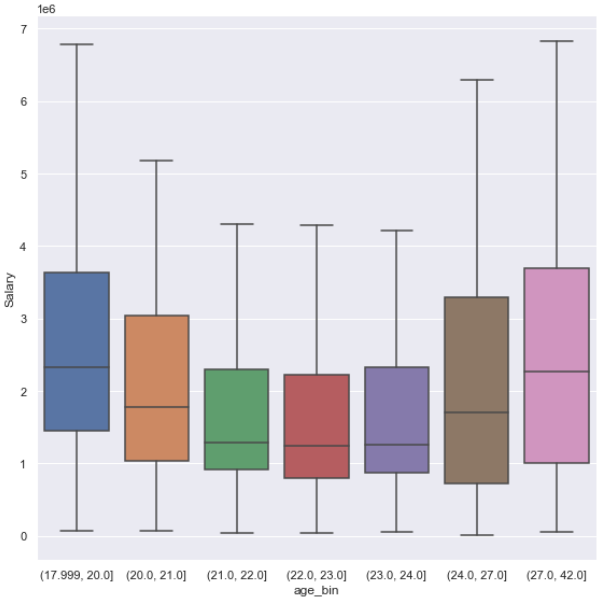
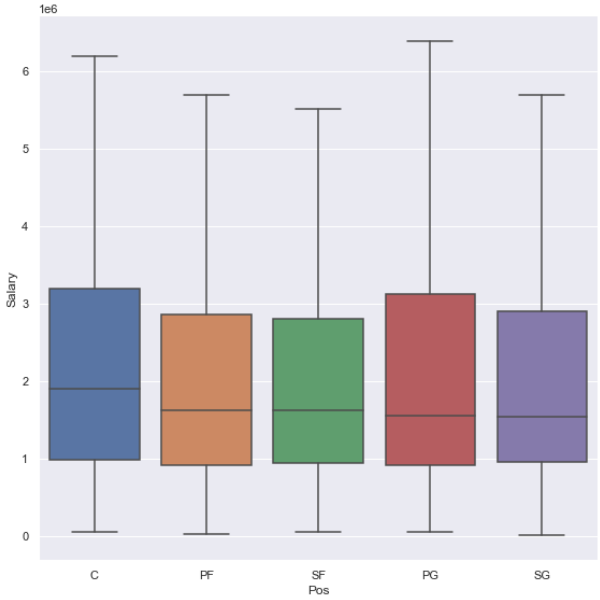


Figure 1.12
Position (Pos) vs. Salary



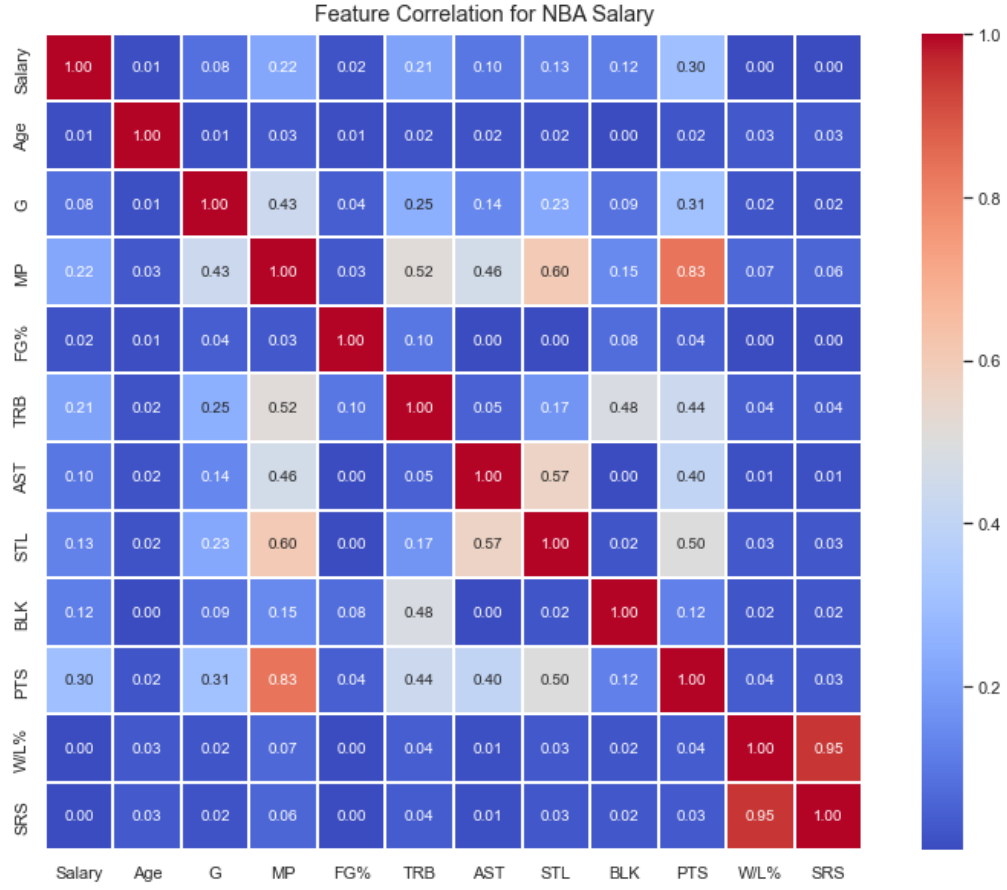
In the age boxplot (Figure 1.11), I used the q-cut method to size each age bin appropriately. Binning by age is significant, especially since the first bin (ages 18-20) is such a substantial bin. In 2006, the NBA required all prospective players to attend at least one year of college and be 19 years old before they could be eligible for the draft. Before that, many players would get drafted right after high school, making them very young and making good money out of the gate[3]. There is a drop after the first bin, which continues dropping before rising again at ages 23-24. While some all-stars may be drafted post-high school or, due to the rule, after one year in college, some players get drafted after college, resulting in that age. The NBA weeds those players out, and if they get to the 24-27 range, that is when the players start to make money, and from the graph, salaries will only go up.

Most of what came out of EDA was expected. The joint plots showed mostly positive skewed relationships, the heatmap highlighted which features corresponded with one another the most, and the box plots showed the different ranges for categorical values.

The following heatmap (Figure 1.13) is the correlation between the features with the Pearson correlation coefficient calculated in each square. In this heatmap, all the values are above or equal to 0.0, meaning none of the features negatively impact each other. Right off the

bat, viewers can infer that most components have little effect on salary. PTS has a coefficient of 0.30, which is notable to point out, which makes sense but is not a large enough value for multicollinearity and can be used in the modeling. Points and minutes played have a high correlation. Still, both features have little correlation to salary, which is good since they can be included in the modeling processes as essential features. High collinearity of two variables means that strong correlation exists between them, making it difficult or impossible to estimate their individual regression coefficients reliably. Hence, it being a good thing so I am able to include them all in the modeling process.

Figure 1.13
Feature Correlation for NBA Salary



Feature importance was run using a random forest regressor, which showed which features are most important and should be used in the modeling process. Figure 1.14 shows which features ranked the highest, and Figure 1.15 is a beeswarm plot of it. By plotting the regressor array, the visualization of essential features such as ‘2P,’ ‘MP,’ and ‘Age’ was intriguing. I did not expect ‘2P’ to be more critical than the general ‘PTS’ feature. ‘MP’ was expected, as mentioned earlier. Interesting to see how the feature importance may differ from plotting features directly against each other. Seeing ‘Age’ as an essential feature was also expected, and I find it interesting to see it impact determining salary, but after seeing the boxplot in Figure 1.11, it makes sense.

Figure 1.14

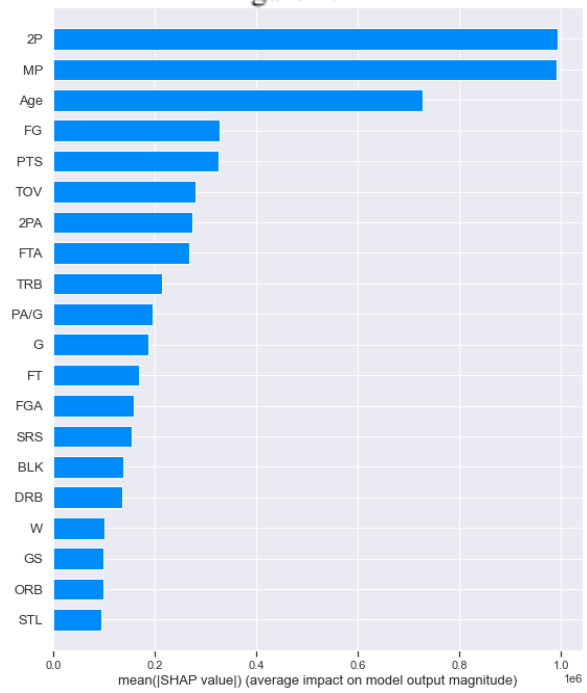
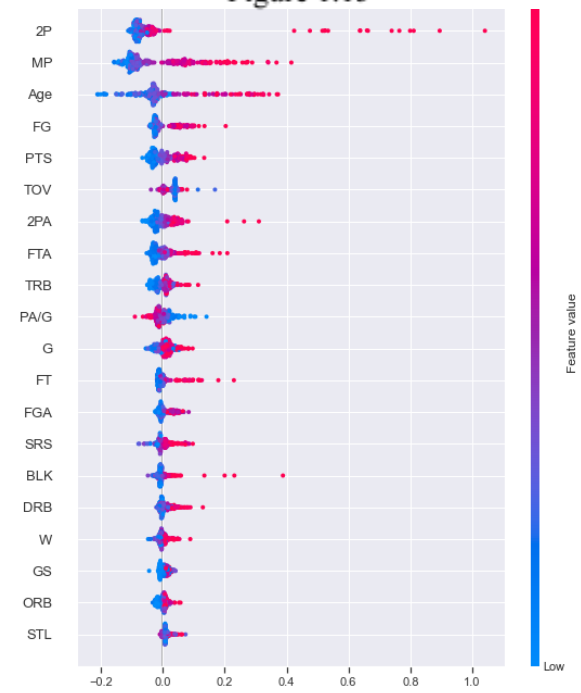


Figure 1.15



Pre-Processing and Training Data Development

The pre-processing and training data development notebook is focused on three tasks:

1. Create dummy or indicator features for categorical variables.
2. Standardize the magnitude of numeric features using a scaler.
3. Split the data into testing and training datasets.

I created dummy variables for positions and contract teams and standardized them using the Scaler() function. I then used scikit-learn's model selection train_test_split function to split the data into testing and training datasets. I chose a 70/30 split for training vs. testing sizes.

Modeling

The project's main goal was to see if we could predict an NBA player's second contract based on their stats from the prior year. In the modeling section, I ran two different tests. I first ran a dummy model where the mean absolute percentage error (MAPE) resulted in 304.7%, which means there is an extremely low accuracy in the model being built. The mean squared error (MSE) resulted in 11,367,831, which is very bad.

Beginning modeling, I first ran a Ridge Linear model to tune the multicollinearity in the dataset. However, when the alpha parameter was 10, the score for the Ridge model resulted in extremely high numbers. The MAPE was 172%, and the MSE was 7,147,890. The MSE is lower than the dummy model, but still quite high if this model were really to be used to help determine a players salary as the MAPE is over 100%.

The second model that was run was a Random Forest Regressor. While the initial best score for the random forest model was -1.31, the train-test-split resulted in a very high number.

The MAPE was 167%, and the MSE resulted in 8,326,467. Yikes. Still resulted in a lower score than the dummy model, but higher than the Ridge Linear Model, meaning both models ended up being over 100% for mean absolute percentage error.

Conclusion

The main goal was to build a model that could be used to predict a player's salary. Ultimately, while a lot went into the process of getting this done, the model produced a very high percentage error, and there would be much work to get those numbers down to use the model for its initial purpose. However, my models did significantly outperform the dummy, so it is possible that this may simply be a difficult prediction task.

I enjoyed being able to challenge myself and learn a lot about what it means to build a machine learning algorithm. There is so much more to learn and, even if the models ended up with high mean errors, my takeaways were from the development process rather than the results. Upon reflection, I could have done a few things differently:

1. **Dealing with time value in money differently.** The value of money is changing every day and what was used in the dataset was a relative salary adjustment by taking the current 2022 salary cap, and the player's salary for that year and dividing the two. If I were to implement an inflation code or scale the weight of their salary compared to each other, the values could be different for each player.
2. **Adding more features and taking out the ones that truly did not matter.** The dataset was extensive and included many features that did not matter. I also added many columns for analysis, but maybe all of them did not have the impact I intended. I am sure there are many advanced features I could have researched and put into this dataset for analysis.
3. **Used more machine learning models.** While I only used two models, I could have implemented a few different ones. I would want the models to result in a lower MSE and MAE, but that would be impacted by work done before the test-train-split. Therefore, more previous work would be valuable for producing different machine-learning algorithms.
4. **Doing more fine tuning.** The parameters used were general numbers that could get general results from the models. If I hyper-tuned the parameters, a lower MAE and MSE could be possible. Still, as mentioned earlier, a lot more work would have to be implemented beforehand to make it possible and render the machine-learning model useful.