Predicting NBA Player Salaries

Jemmy Zhou

Spring 2020

Abstract

1 Introduction

The average salary for an NBA player in the 2018-2019 season was $6,388,007. Stephen Curry, star point guard of the Golden State Warriors, was the highest paid player during that season and received $37.46 million. In this paper, I analyze data from two data sets and develop a machine learning classification model that is capable of predicting the annual salary of an NBA player in any given season. I use the provided `Player Box Score` data set and ESPN's `Salary` data set.

2 Data Sets and Data Pre-processing

2.1 Salary Data Set Collection

To predict an NBA player's salary, we must obtain the salary of every NBA player per year. Since `Player Box Score` only spans the 2013-2018 seasons, we will first focus on obtaining the salary figures for those specific seasons.

ESPN keeps track of every player's salary in a given season. The first page of players' salaries in the 2012-2013 season can be found here [1]. I wrote a web scraping script to iterate through the six seasons and saved the data set into `salaries.csv`. The original data set on ESPN's site doesn't include the season. Since I need the annual salary for each player, I appended the season onto each row of the data set as I was saving the data to `salaries.csv`.

2.2 Cleaning the Salaries Data Set

ESPN includes the column names every ten rows in the original data set. While these rows improve clarity of the data for ESPN's purposes, we need to remove them for our model. I started cleaning the data by filtering out any rows with `"RK"` in the `RK` (Rank) column. The rank column orders the players from highest annual salary to lowest. This serves no purpose for our data set, so I dropped the column after this step.

ESPN identifies player names as `"First Name Last Name, Position"`. Using regex, I extracted the first name and last name and put them in separate columns. Originally, I intended to include the position in the

data set and One-Hot Encode the feature. However, after some research, I realized the average pay across positions is relatively evenly distributed [2].

The salary for each player is currently stored as a string. In order to perform classification, I extracted the numbers from each string and converted the type to integers. I also renamed the `"2013"` column to `"season"`.

While a player's first name and last name is generally enough to match players when joining the two tables, there's a possibility of two or more players with the same name. In this case, we must join on the condition that the first name, last name, and team matches. We assume that there are no two players on the same team with the same name [3].

The final column we need to clean is `team`. Currently, the `Salary` data set contains the full team name, while the `Player Box Score` uses the team abbreviation. In order to be able to compare them when joining, we'll convert each team name in the `"team"` column for the `Salary` data set to the respective team abbreviation. I used the abbreviations provided in `playerBoxScore.pdf`.

Prior to the 2013 season, the New Orleans Hornets changed their team name to the New Orleans Pelicans [4]. Before the 2014 season, the Charlotte Bobcats changed their team name to the Charlotte Hornets [5]. After checking the `Player Box Score` data set, I confirmed that they used the same abbreviations for both teams before and after the name change. I defined a dictionary of the team name to the abbreviations and changed the `"team"` column of the `Salary` data set accordingly.

I realized some players still had `NaN` as their `"team"`. Upon further research, I discovered that ESPN kept track of ex-NBA players who played in foreign teams. I dropped these rows as they are unrelated to our problem.

2.3 Cleaning the Player Box Score Data Set

The `Player Box Score` data set comes with many features, most having no correlation with players' annual salaries. The features I dropped are: `gmTime`, `teamConf`, `teamLoc`, `teamRslt`, `teamDayOff`, `offLNm1`, `offFNm1` , `offLNm2`, `offFNm2`, `offLNm3`, `offFNm3`, `PlayDispNm`, `playPos`, `playHeight`, `playWeight`, `playBDate`, `opptConf`, `opptDiv`, `opptLoc`, `opptRslt` `opptDayOff`.

Since many teams have failed to reach the playoffs during the 2013 through 2018 seasons, we will only look at players' stats from regular season games. We also exclude pre-season games since many teams use different personnel for those games.

Currently, every sample point contains the game date. We're interested in season averages so we must convert these dates into seasons. To do this, I used regex to extract the month and year from the game date string. Since each NBA regular season ranges from October to April, we can divide up the games with the following rule. **"If the game is played after July of year X, it will belong to the $X+1$ season. Otherwise, the game is part of the X season."** For example, games played in October 2013 is part of the 2014 season. Games played in February 2013 is part of the 2013 season. After the transformation, we only keep the `"season"` column and drop all the intermediate columns.

Since starters on a team are almost always paid higher than bench players, we'll include this as part of our data set. However, before we can use them, we must One-Hot Encode the categorical data.

Finally, we group by `playFNm`, `playLNm`, `teamAbbr`, and `season`. Taking the average gives us each players' statistics per season.

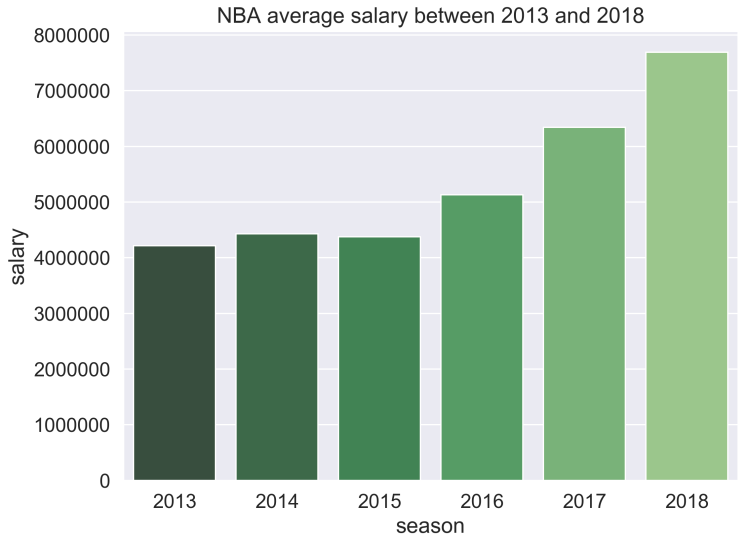

Figure 1: NBA average salary of players between the 2013 and 2018 seasons

2.4 Combining the Two Data Sets

To combine the two data sets, we perform an inner merge on the player's first name, player's last name, team, and season. I chose to use inner merge because there's no good way to process sample points with Null values.

During the merge, I realised the `Player Box Score` data set identifies my favorite player as Wardell Curry, while ESPN's `Salary` data set identifies him as Stephen Curry. I changed his first name in the `Player Box Score` data set to Stephen, so the inner merge would properly merge his statistics. We exclude all other people with different names in the data sets, as there is no efficient way to identify them and fix the data sets accordingly.

2.5 Normalization

Now that we have our data set, we can visualize some of the data. First, we'll plot the average annual salary between the 2013 and 2018 seasons. As we can see in Figure 1, the average annual salary has been increasing over the years. To account for the increase, we need to normalize each player's annual salary. Rather than normalizing by dividing the `salary` column by each year's average salary, we will normalize by dividing the `salary` column by each year's NBA salary cap limit.

The salary cap limit is the total amount of money an NBA team is allowed to spend on all players. This limit is increased almost every year. This allows NBA teams to increase players' salaries (since NBA teams are limited to 15 players on their active roster, they cannot spend the money on increasing the number of players). A player's salary divided by the salary cap limit is a better indicator of their skill level, which should have better correlation with the features than if we divided by the annual average salary.

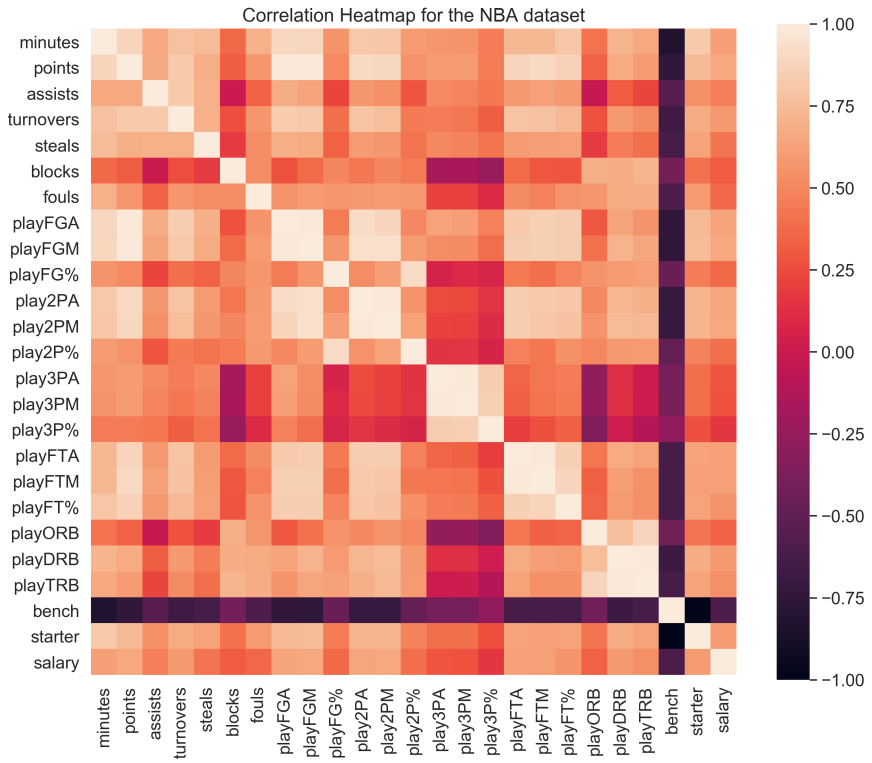

Figure 2: Correlation heat map of the NBA data set

3 Models

3.1 Choosing a Model

The problem of estimating an NBA player's annual salary is a machine learning classification problem. Regression is a natural choice for classification problems because it's mainly used to estimate relationships between independent features and the outcome. While logistic regression is often used to two-class classification, (i.e. win/lose, pass/fail), (multiple) linear regression removes the binary limitations of logistic regression.

3.2 Feature Engineering

In order to choose the features we want, we need to first visualize the impact each feature has on the `salary` column. I computed the correlation matrix and plotted a correlation heat map to visualize the correlation between each feature. As we can see in Figure 2, certain features like `playFGM`, `points`, `play2PM` etc. have higher correlation with the `salary` variable than `play3PA`, `play3PM`, `play3P%`, etc.

To further visualize the linear relationship between features and `salary`, I took the four features with the highest correlation and plotted it against `salary`. We can identify the linear relationship between the variables from the scatter plots in Figure 3

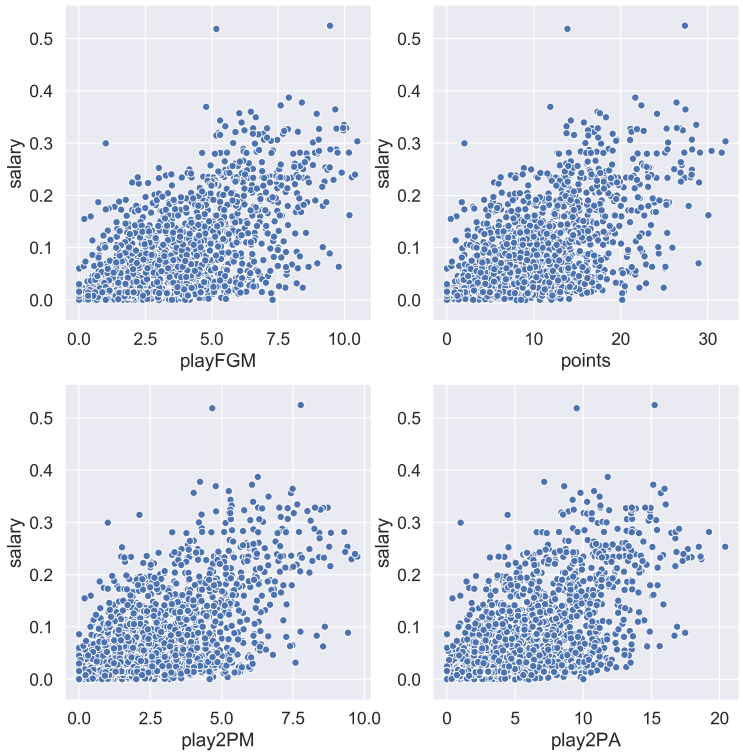

Figure 3: Linear relationships between `salary` and four features with the highest correlation.

3.3 10-feature Linear Regression

Our first model uses the 10 best features, in terms of correlation. We will be using RMSE as our error function. I split the data into train, validation, and test sets using `sklearn.model_selection.train_test_split`. The test set was 10% of the entire data set, while the validation set was 20% of the training set. After fitting the model on the training set and verifying it against the validation and test sets, the model predicted with the following error rates:

Training error: 0.0554
Validation error: 0.0602
Test error: 0.0583

3.4 24-feature Linear Regression

Using only 10-features already created a model that could predict with roughly 94.17% accuracy an NBA player's annual salary. Hoping to reduce the error rate, I decided to use every feature in the next model. Using the same train, validation, and test sets split, I obtained the following results:

Training error: 0.0532
Validation error: 0.0584
Test error: 0.0568

Figure 4: Relationship between actual salary vs. predicted salary of the test set.

3.5 Linear Regression with Cross Validation

To further improve feature selection, I decided to order the features by their correlation value with `salary`. Then, I used cross validation to selection the first n features that produced the lowest. This process selected every model resulted in a cross-validation error of 0.0548. When using the model to predict training and test sets, I obtained the following results:

Training error: 0.0543
Test error: 0.0554

4 Analysis and Conclusion

The best model was the linear regression model with cross validation. It lowered the error rate by 0.0029 when compared against the first model we tried. In the end, we were able to predict NBA player annual salaries with 94.46% accuracy. As we can see in Figure 4, the relationship between the actual sample points and predicted sample points is fairly close to the red line. The red line symbolizes points where the actual sample point and the predicted sample points are the same.

In Figure 5, we can further conclude that the linear model fits the data set fairly well, we there are no obvious patterns or shapes in the residual plot. While there is a fair bit of clustering near the left side of the residual plot, this makes sense because most NBA players' salaries are in the lower range.

In addressing the seven specific questions,

(i) Some interesting features I came across was `play3PM` and `playORB`. I was surprised 3-point statistics had such low correlation with `salary` as they have become the increasingly popular way to score points in the modern NBA. I was also surprised offensive rebounds was much lower than defensive rebounds, as I expected them to be about the same.

Figure 5: Residual plot of the predicted values of the test set.

(ii) As we saw in the final model, every feature I kept turned out to be useful for the final model.

(iii) There were several challenges when cleaning the data. The first challenge was matching players in the two data sets. In the end, I chose to match players with first name, last name, and team, and decided to drop any rows that couldn't match all three properties. This posed an issue when some players had preferred names other than their first name, i.e. Stephen Curry.

(iv) I assumed that there were no two players on the same team that had the same name. There was no efficient way to confirm this, and the closest result on the internet came from Reddit.

(v) I did not have any ethical dilemmas with this data set as all data was made public.

(vi) If there was a data set that contained all players' birthdays, then we could combine that with the salaries table and use that as a join condition instead of the team. It is far less likely that two players with identical names were born on the same day.

(vii) Since NBA salary data is public information as determined by the National Basketball Players Association, and game statistics are also made public by the NBA, there is no ethical concerns when researching this problem. Potential uses for these models are for players in the NBA to gauge their own salary worth based off their seasonal statistics.

5 Future Work

A concern that I had during the process was both the `salary` data set and `Player Box Score` data set had roughly 3,000 sample points. After merging the two data sets, I lost roughly 1,000 sample points. A possible source for this loss was mismatch in player names between the two data sets (as evidenced by Stephen Curry vs. Wardell Curry). In the future, I would find better ways to join the two tables and try to minimize the lost sample points.

References

[1] ESPN: NBA Player Salaries - 2012-2013
http://www.espn.com/nba/salaries/_/year/2013/page/1

[2] NBA Positional Payrolls
https://www.spotrac.com/nba/positional/

[3] Reddit
https://www.reddit.com/r/nba/comments/52pz9w/have_any_two_players_with_the_exact_same_name/

[4] New Orleans Pelicans https://en.wikipedia.org/wiki/New_Orleans_Pelicans

[5] Charlotte Hornets https://en.wikipedia.org/wiki/Charlotte_Hornets