Alvaro Hu

SpringBoard

October 8, 2018

Using Machine Learning to Predict the Salary of NBA Players

I. The Proposal

The purpose of my project is to give a salary, a finite and concrete number, to players in the NBA based solely upon their stats. I will also be working on figuring out which game statistics allow for not only the biggest increase in salary, but also the ones that are *most likely* to give a player an increase in their salary. The difference here relies on the R^2 value, meaning how strong a correlation exists between the two variables. For example, and increase in a certain feature by 1 might only increase the salary by $50,000, but if it has a strong R^2 value, then it has a strong likelihood of that increase actually occurring. This is useful for a wide variety of reasons, but mainly for these two; **1.)** that General Managers (GMs) are informed with an actual number for how much a player *should* be worth based on their stats so that they can make decisions on players, but also **2.)** to give the players an idea on what to improve in their game in order to get either as significant or as sure an increase in salary as possible.

To do this, I went online onto basketball-reference.com and downloaded three datasets. One data set with all of the players' names along with their salaries for the upcoming season, another with all of the players' names with their common per-game stats, and a third dataset to retrieve information on some more advanced statistics, like Win Shares, Player Efficiency Rating, and Assist Percentage. The reason why I used next year's salaries and last year's statistics is because it is common practice to sign extensions and make deals over the offseason based on a player's previous season stats for the next year. I merged these three datasets together using the player's name as the axis, and got a data set with far too many features. I used the .drop() feature in pandas to drop a majority of them. I still kept quite a few

features, but only to see if they would have an effect on the salary in a later stage of the project. Some of the important features that I kept in my dataset were points per game (PS/G), Win Shares (WS), Assists (AST), Games Started (GS), and Rebounds (REB). There is a total of 31 columns, of which you can find the abbreviation and what it stands for in a table in the code.

II. The Wrangling

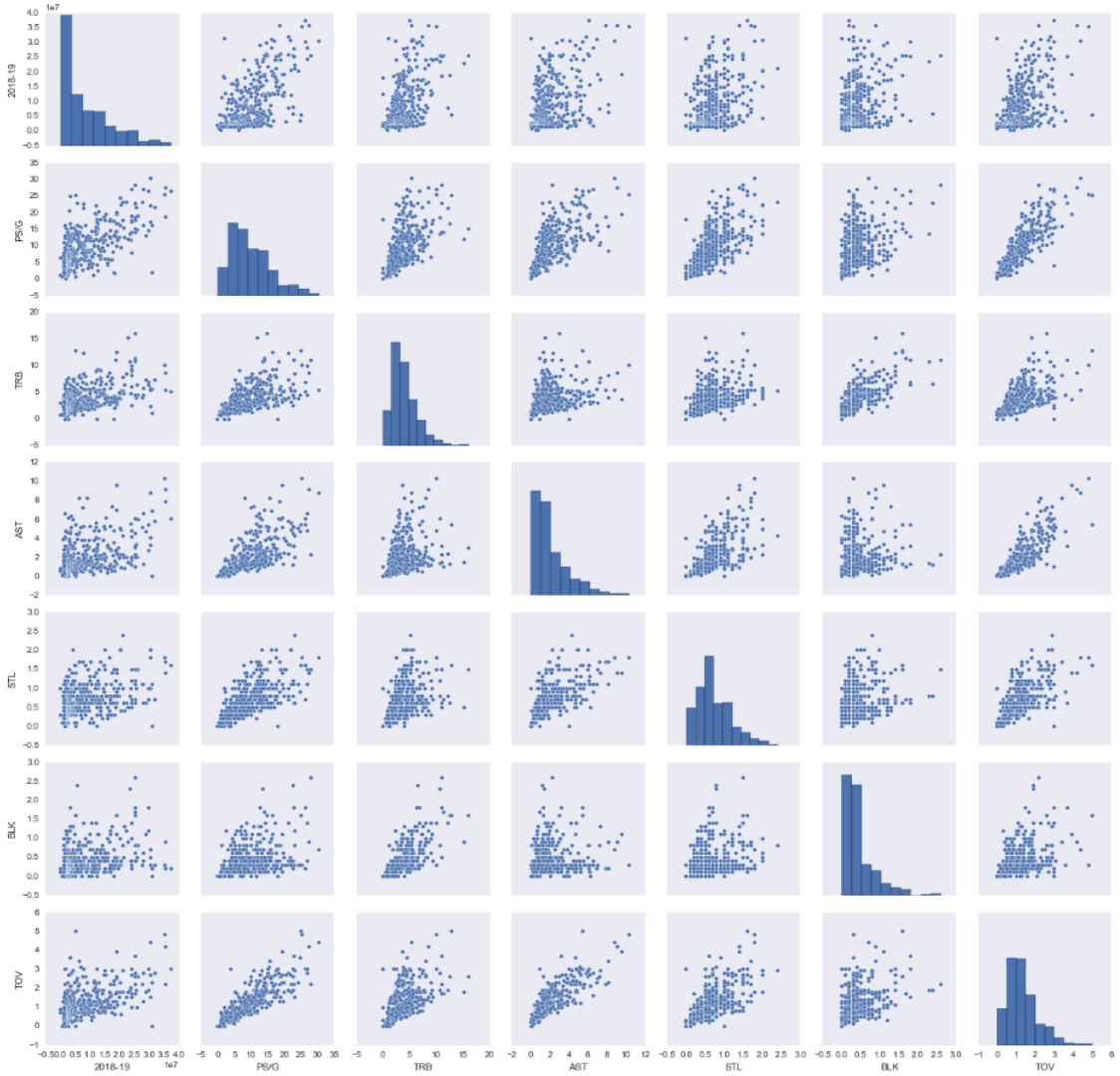
When I first read the data into my project, there were a lot of odd things in the data that I had to deal with. For one, the 'Name' column in all 3 of the datasets had a string attached to it so that the player's name would be something like "Stephen Curry/stephencurry". So I had to split that column by the "/" figure before merging all of the data. Also, there were also many players who showed up multiple times in the **per-game** dataset, since they played on multiple teams. To get around this, I checked to see how many players appeared with "TOT" in the Team column, which indicates that they played on multiple teams, and were given a Total column. The number I got was 59 players. I then counted up how many times these players showed up in the dataset, and found that these 59 names appeared 183 times, which makes sense because each of these players will be in at least 2 other rows. I used the .drop_duplicates() attribute and kept the topmost column of the occurrences, since that column was the TOT row. I then merged the **salary** dataset with this one, but since the salary dataset only included one row per player with a column for the team that each player *ended* the season with, I was able to drop the Team column from the per-game dataset, effectively throwing out all of the "TOT" entries and replacing them with their current team.

From here on, the rest of the data wrangling was fairly straightforward. The only thing that I had to do was get rid of the "$" sign in the salary column and turn them all into integers, so that I can order them correctly. They were floats at first, which meant that the numbers were read in an alphanumeric fashion. For example, 10,000 would be placed ahead of 999,999 because "1" technically came before "9" in the alphabet. I also decided to drop players who did

not play last season, as it will be impossible to predict their salary from games they never played, and also players without a salary for the upcoming year, as I will be unable to use them in my regression to predict a player's salary. I then output my dataframe as a csv called 'Total2018.csv'.

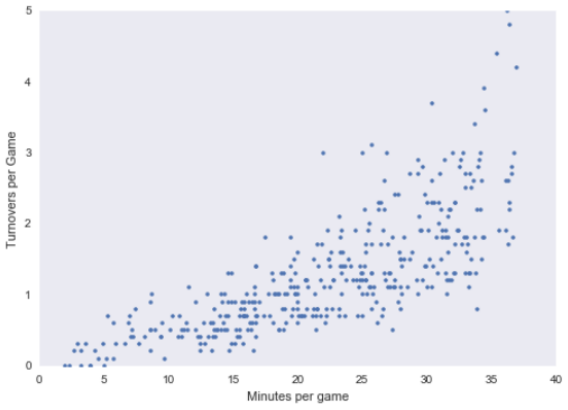
III. The Visual Exploration:

After I wrangled all of the data and came up with a dataset with all of the features I wanted to use, I decided to visually explore the data to see if, right off the bat, I might be able to see some patterns. The first thing that I did was to make a few pair plots. I made one with the entire-season stats, like games played, win shares, etc., and I made one with the per game stats. Below is the pair plot of the per-game stats.



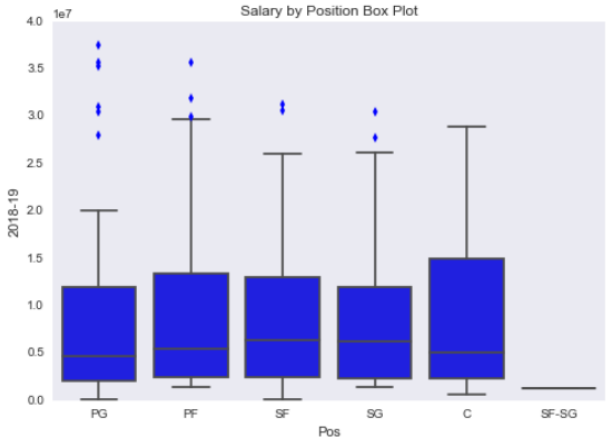
One interesting thing to take note of from this pair plot is that almost every statistical category has a fairly linear relationship with every other statistical category. By this, I mean that in every box on the grid, increasing the value of the x-axis also leads to an increase in the y-axis, no matter what the axes are. This may lead to a low number of principal components, which we will confirm later in the project. The other thing to take note of is the non-normality of the distributions of these stats. The other pair plot shows that many of those stats are correlated too, but not quite as much. From these two pair plots, I found that Win Shares, Games Started, Points per Game, and turnovers had the strongest correlations with salary. I based this off of the Pearson Correlation Coefficient values of 0.59, 0.57, 0.64, and 0.52, respectively. This Correlation Coefficient, R, is the basis of the R^2 value that I decided would be my standard would be for judging how likely an increase in one of these features would increase the salary. It was also interesting to note that Rebounds, Points, and Assists are all strongly intercorrelated with one another, which can be explained by a change in the way basketball is played these days. Awhile ago, people could get by with being solely a scorer, or a pure rebounder or passer, but these days, in order to be a competitive player, you need to be good at all three of these things.

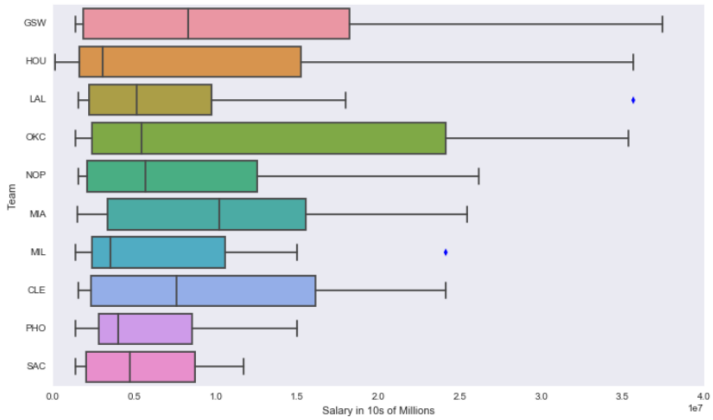
I was intrigued by the strong positive correlation between turnovers and salary. Wouldn't it be that the more you lose the ball, the worse of a player you are which would in turn result in less pay? I did a quick scatter plot of turnovers per game and minutes per game and found that these two were also strongly correlated. My conclusion for this was that, if you are a good player, you will play and have the ball more, and if you have the ball more, there are more chances for you to lose the ball. So, if a basketball player wants to earn more money, he probably shouldn't



purposely lose the ball just because there is a correlation between turnovers and salary, and instead work hard and play more minutes.

The last thing that I did during this stage of analysis was to see if there was a significant difference in average salary between the positions that players played or between the teams they played for. To do this, I made two sets of boxplots; one for the positions and one for different teams.





As one can tell from the plots, although there are many outliers in the point guard position, all five positions earn about the same median salary. As for different teams, I selected 10 teams to show on the boxplot, so a third of the total, and it is quite evident that some of the highest paid teams have much higher median salaries, but also higher salaries across the board. This may be caused by the NBA's soft salary cap, which enables teams and players to exceed the salary cap if they sign a multi-year deal or extension in order to promote "hometown loyalty". These large differences, though they seem to be significant, can easily be swayed by 1 or 2 players with a large salary, as each team only has 12 to 15 players on the roster. In the next section, I will check to see if these differences and correlations are statistically significant at a 95% confidence interval.

IV. The Statistical Exploration

I decided to do some tests to see if I can take these correlations and differences, or lack thereof, confidently. In order to test plausibility of the correlation coefficients between the different features. My hypothesis test for each of these read like this: The null hypothesis is that the Pearson correlation coefficient for the population is equal to the one I obtained from the data, and the alternative hypothesis is that it is not equal to the one I obtained from the data. In order to check this, I shuffled up the data, keeping the pairs together, and split the pairs into 2 sets; one with 90% of the data, and one with the other 10%. From there, I split the pairs in the 10% of the data, and shuffled one of the features before repairing them back up. This would make sure to un-correlate that portion of the data. I then concatenate the shuffled 10% back on to the 90% and took the pearson correlation coefficient once more to compare it to the original one. I do this 10,000 times to see the likelihood of achieving a correlation as strong as the one from the data. I yielded p-values of 0.004, 0.015, 0.009, and 0.006 for points per game, win shares, games started, and minutes per game with salary, meaning for all 4 of these features, the empirical value of the correlation coefficient was accurate at a 95% confidence level. Because the distributions of these features were not normal, I checked my answers using the Mann-Whitney U test, a common test for non-normal distributions. I achieved incredibly low p-values from these tests on the order of -125.

The next thing that I did was check to see if the differences between the means of each team could be taken as truth, and not just with a grain of salt. My null hypothesis was that there would be no difference, while my alternative hypothesis was that there was a difference. In mathematical format, it looks like this:

$$H_0: \mu_{Team\ 1} = \mu_{Team\ 2}$$

$$H_a: \mu_{Team\ 1} \neq \mu_{Team\ 2}$$

I started with the two highest paid teams on average; the Golden State Warriors and the Oklahoma City Thunder. I found the difference in their means and did a bootstrapping test to

see how often this difference in means might come up. To do this I took the salaries from both teams, put them into one array, shuffled them, and redistributed the salaries to the teams. Doing this simulated the idea that if there truly was no difference between the two teams' salaries, then it would not matter that they were coming from the same set. I then check to see how likely it is, out of all of the times I shuffled the data, that the difference between the two means was as extreme as it was in my empirical data. The fraction of those that are as extreme becomes my p-value. For this particular test, I received a p-value of 0.466, which is higher than the alpha-level of 0.05 that I had set as my threshold, meaning I could not conclude that there was a difference in mean salary between the two highest paid teams. However, running the same test with the Warriors and the Kings, the highest and lowest paid teams, I yielded a p-value of 0.026, meaning that I **can** conclude that there is a difference in pay between the highest paid team and the lowest paid team, confirming what I had seen in my initial visual analysis. Meaning, though there might not be a difference in every teams' salaries, there might be a difference when looking at specific teams.

The final thing I did was principal component analysis of the dataset, and found that when all things considered, ***there is only 1 principal component.*** I did not find this too surprising, since I saw during my EDA that many of the features had strong correlations with one another. Armed with this knowledge, I set out to perform an Ordinary Least Squares (OLS) Fit to obtain some p-values. I took two different OLS's; one with the minimum number of features, and one by removing the feature with the highest p-value until they were all reasonably low. For my first OLS, I looked at the 4 stat categories that I deemed most influential on the salary from my EDA and found that points per game and games played were the only two that weren't strongly correlated with one another. From this, I got a model with an R^2 of 0.434, an F-stat of 143.5, and both variables having low p-values < 0.05. The other OLS gave me a regression model with games played, assists per game, points per game, win shares, and assists percentage. This second model gave me an R^2 value of 0.504, so better than the first

one, but not as high of an F-statistic of 75.38. The coefficients for the five of these features were -$95,000, $2,481,000, $458,000, $1,112,000, and -$428,000, respectively. What these coefficients means is that, when all other things held equal, a change in that variable by 1 will change the salary by that amount. The three things I found important from this information is that assists per game had the largest coefficient, point per game had a relatively low one, and games played had a negative coefficient. The large coefficient for assists per game makes sense because the average amount of assists in the NBA is low compared to other stats, and it is hard to increase the number of assists a player makes when the most assists a player makes is around 13 per game. The opposite is true for why the points per game coefficient is low. The highest points per game was over 30, and increasing by 1 is not that impactful. The negative value of the coefficient for games played may because if a player plays more, you'd expect them to be more productive, not just stay the same. The F-stat tells me how statistically significant my model is, and both of these numbers are actually quite high, meaning both models can be classified as significant.

V. Machine Learning

For this final section, I ran many different regression models to find the best R^2 value possible. I started out with a univariate OLS using only points per game, the highest correlated feature with salary. I achieved an R^2 value of 0.405, which if you look at the correlation coefficient I found earlier in section 3, is that value squared. I then moved onto two more OLS regressions, one with 3 features strongly correlated with salary, and the other with the 5 features I ended up with in the last section. These each resulted in increases of 0.05 to the R^2 value, with the 5 feature-OLS yielding a 0.504 score. Judging by the type of dataset that I have, a score of 0.504 is not bad, considering that 1 is completely correlated. So just using this model may prove a decent predictor of a player's salary. I must also consider that with these 3 OLS models, I have used 100% of the data in my training set, and have not yet tried to predict the salaries of unseen data.

In order to test my regression model with unseen data, I split my dataset into a training set consisting of 70% of the data, and a testing set using the remaining 30%. After training the OLS with 5 variables with the training set, I then predicted the salaries of the test set and took the correlation between the actual values of the salaries and the predicted values of the salaries. I then squared this to get the R^2, which is what Linear Regression models typically use to score their accuracy. For this particular testing set, the value decreased from using the entire set as training and became 0.474.

I moved on to using a Linear Regression model from the sklearn package. With this model I used the same 5 variables used in the OLS and received a score of 0.507. This may seem like an improvement, but after cross validation, the score wound up being 0.45. One thing to note is that when I initially ran my cross validation, I was receiving incredibly low scores. Scores lower than I even thought possible (in the negatives). I realized that this was because my dataset was ordered by salary, and certain sections of the data, when isolated, either have a negative or no correlation at all with the salary. For example, there are 25 players with the same minimum 1-year salary, no matter what their stats are. The way the cross_val_score module from the sklearn package works is that it splits the data into a number of slices based on the user's input, and uses each slice as a testing set. But the way it slices the data is chronological and does not do any shuffling, so it picks up on these spots of the data with poor correlations. In order to remedy this, I pre-shuffled the data before feeding it into the cross validator.

My next two regressions are Lasso and Ridge regressions. These two operate differently from Simple Linear regression and OLS by implementing a function that penalizes features with poor correlations to the target by reducing their coefficients either to 0 like in Lasso Regression, or down to a small value like in Ridge. Because of this, I used all of the features for Lasso and Ridge regression and achieved cross-validated scores of 0.44 and 0.49 respectively. Based on these results, Ridge Regression seems to be the best model to use moving forward. I noticed that with this model, I was receiving predictions of negative value, which is not a good sign for

the player. I believe that this is caused by the presence of a minimum salary, such that even if a player is playing much worse than another player being paid the minimum salary, they will still be paid the same amount. To fix this, I took the log of the salary column and used that as the target variable, and after predicting this column, I transformed it back into the salary by taking the exponential. This gave me my best score of around 0.58, even with cross validation. Most importantly though, all predicted values remained positive. This is good for the players, because it would be quite heartbreaking to hear that you are not even worth any money at all (even though, statistically, that might actually be the case). After trying out all of these various models with different meta parameters, cross validation, and with various features, I have decided to conclude that the Ridge Regression with a log transformation for the salary is the best model for me to predict players' salaries.

VI. Concluding Thoughts and Revelations

There are a few main things to take away from these findings. The first is that because of the way basketball is played these days, many of the features in this dataset are correlated with one another, leading to only 1 principal component. Some other things are that the correlations between points, rebounds, win shares, and minutes per game with salary are the numbers that I found in my initial EDA at a 95% confidence interval. This means that the statistics that allow for the most guaranteed increase in salary are points per game, games started and win shares. I also found out that even though points per game has a strong correlation with salary, increasing points per game by 1 will not result in that large of an increase in salary, so you would have to increase your points per game by more than just 1 if you wanted to see results in your salary. On the other hand, changing your assists may not be as guaranteed to increase your salary, but if you do increase your assists per game by 1, your salary can potentially be increased by over $2,000,000.

One small revelation I came to was that that at a certain level, there is a difference between the average salaries between teams, using teams as a means to predict a player's salary is not feasible. The same can be said about what position the player plays. I did get a model that explained almost 60% of the variance in the salary of NBA players. So, although my model may not be able to predict exactly what a player should be paid, I can come up with a good ballpark number that the GM or the player can use when deciding on business decisions. The lack of predictive power in my model most likely arises from non-statistical factors that go into the salary, such as popularity, previous injuries, and players who willingly take pay cuts in order to stay on a team that has a better chance of making it to the finals.