Machine Learning Engineer Nanodegree

Capstone Project

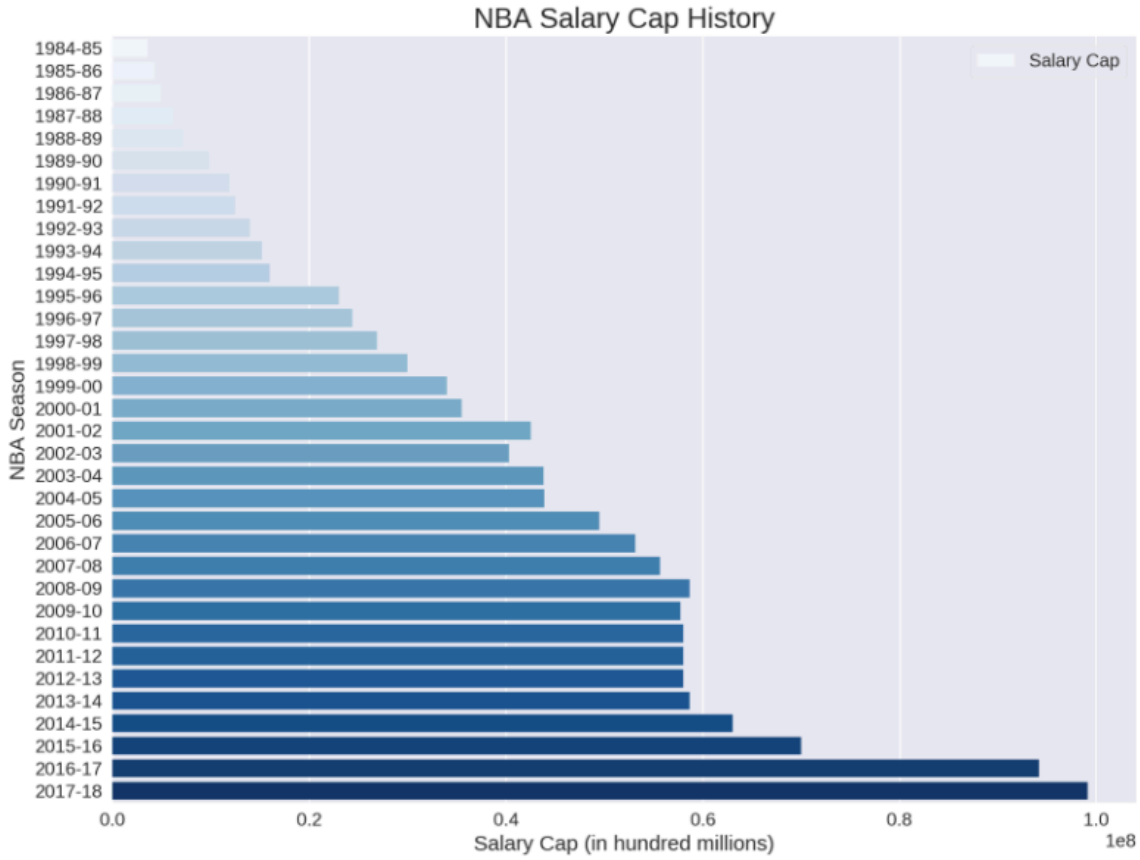
Michael Kim
December 13th, 2018

I. Definition

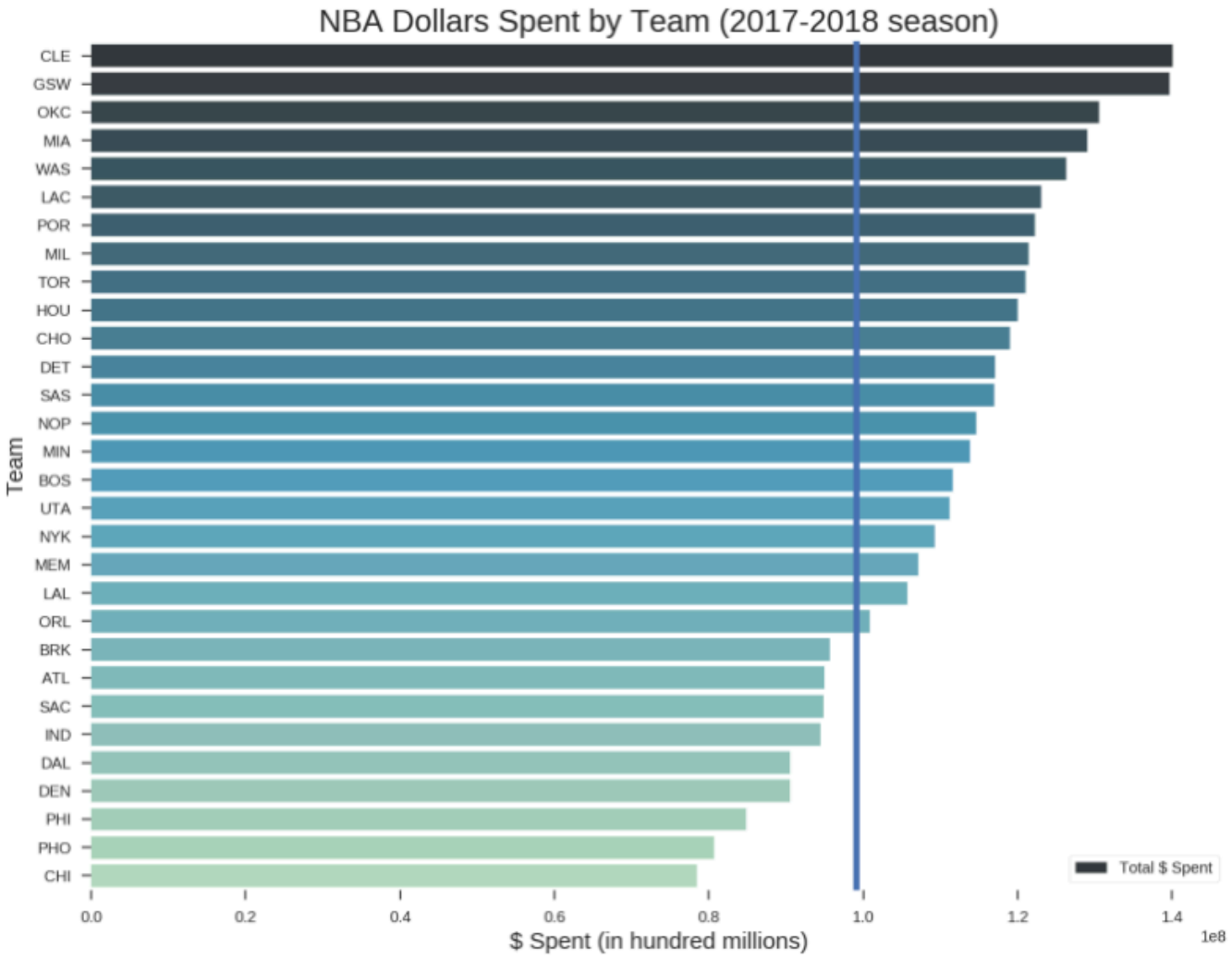
Project Overview

The National Basketball Association (NBA) is widely considered to be the premier men's professional basketball league in the world and features some of the most well-known athletes. As is such, NBA players are the world's best paid athletes by average annual salary per player. And their increasing salaries have led to salary caps which limit teams' total salaries. This limit is subject to a complex system of rules and exceptions; therefore this is considered a "soft" cap.

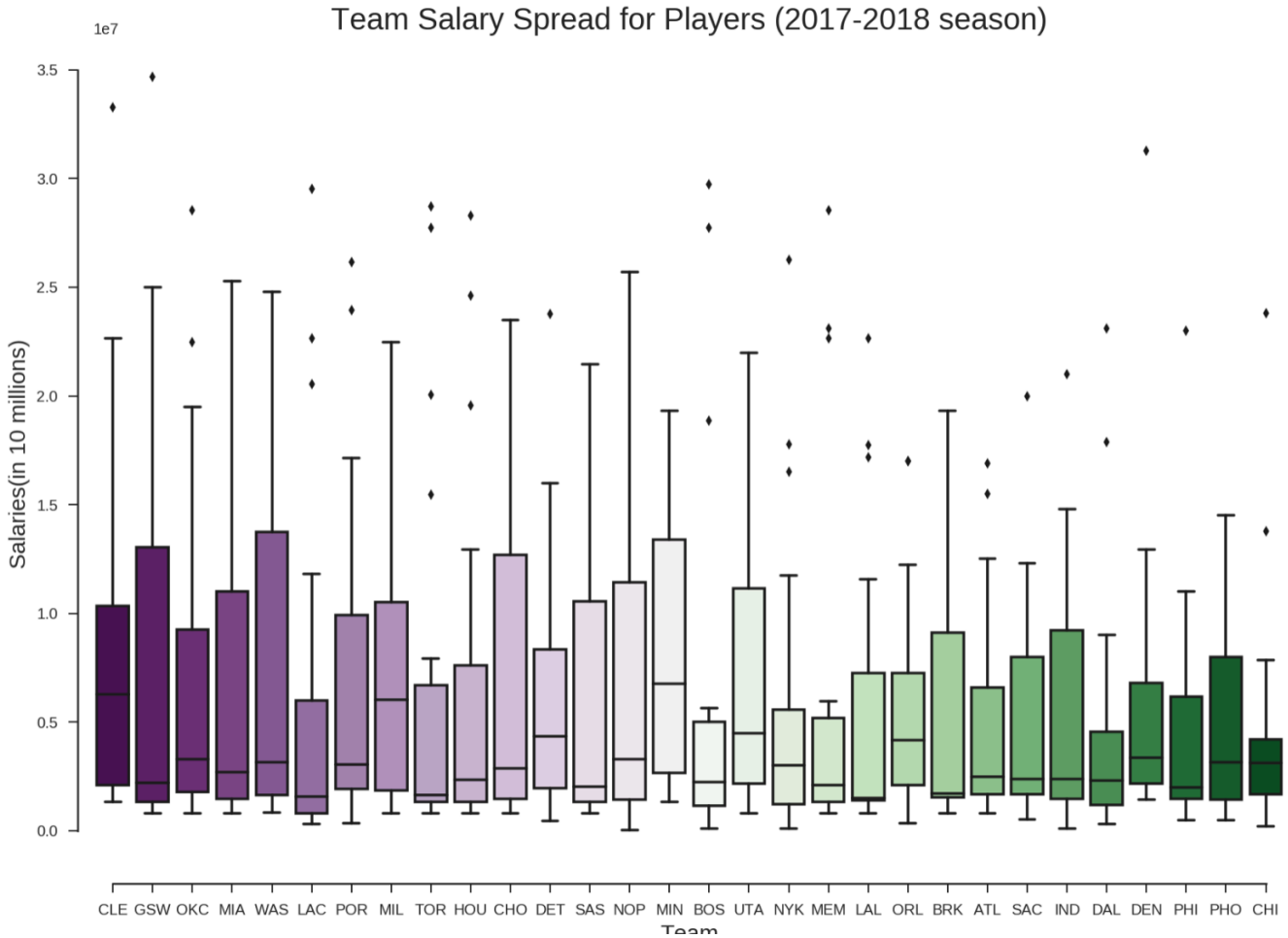
The plot below shows the historical salary cap values of the NBA. The NBA players' salary cap was instituted in the 1984-1985 season. That year the salary cap was 3.6 million and the average player salary was 300,000. By 2013, 66 percent of NBA players earned more than 1 million or more. By the 2012-2013 season, the salary cap increased to 58 million and the average NBA player salary was 5.1 million. In the upcoming 2017-2018 NBA season, the salary cap is almost at the 100 million mark with the highest salary at nearly 35 million and average at 6.5 million.



The plot below shows salaries used by each team for the upcoming 2017-2018 season. As you can see, more than half of all NBA teams surpassed the designated salary cap showing that it's a "soft" cap with many exceptions. The vertical line indicates the salary cap for the season.



The escalation of NBA player salaries has not only been explosive, but it has also created a large earnings gap between players. The box plot below shows the spread of teams' salaries and the spread per player (teams with the largest total salary on the left and lowest on right). As seen below, there are many outliers well beyond the median (as represented by the ticks). For example, the Golden State Warriors are paying Stephen Curry a whopping 34,682,550 (the highest tick mark) this season which is greatly higher than Warrior's teammate Jason Thompson at just 945,126.



Problem Statement

The issue of salaries has been a huge topic of controversy in the NBA and has led to several lockouts over these labor disputes. There were NBA lockouts in the 1995, 1996, 1998 and 2011 NBA seasons and the main issue in each of them was related to players' salaries. Owners felt players were overpaid, and players felt as if their earning power was restricted. And the topic of NBA salaries continues to be a hot topic of discussion today.

Increasingly, NBA players have come under intense scrutiny for their large salaries. Many questions have been asked, including why they are paid what they are and how are salary decisions made. Although these questions are beyond my scope as I don’t know anything of negotiating contracts and deals with professional athletes. The topic did peak my interest and I wanted to build out a machine learning model using quantitative data to predict these large salaries.

I will be using supervised learning machine learning models to develop the predictive model for salaries. The purpose of this project is to help settle disputes regarding NBA players’ salaries and to identify the variables that are most likely to contribute to a player’s salary.

The purpose of this project is to help settle disputes regarding NBA players' salaries and to identify the variables that are most likely to contribute to a player's salary. Unlike coaches who are mainly hired and paid based on a single metric (wins), players are hired and paid based on individual performance, which can be

measured by their on-the-court metrics. Though there is a lot of literature regarding what determines NBA salaries, there is very little statistical backing to their claims.

I believe there are certain performance determinants that will offer an accurate prediction for salary. Perhaps teams put greater premium for better performance in certain aspects of the game. These kind of questions are what I hope to explore and answer.

This project has five parts: webscraping, data cleaning, exploratory data analysis, algorithm development, unsupervised learning analysis.

The machine learning pipeline will include some steps including scaling the data, using pipelines to chain steps together, using grid search cross validation in order to tune hyperparameters and then building machine learning models using the following algorithms: linear models, ensemble decision trees, neural networks, k-means and principal component analysis.

Metrics

To assess the accuracy of the models that I develop, I will be evaluating the R-squared value, mean absolute error and mean squared error. I will also look at coefficients and feature importance in order to understand which independent variables were better predictors of salary. The R-squared value is a useful measure that ranges from zero to one – with zero indicating that the proposed model does not improve prediction over the mean model, and one indicating perfect prediction. Improvement in the regression model results increases the value of R-squared -therefore, making it a useful measure to look at in order to compare models developed for this project. The mean squared error is an absolute measure of fit and it can be interpreted as the standard deviation of the unexplained variance – and has the useful property of being in the same units as the response variable. Lower values of mean squared error indicate a better fit. We'll also measure mean absolute error which is more robust to outliers as it doesn't penalize large deviations compared to mean squared error.

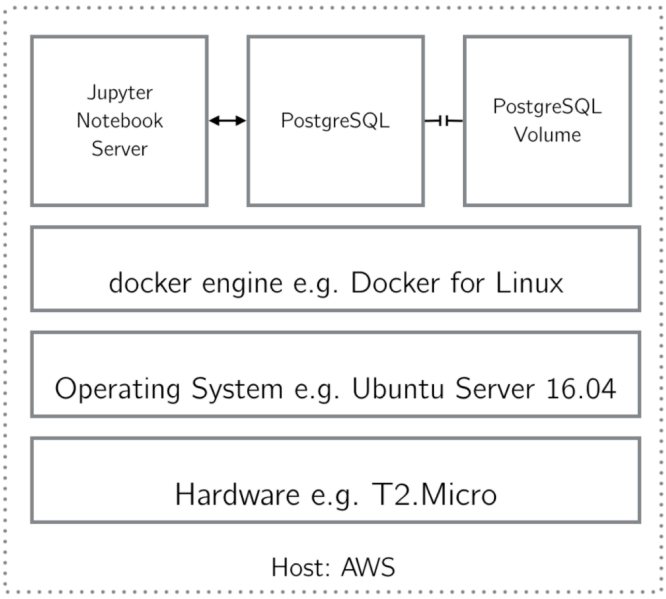
I will also be assessing P-values to understand which statistical features were relevant to the data. In this case, the null hypothesis would be that there is no relationship between the particular feature and the target variable (salary). So the null hypothesis would assume that on-court-performance statistics are independent of salary. Therefore, if any particular feature has a low p-value, specifically lower than 0.05, then we say that particular feature is statistically significant and we reject the null hypothesis and conclude that there is a relationship between that feature and the target.

II. Analysis

Data Exploration

Unlike Glassdoor or Indeed, NBA salary information is available online, as well as player statistics. Though there are many datasets different people acquired and put out on the web, none of them were substantial enough to develop a model on. Therefore, I am planning to scrape basketball sites in order to get the data for this project.

I will be using the BeautifulSoup Python library in order to scrape the data and load them into tables within a PostGres database. All the code for the web scraping will be included as a part of my project. In order to manage environments and data storage, I am going to engineer the entire setup using Docker. With setting up Docker containers, we run into the issue of data persistence as when the PostGres container goes down, that data within that container is lost. And so to solve this issue, I created a PostGres volume that houses all the data so that whenever our container goes down, the volume will hold all of it. And when the container comes back up, the volume will operate within that container. All relevant Docker and yml files will be included. A diagram of the environment that I will be using is below.



The dependent variable for this study was NBA player salaries and the independent variables were the offensive and defensive statistical categories.

We are going to use the salaries and statistics of 486 NBA players from the most recent season. I decided to only use the statistics from the most recent season as they would be most reflective of the current salary rates. The salary cap for the NBA has been increasing at a rate faster than inflation and so it wouldn't make for a good model to bring in statistics from multiple years.

Data Dictionary below but a more detailed descriptions of data can be found in the “data_descriptions.txt”

- Player: Player Name -- TEXT
- Position: Position - TEXT
- Shooting_Hand: Hand that player shoots with -- TEXT
- Height_inches: Height of player -- INTEGER
- Weight_lbs: Weight of player -- FLOAT
- College: College that player played at -- TEXT
- Draft_Year: Year player was drafted -- INTEGER
- Draft_Position: Rank in draft -- INTEGER
- Season_Count: Number of seasons played in NBA - INTEGER
- Age: Age of Player at the start of February 1st of that season -- INTEGER
- G: Games -- INTEGER
- GS: Games Started -- INTEGER

- MP: Minutes Played -- FLOAT
- FG: Field Goals -- FLOAT
- FGA: Field Goal Attempts -- FLOAT
- FG_Perc: Field Goal Percentage -- FLOAT
- Three_P: 3-Point Field Goals -- FLOAT
- Three_Att: 3-Point Field Goal Attempts -- FLOAT
- Three_Perc: 3-Point Field Goal Percentage -- FLOAT
- Two_P: 2-Point Field Goals -- FLOAT
- Two_Att: 2-Point Field Goal Attempts -- FLOAT
- Two_Perc: 2-Point Field Goal Percentage -- FLOAT
- EFG_Perc: Effective Field Goal Percentage -- FLOAT
- This statistic adjusts for the fact that a 3-point field goal is worth one more point than a 2-point field goal
- FT: Free Throws -- FLOAT
- FTA: Free Throw Attempts -- FLOAT
- FT_Perc: Free Throw Percentage -- FLOAT
- ORB: Offensive Rebounds -- FLOAT
- DRB: Defensive Rebounds -- FLOAT
- TRB: Total Rebounds -- FLOAT
- AST: Assists -- FLOAT
- STL: Steals -- FLOAT
- BLK: Blocks -- FLOAT
- All_Star: All Star status, 1 if they were all star at some point in career, 0 if not -- INTEGER
- TOV: Turnovers -- FLOAT
- PF: Personal Fouls --- FLOAT
- PTS: Points -- FLOAT
- PER: Player Efficiency Rating - FLOAT
- A measure of per-minute production standardized such that the league average is 15
- WS: Win Shares -- FLOAT
- An estimate of the number of wins contributed by a player
- Salary: Salary for the 2016-2017 season – FLOAT

Preview of the data

	age	all_star	ast	blk	college	draft_position	draft_year	drb	efg_perc	fg	...	three_att	three_p	three_perc
0	24	0	4.0	13.0	Purdue University	46	2016	28.0	0.464	17.0	...	10.0	5.0	0.500
1	32	0	125.0	9.0	University of Oregon	26	2007	51.0	0.483	121.0	...	128.0	48.0	0.375
2	21	0	150.0	40.0	University of Arizona	4	2014	289.0	0.499	393.0	...	267.0	77.0	0.288
3	22	0	3.0	0.0	University of Kentucky	0	0	3.0	0.000	0.0	...	2.0	0.0	0.000
4	25	0	7.0	7.0	Michigan State University	15	2014	24.0	0.454	23.0	...	15.0	3.0	0.200

Describe statistics of the data

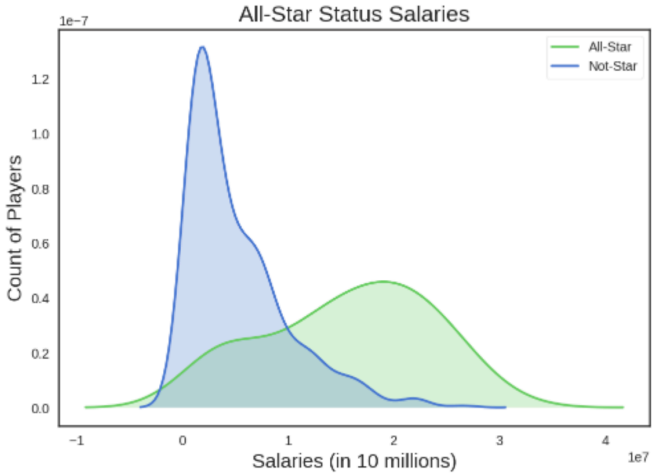
	ast	blk	efg_perc	fg_perc	ft_perc	per	pf	pts	salary	season_count	stl	three_perc
count	486.000000	486.000000	485.000000	485.000000	471.000000	486.000000	486.000000	486.000000	4.860000e+02	486.000000	486.000000	450.000000
mean	123.726337	26.296296	0.492686	0.440458	0.741089	13.096914	110.462963	581.205761	6.405725e+06	6.662551	42.425926	0.299148
std	144.106594	33.628392	0.098757	0.099734	0.139716	5.596723	79.373044	544.548491	6.159416e+06	5.018452	36.621126	0.131579
min	0.000000	0.000000	0.000000	0.000000	0.000000	-17.600000	0.000000	0.000000	2.500000e+04	1.000000	0.000000	0.000000
25%	26.250000	5.000000	0.464000	0.399000	0.678500	9.800000	43.000000	155.000000	1.551659e+06	2.000000	14.000000	0.267000
50%	77.000000	16.000000	0.500667	0.442000	0.766000	12.800000	111.500000	444.500000	4.106147e+06	6.000000	36.000000	0.333000
75%	164.000000	34.000000	0.537000	0.485333	0.832000	15.800000	154.000000	833.500000	8.590168e+06	10.000000	61.750000	0.375000
max	907.000000	248.000000	1.000000	1.000000	1.000000	31.500000	556.000000	3884.000000	3.096345e+07	23.000000	232.000000	1.000000

Exploratory Visualization

In an effort to get an idea of what players are worth, judging strictly on quantative data, I am building a machine learning model to estimate player salaries based on their on-the-court performance.

But why use on-court performance statistics as our predictors? What makes them good variables to use? Aside from the inherent assumption that better performance leads to higher pay, what hard evidence from the data would suggest this? I didn't want to build out a model based on assumptions, so in my exploration of the data, I tried finding a variable in the data that would encompass all the performance statistics. So I looked at the All-Star status field - this is a binary indicator of whether a player is an NBA all star or not.

NBA all stars are selected based on their high performance on the court. Below is a histogram that shows the salary distributions based on whether a player was an all-star or not. A player's all-star status is usually attributed to their performance on the court. There are other factors as well but by in large, it's based on how well they are performing. So we could see that there is probably a relationship between on-court performance and salaries.



Algorithms and Techniques

This project will comprise of five parts: web scraping, data cleaning, exploratory data analysis, supervised learning algorithm development, and unsupervised learning analysis. The unsupervised learning analysis will

be something extra that I want to do to see if I could develop a clustering algorithm that would re-define the conventional 5 positions in basketball using basketball statistics.

Tools that are being used for this project are: Python, BeautifulSoup, RegEx, Pandas, Numpy, PostGres, Seaborn, Matplotlib, Docker, AWS, Machine Learning algorithms (linear models, ensemble decision trees, neural networks).

Scaling our data

We need to scale our data for the following reasons:

- To handle disparities in units
- Cut computational expense
- Improve model performance (Especially Machine Learning)
- We scale for models to prevent the steps on different axes from varying widely

The most common method of scaling is standardization and this is the type of scaling that I'll be doing for this data set. In standardization we first center the data, then we divide by the standard deviation to enforce that the standard deviation of the variable is one:

$$X_{std} = \frac{X - \bar{X}}{s_X}$$

There are many benefits to standardizing the data, especially when we have more than one predictor:

- Intercepts are interpreted as the estimate when all predictors are at their mean value
- Coefficients are in units of standard deviations of the original predictors. This allows for direct comparison of the magnitude of impact between different predictors
- Optimization methods (minimizing loss functions) are faster and more stable
- It is required for regularization penalties where the magnitude of coefficients for different predictors must have the same meaning
- In K-Nearest Neighbors methods it is necessary if you want features to contribute equally since these models use the distance between observations calculated from the features
- In logistic regression, neural networks, and support vector machines unscaled data can result in a disproportionate effect of some data points over others

Pipeline

Pipeline is a class in sklearn that allows us to chain steps together.

We add steps to the pipeline using a list of tuples of the form [('step name', sklearn object)...]

I am planning to use a pipeline in order to chain these pre-processing steps together and run that through a GridSearch Cross-Validation (about that below).

GridSearchCV

Each of the models we are going to build have what are called hyperparameters, which affect how the model learns the data. By changing them, we can optimize a model's performance on new data and reduce variance!

The trick is to add another set of splits within our model. We're going to use cross-validation within the training set to fit our models and tune our hyperparameters.

Setting up GridSearchCV

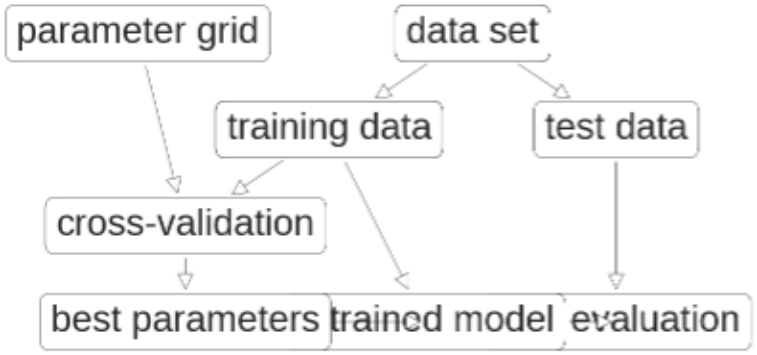
1. Estimator
First, pick your estimator (which will be the models we use). This can be any sklearn model. This is passed to the estimator argument.

2. Parameter Grid:
Then we need to create a parameter grid. This allows us to input values for all the hyperparameters we'd like to tune! Your param grid is then passed to your grid search object through the argument param_grid.

3. Cross-Validation:
Cross-validation is defined by the cv= argument. cv can be an integer, ShuffleSplit, or StratifiedShuffleSplit object.

- cv=5 will perform 5-fold cross-validation
- ShuffleSplit will shuffle your samples for random validation sets. Default cv argument for this object is 10. If you want 5-fold cross-validation using all the data with ShuffleSplit, use cv=ShuffleSplit(cv=10, test_size = .2).
- StratifiedShuffleSplit is similar to ShuffleSplit, but can only be used for classificaion. This ensures that the class proportion in your target are preserved in each validation split.

GridSearchCV also takes an argument for n_jobs, with a default of 1. Many sklearn objects take this argument. If you set n_jobs=-1, sklearn will parallelize this process accross all the cores in your machine. This really speeds up the process and so that is what I'm going to do for the models that we implement here.



Linear Models

Implement a series of linear models with regularization to add constraints or penalize the model to prevent overfitting and improve generalization. One such regularized linear model is ElasticNet regression. I chose to build linear models as many of the data points seem to have fairly good linear separation.

Ensemble Decision Trees

I also implemented non-parametric models such as Random Forest and Extra Tree regression. These ensemble decision tree models help us avoid the dangers of mismodeling the underlying distribution of the data. For example, if our data has no linear relationships, linear models aren't going to perform well on that data.

Random forests are a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. The basic principle is that a group of "weak learners" can come together to form a "strong learner." Random Forests are a wonderful tool for making predictions considering they have a lower tendency of overfitting because of the law of large numbers.

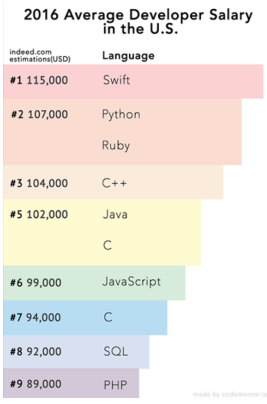
Neural Networks

We can create neural networks for our regression problem by using Keras and evaluate them with scikit-learn by using the wrap provided by the Keras library. he efficient ADAM optimization algorithm is used and a mean squared error loss function is optimized. This will be the same metric that we will use to evaluate the performance of the model. It is a desirable metric because by taking the square root gives us an error value we can directly understand in the context of the problem.

Benchmark

The development of this project is based on the hypothesis that a player's performance variables such as points per game, field goals, etc. would be significant contributors to player salaries. The dependent variable for this project was NBA player salaries and the independent variables were the offensive and defensive statistical categories.

In the rest of the job market, one's skill-set and the amount of experience they have using those skills determines salaries and with that information, applications like Glassdoor and Indeed can accurately predict how much a person would make for any given job. The plot below was determined with such information by Indeed.



Likewise with the NBA, I believe there are certain performance determinants that contribute more to higher salaries. Perhaps teams put greater premium on points per game than they do on total rebounds per game. These kind of questions are what I hope to explore and answer. The benchmark model would then be a very basic model where all the independent variables all had equal weight in determining salary. I am forecasting that such a benchmark model will perform poorly in predicting salary as there are probably certain statistical measures that teams are looking for.

III. Methodology

Data Preprocessing

With data that is scraped from the web, the data is messy and needs to be preprocessed before it can be used.

- Get rid of a couple of rows (that were header rows) that contain only NoneType values
- Rename some of the columns
- Change to proper data types
- Deal with missing values

Additional data pre-processing that were important for the model are detailed below.

Dropping features not related to on-court performance

There are some features that are not related to on-court performance. There might be indicators that are related to past performance, like draft_position, which is related to how well a person did in college. But our model is more concerned about actual on-court performance at the professional level. And there are other fields that are simply not reflective of performance at all, such as shooting_hand, height and weight. So we are going to drop the following columns.

Dropped these fields

- Player
- Position
- Shooting_Hand
- Height_inches
- Weight_lbs
- College
- Draft_Year
- Draft_Position
- All_star

Dropping Player Efficiency Rating (PER)

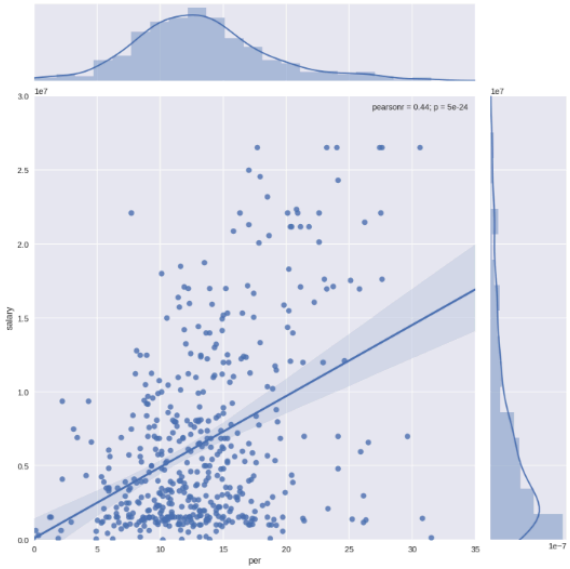
PER is an advanced statistical measure of per-minute production standardized such that the league average is 15. It combines various factors but there is a major flaw with it.

PER largely measures offensive performance. Two of the defensive statistics it incorporates—blocks and steals (which was not tracked as an official stat until 1973)—can produce a distorted picture of a player's value and that PER is not a reliable measure of a player's defensive acumen.

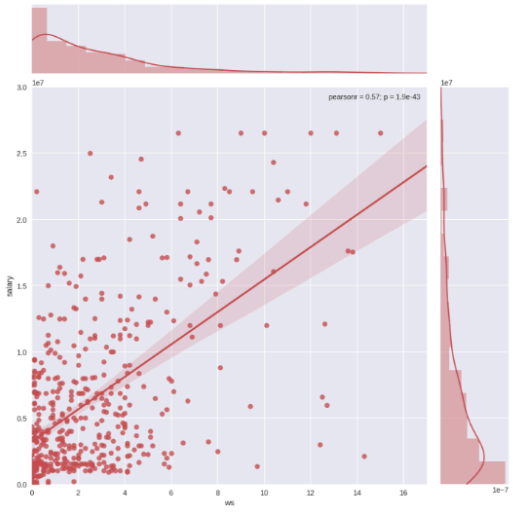
Therefore, it's a statistic that will add onto offensive statistics and may even "double" count it without including defensive statistical factors.

Because of this unbalance weight towards offensive statistics, we are going to drop this column.

Below is a diagram of PER and salaries. If PER would be a good determinant of salary, then it should follow a similar distribution as the salaries. But from the joint plot below, the distributions seem to be different.



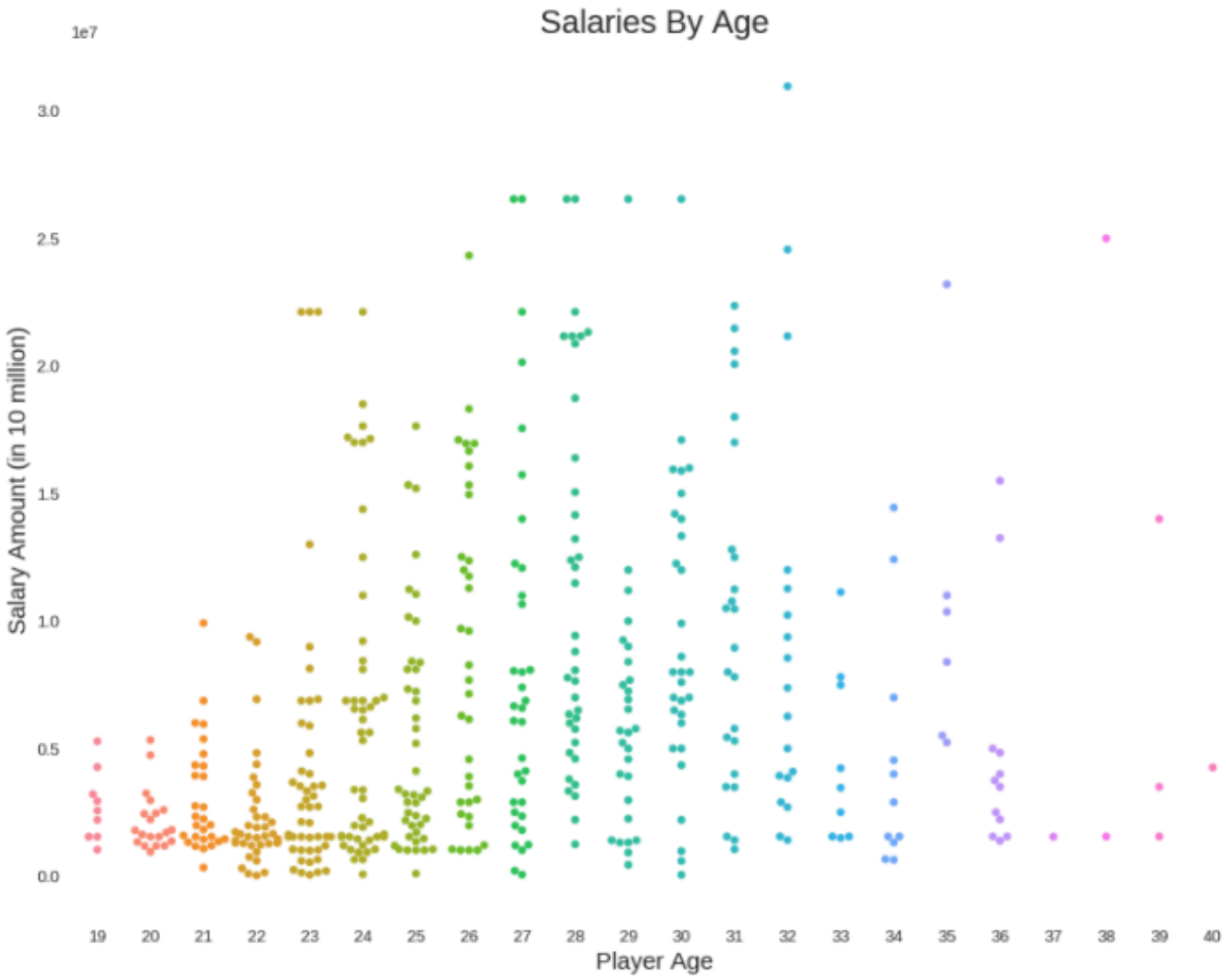
Another advanced statistic similar to that of PER is Win-Shares (WS). As stated in the data dictionary, it is an estimate of the number of wins contributed by a player. And this statistic is more comprehensive than PER so we're going to keep this statistic in the data set. As you can see from the joint plot below, it follows a similar distribution as salaries.



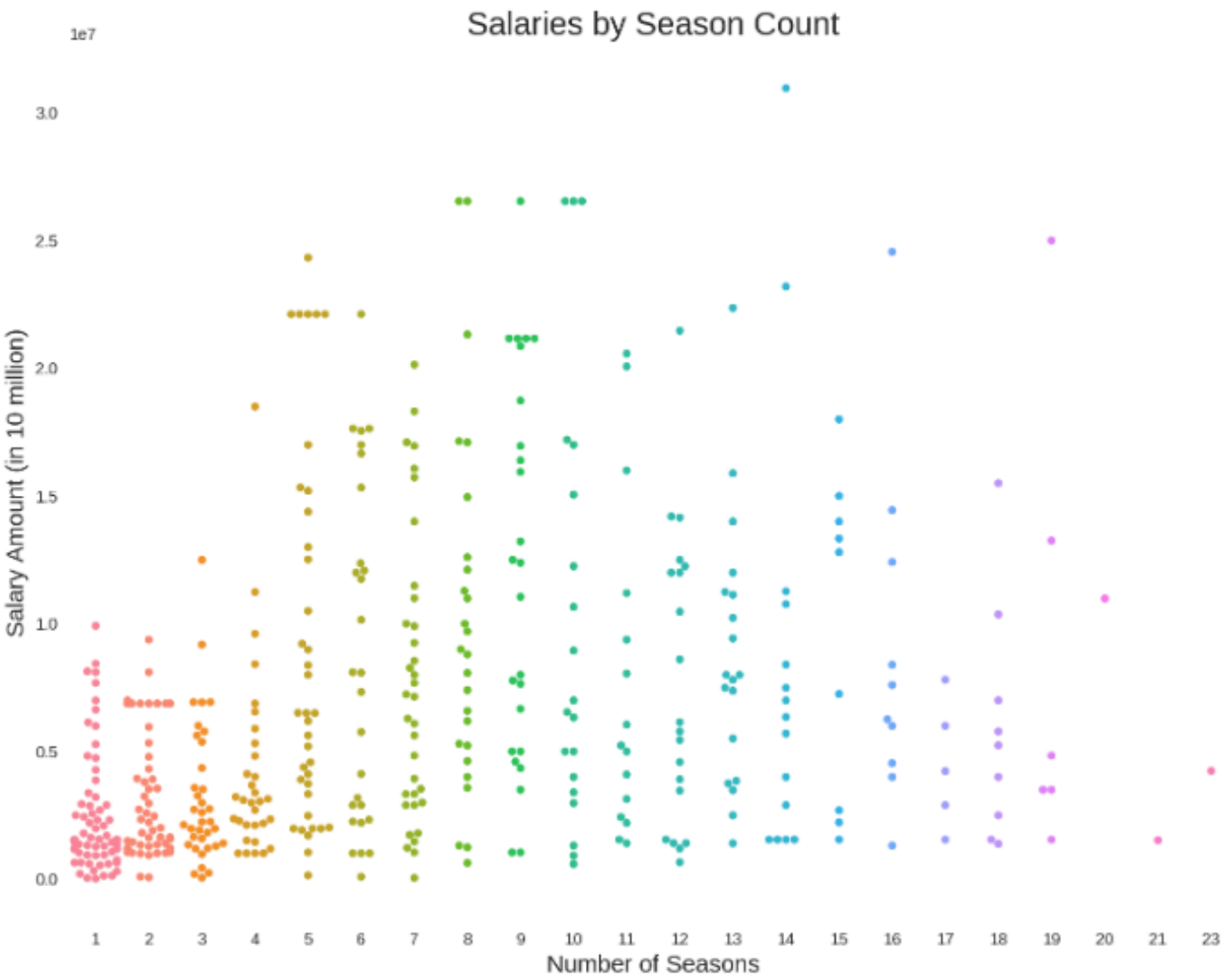
Excluding rookies

Season count was a key indicator of whether a player was in their rookie contract. Age didn't really give any kind of relevant information because there could be someone who is older but in their rookie contract. Season count as more explanatory power as it shows which players are actually in their rookie contracts.

Below is a swarm plot of salaries by age.



The same swarm plot is below only it shows salaries by the amount of seasons in the NBA.



For rookies, their salaries are constrainted by the rookie salary cap which is typically in force for three years. Thus the swarm plot in our EDA notebook showed that there was a large swarm of lower salaries at season counts of 1-3. Therefore, we are going to exclude rookies from this data set. If a rookie or a second year player performed well statistically, they would not be compensated accordingly because they are "locked" into a contract that my not reward them for their excellent play. And the whole premise of our data model is determining salary based on how well a player performs.

Drop these rows

- Any rookies which mean any rows that have a player with a season count lower than 3
    - 172 rookies
    - 486 total players
    - 314 players in our final data set

Drop these fields

- Age
- Season_count

Per Game Statistics

Currently, the data has total numbers for the entire season. For example, the feature "PTS" includes all the points that the player made for all of 2016-2017. However, this might not be entirely reflective of an individual player's performance compared to someone else's because the total number doesn't compensate for any lost statistics due to injuries, suspensions or any other reason for why a player couldn't play. Therefore, the columns that are indicated as totals are going to be divided by the total number of games that they played so that the statistics would be "per game." And then, going to drop the columns that are related to the number of games or minutes that a player played as they aren't statistics related to player performance.

The columns that are percentages aren't affected because the percentage will remain the same whether it's a total or per-game statistic.

Adjust these fields to per game

- FG: Field Goals
- FGA: Field Goal Attempts
- Three_P: 3-Point Field Goals
- Three_Att: 3-Point Field Goal Attempts
- Two_P: 2-Point Field Goals
- Two_Att: 2-Point Field Goal Attempts
- FT: Free Throws
- FTA: Free Throw Attempts
- ORB: Offensive Rebounds
- DRB: Defensive Rebounds
- TRB: Total Rebounds
- AST: Assists
- STL: Steals
- BLK: Blocks
- TOV: Turnovers
- PF: Personal Fouls
- PTS: Points

Drop these fields

- G: Games
- GS: Games Start
- MP: Minutes Played

Dropping columns that have high correlation with others

In regression, "multicollinearity" refers to predictors that are correlated with other predictors. Multicollinearity occurs when the model includes multiple factors that are correlated not just to the response variable, but also to each other. In other words, it results when you have factors that are a bit redundant.

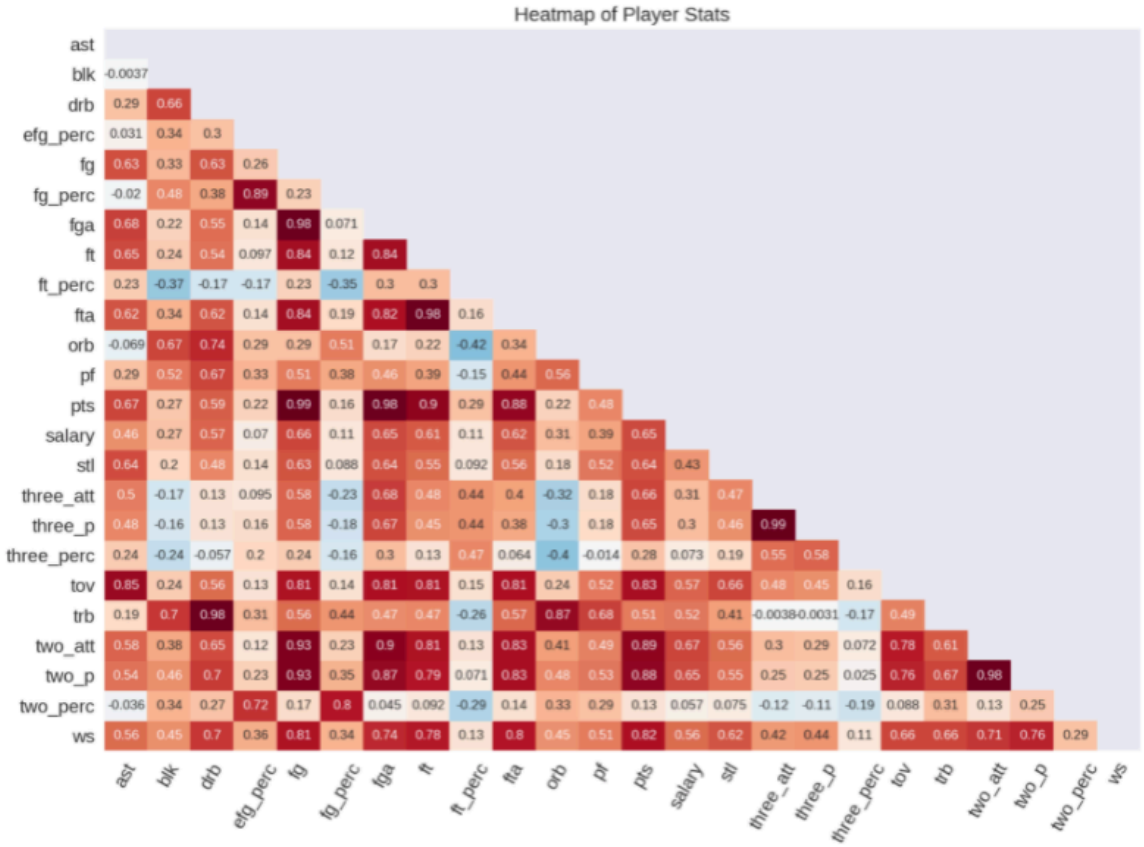
Multicollinearity increases the standard errors of the coefficients. Over-inflating the standard errors, multicollinearity makes some variables statistically insignificant when they should be significant. Without multicollinearity (with lower standard errors), those coefficients might be significant.

A little bit of multicollinearity isn't necessarily a huge problem. But severe multicollinearity is a major problem, because it increases the variance of the regression coefficients. And the more variance they have, the more difficult it is to interpret the coefficients.

One way to measure multicollinearity is the VARIANCE INFLATION FACTOR (or VIF). It is a measure for the increase of the variance of the parameter estimates if an additional variable is added to the linear regression. A high VIF (usually above 10) will indicate that the variable has high multicollinearity. This is one way and it could be found in the stats model package.

But another way to visualize this in a way that all of us could understand and are more familiar with is to look at the correlations. Because higher the correlation between two variables, the more likely we'll run into the issue of multicollinearity. A VIF of 10 or greater is equivalent of correlation of r >= 0.95. If the correlation between two variables were r ≥ 0.95, then most data analysts would say you have problematic collinearity.

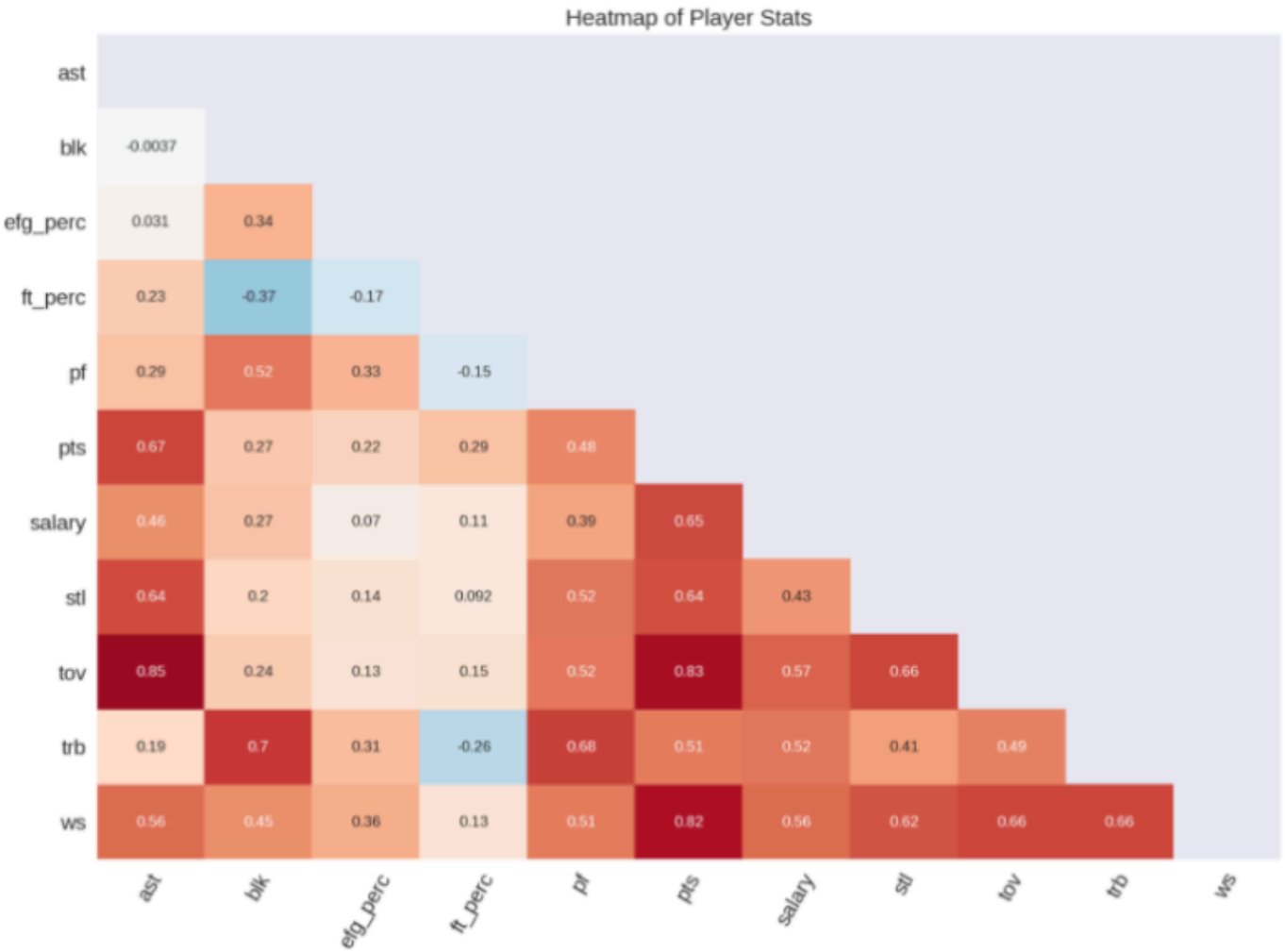
So to check the correlations, I created this heatmap. From the heatmap diagram below, we notice that there are stats columns that are highly (or even directly) correlated with one another. This will affect our model as it'll be unevenly weighting certain statistics and not others because it's appearing more than once in any given column. For example, field goals, field goal attempts and field goal percentage are directly correlated with one another. Or 2 points, 3 points, free throws are all contributing to the statistic points (which is a sum of all of these). So there are certain features that would be getting counted more than once and unevenly favoring those features.



From the heatmap above, we could identify the stats columns that are highly correlated with one another and drop those columns. So we are going to drop the following.

- fg
- fga
- fg_percent
- three_p
- three_att
- three_perc
- two_p
- two_att
- two_perc
- fta
- orb
- drb
- ft

Highly correlated features are removed as shown in the heat map below.



Implementation

For detailed steps, hyperparameter selection and code, please refer to the following notebook – "Part_4_predictive_model.ipynb"

One of the complications that I ran into while implementing the code was trying to get the data scraping to work properly. When scraping data and parsing through all that is collected from a page based on HTML tags, it was sometimes difficult to get only the data points that I needed or wanted. It took me several iterations of working through the data scraping code to get it to the work as I needed it to.

Below, I provide a brief summary of all the algorithms and techniques that I used.

Scaling and Train-Test-Split

Statistics are going to vary in scale as there are some statistics that are just going to be inherently higher values than others. For example, typically players will have more free throws than they do blocks. But the magnitude (or the number of free throws versus blocks) shouldn't determine the predictive power of the end result (salary). So I am going to scale the data so that the statistical variables are standardized by scaling to unit variance.

Linear Model with Regularization

Now we are going to build out a predictive model - starting with linear regression with regularization. Regularization is a method for adding additional constraints or penalty to a model, with the goal of preventing overfitting and improving generalization.

More complex linear model: ElasticNet

The R2 for the linear model was around 42% for the test data. The basic linear regression model may not be the best model given that basketball statistics and a player's performance have many variables and there are other factors that contribute to player's salaries. Building out a more complex model that includes these other factors could greatly improve the accuracy of our prediction.

But after building out this model, the ElasticNet didn't do that much better than the linear model. Actually, the basic linear model seems to have better scores and less error.

Random Forest Regressor

Tried some more models - maybe the linear model isn't the best model for this data set. Tried using a Random Forest which should have a lot of benefits for our data. It's a non-parametric model - so it can predict variables that are non-normally distributed - which is our data as salaries do not follow a normal distribution.

Random Forests are a combination of tree predictors where each tree depends on the values of a random vector sampled independently with the same distribution for all trees in the forest. The basic principle is that a group of "weak learners" can come together to form a "strong learner". Random Forests are a wonderful tool for making predictions considering they do not overfit because of the law of large numbers.

After I implemented this model, the R2 score was lower and error was higher than the linear model. Though salaries follow a non-normal distribution and have a non-linear pattern, the same could not be said probably of the actual set of features (statistics), therfore the lower score than the previous linear models

Extra Trees Regressor

Tried another another tree method - Extra Trees. Seeing if this may produce better results than Random Forest. Looking at the results though, since this has a very high training score but significantly lower test score, the model seems to be overfitting. And so this model wouldn't be good for this data set. Also, the R2 is lower than that of the linear model.

Support Vector Regressor

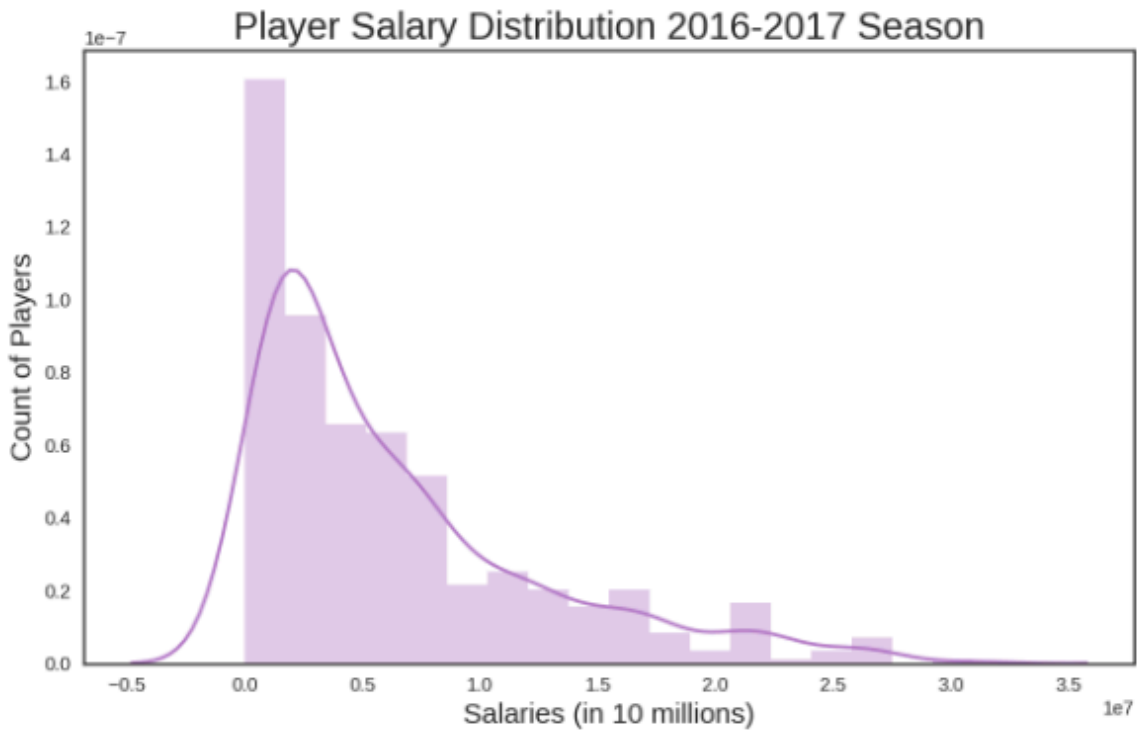
Tried using a Support Vector Regressor to see if this may produce better results. But like the other models, it had a lower R2 score and higher error than the linear model.

Neural Net Regression

Developed a Neural Network model with Keras to see how well that would perform. As Neural Nets are primarily used for classification problems, I wasn't able to find a way to calculate the R2 score. Was able to find a way to use it for regression and calculate mean squared error. But comparing mean squared error to the linear model, it seems that the linear model still performed the best.

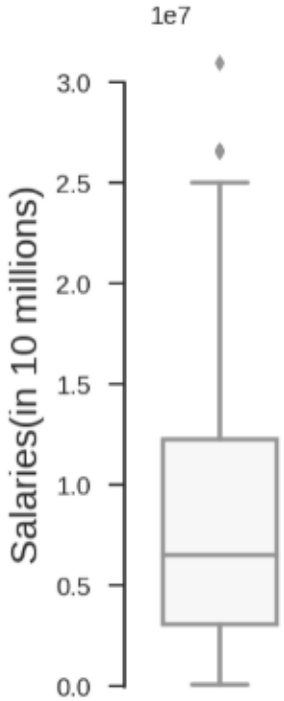
Refinement

In the model, the actual salaries vary quite a bit from the predicted, both on the downside and the upside. What this says is that if this model is a base for what players are worth (which it is not, completely, due to fan appeal and other non-quantitative data), then some players are vastly overpaid while others are vastly underpaid. The histogram below shows the skew of salaries which is characteristic of all teams across the NBA. The plot below is distribution of salaries across the entire league.



With this kind of distribution, it is inevitable that there are outliers on the higher end. To show this, I'm going to take out just one outlier and examine the R2 results. In the 2016-2017 season, the highest paid player was Lebron James with the only salary over $30 million. Outliers like Lebron James definitely affect the model so let's take a look at a linear model without Lebron James in it.

The boxplot below shows the one highest outlier which is Lebron James.



The R2 score of this linear model is 0.468. In other words, 46.8 percent of the variation in salaries of NBA players other than Lebron is explained by the variations in the statistical categories used in this model.

Previously, with Lebron, the R2 score was 0.4185. So taking Lebron out helped our model by increasing explanatory power by about 5 percent.

	linear	linear_lebron_effect
MAE	3.56563e+06	3.50271e+06
MSE	2.12968e+13	1.94824e+13
R2	0.418456	0.468001

With each team in the NBA, there are outliers like Lebron. And there is probably various reasons that they are outliers - but most likely their higher salaries are attributed to factors outside of the on-court performence statistics.

IV. Results

Model Evaluation and Validation

Below is a summary of the scores for the models that I developed.

	linear	elastic_net	random_forest	extra_trees	suppor_vector	neural_net
MAE	3.56563e+06	8.06357e+06	5.76638e+06	5.76638e+06	5.2414e+06	N/A
MSE	2.12968e+13	1.09359e+14	5.84868e+13	5.84868e+13	4.61112e+13	2.71154e+13
R2	0.418456	0.410441	0.391216	0.38823	-0.25914	N/A

Of all the models, it looks like the linear model with lasso regularization performed the best with the best R2 score, as well as the least amount of error.

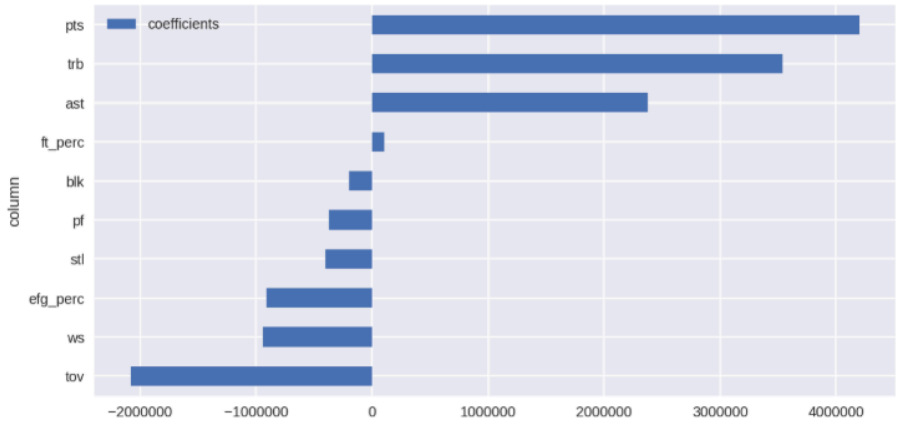
The linear model gave an R2 value of 0.4185. In other words, roughly 42 percent of the variation in salaries is explained in the model by the variation in the statistical fields used.

From this, we could conclude that NBA players' salaries are not fully determined by their on-court performance. About 68% of variation in salaries is explained by other variables that aren't a part of this model. Probably things like how marketable a player is, fan appeal, etc. There is more that goes into a player's salary than just their on-court performance but 42% is still enough that players definitely need to pay attention to their play.

Feature Importances

When all features are on the same scale, the most important features should have the highest coefficients in the model, while features uncorrelated with the target variable should have coefficient values close to zero.

Points per game, total rebounds per game and assists per game proved to be the most statistically significant variables in determining player salary, however, turnovers was also a significant contributor with a high negative coefficient. All of these make sense: the more points, rebounds and assists and fewer turnovers a player has, the higher salary they'll have.



Justification

We could see if a particular feature is statistically significant or not by looking at the p-values. Below I use the statsmodel library to produce an Ordinary Least Squares Regression report to quickly calculate p-values for all our statistical features.

In this case, the null hypothesis would be that there is no relationship between the particular feature and the target variable (salary). So the null hypothesis would assume that on-court-performence statistics are independent of salary. Therefore, if any particular feature has a low p-value, specifically lower than 0.05, then we say that particular feature is statistically significant and we reject the null hypothesis and conclude that there is a relationship between that feature and the target.

Examining the p-values, we see that points, total rebounds, assists, effective field goal percentage, turnovers and win-shares have p-values lower than 0.05.

But of these, the features that had the highest coefficient and the smallest p-values were:

- points
- total rebounds
- assists

So it would benefit a player if they focused their play on improving these particular statistics.

	coef	std err	t	P>|t|	[0.025	0.975]
const	2.68e+06	2.55e+06	1.052	0.294	-2.33e+06	7.69e+06
ast	1.068e+06	3.64e+05	2.937	0.004	3.52e+05	1.78e+06
blk	-1.721e+05	9.42e+05	-0.183	0.855	-2.03e+06	1.68e+06
efg_perc	-8.57e+06	3.68e+06	-2.328	0.021	-1.58e+07	-1.33e+06
ft_perc	1.835e+06	2.23e+06	0.822	0.412	-2.56e+06	6.23e+06
pf	-3.744e+05	6.16e+05	-0.608	0.544	-1.59e+06	8.38e+05
pts	6.905e+05	1.11e+05	6.215	0.000	4.72e+05	9.09e+05
stl	-8.398e+05	9.77e+05	-0.859	0.391	-2.76e+06	1.08e+06
tov	-2.235e+06	1.06e+06	-2.101	0.036	-4.33e+06	-1.41e+05
trb	1.283e+06	2.16e+05	5.934	0.000	8.57e+05	1.71e+06
ws	-4.211e+05	2.03e+05	-2.071	0.039	-8.21e+05	-2.1e+04

The stats model package uses ordinary least squares which is a little different from the linear model with lasso regularization that we built in sklearn. To ensure that we are remaining consistent in the model that we are evaluating, I built out the same report using our linear with lasso regularization model below using numpy, pandas and sklearn. Also, the stats model bases all of its predictions on the entire data set - it fits on the whole data set and then predicts on that same data set. For the purposes of our model, we want to know the p-values of our test data set as that is how we are scoring our model.

The test data is significantly smaller than the entire data set and it contains 79 players, which isn't a whole lot. So the results of this second report may differ from the stats model report

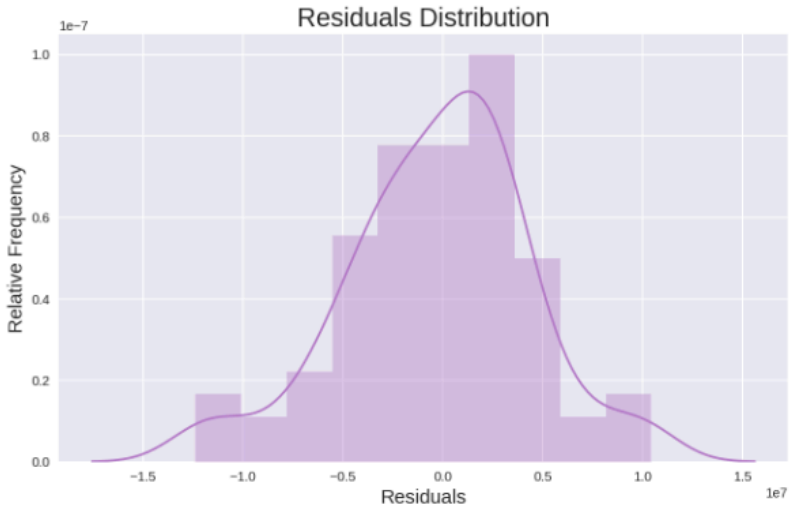
	Coefficients	Standard Errors	t values	Probabilites
Constant	8.135541e+06	8.855644e+06	0.919	0.361
ast	2.374538e+06	8.924847e+05	2.661	0.009
blk	-1.989800e+05	1.917665e+06	-0.104	0.918
efg_perc	-9.111783e+05	1.159228e+07	-0.079	0.938
ft_perc	1.030918e+05	6.697666e+06	0.015	0.988
pf	-3.696629e+05	1.497037e+06	-0.247	0.806
pts	4.199424e+06	2.887020e+05	14.546	0.000
stl	-3.995851e+05	2.013341e+06	-0.198	0.843
tov	-2.077641e+06	2.767409e+06	-0.751	0.455
trb	3.538506e+06	3.986071e+05	8.877	0.000
ws	-9.382274e+05	5.884472e+05	-1.594	0.115

IV. Conclusion

Free-Form Visualization

Residuals are estimates of experimental error obtained by subtracting the observed responses from the predicted responses. The predicted response is calculated from the chosen model, after all the unknown model parameters have been estimated from the experimental data. Carefully looking at residuals can tell us whether our assumptions are reasonable and our choice of model is appropriate.

Residuals can be thought of as elements of variation unexplained by the fitted model. Since this is a form of error, the same general assumptions apply to the group of residuals that we typically use for errors in general: one expects them to be (roughly) normal and (approximately) independently distributed with a mean of 0 and some constant variance. The distribution of residuals for the linear model is plotted below:



From this histogram above, we see that the residuals follow a normal distribution. This incidates that our model is an accurate model in predicting salaries, though there is room for improvement. More than 50% of all the residuals are centered around 0 which means that most of the predictions are getting close to the true salary value with tails on each side. Our data set was relatively small to work with (limited by the number of NBA players for the 2016-2017 season). But given a larger dataset to train on, we will probably see the distribution of residuals with smaller tails on each end.

Reflection

It is possible to create a useful model for predicting the salaries of NBA players based solely on their statistical performance in anyone year. Using scraped data on basketball players' on-court statistics, I created a model which stands to explain how much players get paid according to their performance.

One thing about this project that I found very interesting is doing all the back-end work of setting up my environments in Docker. It was really neat to build it all out and to see it come together from data that I scraped and collected. What was also interesting about this project was that I learned that the problem that I was working on was more complex than it initially seemed. Originally, I had thought that it would be simple to just take several different statistical features of a player's performance and have that be the predictors to their salary. It seemed like a simple problem initially. However, as I dug into the data and thought about all the

complexities and factors that needed to be accounted for in determining a player's salary, I realized that there is much more to the problem than I had originally anticipated, thus leading to so many steps in my data pre-processing. This project was validation for me of the importance of exploratory data analysis before any kind of further data modelling. Such analysis and visualizing of the data is required before any kind of modelling can happen so that we are able to adequately understand the problem. I found this part of the project to be very interesting – as well as difficult as it presented some tough decisions to make about the data (such as if it would be better to leave certain data points in or out).

Also, another reason we created this model was to identify the variables that were most likely to contribute to NBA player salaries. We found that points and rebounds were the two main contributors to player salary. In other words, not only is it beneficial that a player have a high scoring average, but he must also be good at getting boards on both offense and defense. Assists and turnovers were also significant. Assists allows players to contribute to the overall points of the team and so it looks like teams are willing to pay a premium on assists also. And players who have an excessive amount of turnovers give their opponents more opportunities to score and thus becomes a liability to the team, therefore we saw a high negative coefficient for turnovers.

In all, players do not get paid according to their performance on the court only. A variety of other factors come into play, such as fan appeal and who represents them as an agent. Some team's management bodies tend to pay more, too. With that said, though, one can still produce a reasonably accurate statistical model predicting salary based solely on on-the-court performance, measured quantitatively.

Improvement

The main limitation to developing this application is that only quantitative data was used. There are other intangibles that we can't account for in this model such as fan appeal, player's prone to injuries, or player's personality, leadership and impact on team morale. However, the model does highlight some trends and clearly shows that data can be used to determine NBA salaries. But these kind's of limitations are something that anyone faces in the normal job market. Applications like Glassdoor and Indeed predict salaries based on hard skills that are determined to be in greater demand. However, their models are limited too in that they are not able to capture intangible factors like a person's soft skills, leadership ability, team fit, business acumen, etc. But such applications are still useful tools that both employers and job-seekers utilize as it gives useful information on expected salaries given a certain type of skill-set. Likewise, Open Court functions as a similar tool for NBA team owners, athletes and their agents.

Another limitation is that this model doesn't account for players who signed long-term contracts well before the 2016-2017 season. Therefore, their salary rate is determined by a prior season of play and isn't fully captured by just the 2016-2017 season alone. For the purposes of this model and project, I am attempted to just predict the 2016-2017 salaries based soley on basketball statistics from that season.

There is much more I wanted to do with this project but didn't get a chance to.

- Build a web application where players can get an estimate for salary by inputting their stats.
- Develop a model for rookie salaries - project "DraftMe"
    - Rookies' salaries are determined by their position in the draft. And so we would need to develop a model that predicts where they'll be picked in the draft based on their performance before the NBA (which is their college statistics). I started the process of scraping college statistics but haven't build out a model yet to predict draft position.

- Once we know draft position, then there are already salary estimates for each position in the draft. And so rookies, even in their first 3 seasons in the NBA, won't be compensated based on their performance in the NBA but more prior to entering professional basketball.