Predicting NBA Player Salaries with Stats

Alex Ojemann

2022-12-12

Overview

The purpose of my project is to to explore the factors that best predict NBA player pay and test different models to see which one best predicts player pay. I built a decision tree model from scratch and used it for the purpose of analyzing the which stats were most prevalent in predicting player salary. Using the knowledge of which stats were most prevalent from my model, I compared linear regression, logistic regression and tree-based models, and determined which model was the best in terms of parsimony and quality. Ultimately linear regression and the decision tree proved to be the best models of the types I explored.

Introduction

Assessing the value of an NBA player is a difficult problem. Assessing the success of a team is simpler; you can just look at wins and losses and still get a decent idea of how good the team was, use net rating to filter out some of the luck, or go a little further with SRS, a measure that also involves the strength of the competition faced. But assessing player value within the context of the team is a much more open ended question because of how dynamic the game is. In contrast to baseball, where outcomes are determined mostly by the hitter and pitcher exclusively, sometimes with a lesser influence from the fielder the ball is hit towards or the park dimensions and altitude, the outcome of an NBA play is much harder to attribute to a specific player. Factors like how well a screen was set, how quickly a defender reacted to an offensive player's cut, or whether a pass was thrown an inch too high or too low all have a significant effect on the outcome of the play, but we are only able to track the most visible aspects of the play like who took the shot, who got the rebound and who passed to the shooter over the course of hundreds of games per season.

One approach to this conundrum of player evaluation is to use predictive analytics to predict player pay based on the stats that are collected universally. There will always be aspects of player value that go unnoticed or underrepresented even by the most informed observers, but this approach at least gives us an idea of how players are valued historically by the decision makers who decide how much salary to offer a player. The model I built is called a decision tree. To build the decision tree, you determine the factor that is most telling and split the data down the middle and repeat this process with the lower and upper half of the data in terms of this factor. An example from the book Predictive Analytics by Eric Siegel is designed to predict whether a mortgage gets prepaid, and the most telling factor was "Is the interest rate $< 7.94\%$?" If the answer to that was yes, the risk of prepayment was 3.8%, if not the risk ballooned to 19.2%. This is a classification problem while predicting NBA player salaries is a regression problem, so our model will be slightly different. There is also the consideration that the model could overlearn, which is avoided by limiting the number of times we split the training data to build the model. One useful aspect of the decision tree outside of its predictive ability is that it's an explainable model, meaning that we can see how each factor contributes to the predictive scores generated by the model. I used this aspect of the tree to analyze which stats were most prevalent to player salaries. After analyzing this tree, I compared the effectiveness of a tree-based model, a linear regression model, and a logistic regression model using various combinations of the most prevalent factors from the tree I built. I determined that the decision tree model was the best in terms of quality and linear regression model was the best in terms of parsimony.

Relevant Project Specific Items

Data

The three data sets I used in this project are a player stats data set, a player salary data set, and a league average stats data set. Both the player stats and player salaries contain entries for each NBA player in each year of their career. The league average stats data set contains an entry for each NBA season. The player stats data set spans from 1950-2017, the salaries data set spans from 1990-2017, and the league average stats data set spans from 1946-2023, so the range of years I explored is from 1990-2017, when data from all three is available. There was one other single column data set that was necessary for the project, the salary cap for each year from 1990-2017, which I manually entered as a vector.

Preprocessing

There were a number of different aspects of the data that needed preprocessing. For one, I had to combine the three data sets in the right way. The player stats and player salaries data sets both had an entry for each player each year of their career, so they could be joined together. I used an inner join because players with an entry in the salaries data set with no stats entry wouldn't have anything to use to predict their salary, and players with an entry in the stats data set but no salary entry wouldn't have anything to predict. Hypothetically there shouldn't have been any but this did remove some data, a price I have to pay. Next, I removed players with less than 200 minutes from the data set because these players often were outliers in certain rate stats. For example, a player that only takes one field goal and makes it has an FG% of 100%, but that player isn't going to get paid very much because he only played enough to take one shot. Then, there was the matter of representing the player salaries as a portion of the salary cap. This is essential because NBA salaries have grown massively over the course of the 27 years I'm studying, with the league salary cap at $11871000 for the 1990-91 season, the first season I'm exploring, and all the way up to $94143000 for the 2016-17 season, the last season I'm exploring, so if we were to predict raw salary numbers, the predictions would be very inaccurate because the players in the earlier part of the data set would have their salaries overestimated and the players in the later part of the data set would have their salaries underestimated. So, after removing the dollar signs and commas and converting it to an integer, I divided the salaries column by the cap in the given year to form the column cap_percentage, which is what I would ultimately predict. Finally, there was the matter of adjusting for the league averages. I did so by subtracting the respective league average percentages from each of the the FG%, FT%, 2P%, 3P%, eFG%, and TS% columns (as is customary with percentage stats) and dividing the other applicable statistical columns by the given league average. Not all stats were applicable here, in particular some of the advanced stats like PER, WS, BPM, and VORP, because the league averages data set I have didn't include them. However, it was important to adjust these stats to the league average as much as possible because the frequency that they are accumulated by players changes significantly over time. For example, NBA teams averaged 106.3 points per game in the 1990-91 season, the first season I'm exploring, dropped all the way down to 93.4 in the 2003-04 season, then rose up again to 105.6 in the 2016-17 season, the last season I'm exploring.

Building The Decision Tree

To build my own decision tree model from scratch, I wrote three functions:

calculate_percentiles(): This function takes a data frame as a parameter and converts all statistical columns of the data set into percentiles. This is necessary for the decision tree because the tree is built by splitting the data at the 50th percentile of a given column. It's called every time the data is split into two new data frames because within those new data frames the spread of each column of data is different.

find_max_split(): This function takes a data frame as a parameter and finds the column where the difference in the average cap percentage of the top 50 percent and the bottom 50 percent in terms of that stat is the greatest. It returns the stat, the average cap percentage for the data where the players are below the 50th percentile in that stat, the average cap percentage for the data where the players are above or at the 50th percentile in that stat, the data frame with the players below the 50th percentile in that stat, and the data frame with the players above or at the 50th percentile in that stat. It's called recursively 2^(x+1)-1 times where x is the maximum depth of the tree.

build_decision_tree(): This function takes both a data frame and an integer as parameters. If that integer is greater than 0, it calls find_max_split and prints the stat, the average cap percentage for the data where the players are below the 50th percentile in that stat, and the average cap percentage for the data where the players are above or at the 50th percentile in that stat returned. It also calls itself with the lower and upper half data frames returned by find_max_split() as the data frame and x-1 as the integer where x is the current integer parameter.

I ultimately decided not to train and test this model and assign predictive scores to the players with it, however, it's still quite useful for the purpose of analyzing the stats that best predict player salary, one of my main goals for this project, because it's a fully explainable model.

Analysis of Decision Tree

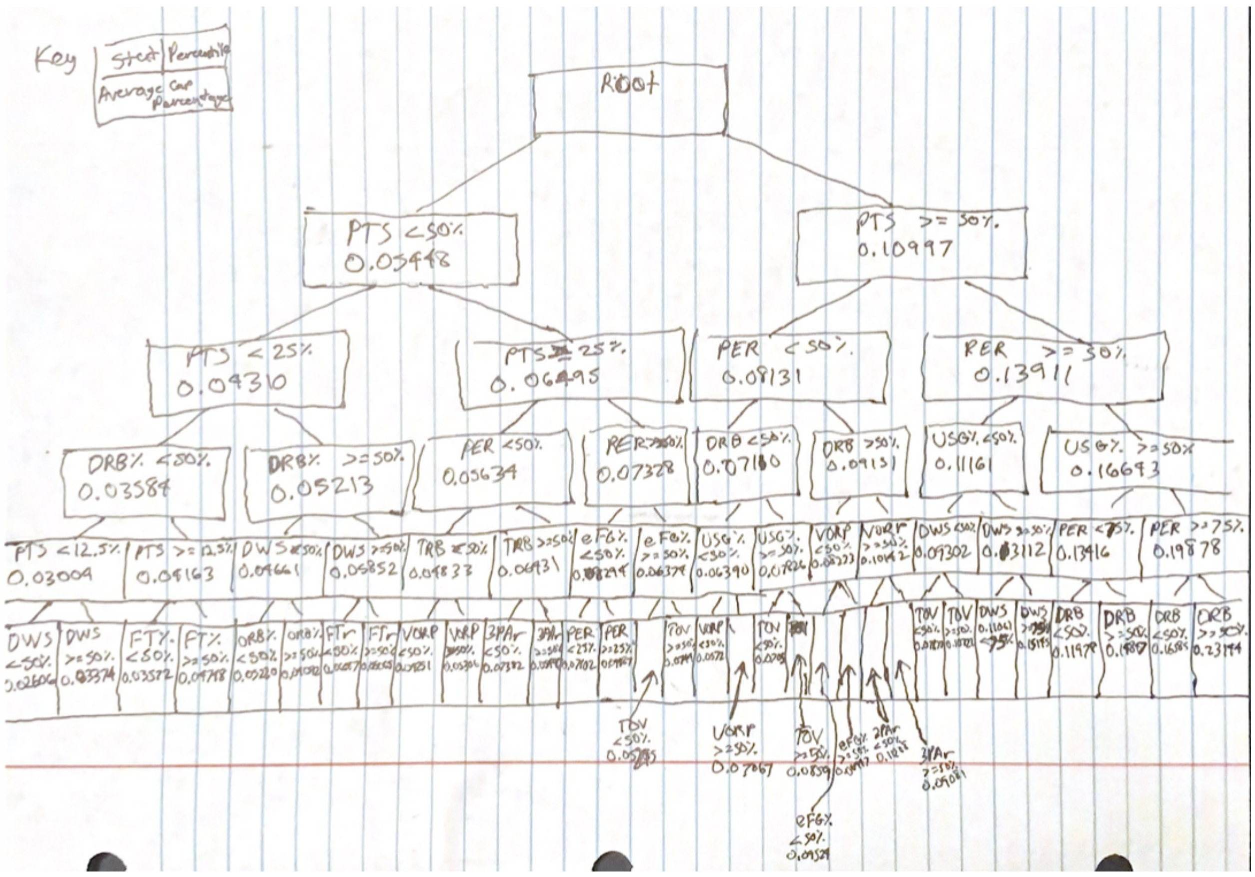

Figure 1: Tree

Above is the decision tree of depth five generated by using the entirety of the data set to train the model. I displayed five rows to give a high number of end categories (2^5 = 32) when displaying the tree. Looking at the fifth row, however, it may not be very useful for prediction because each of the data sets represented by each category only had 7661/32 = roughly 239 players each. Also supporting this idea is that three of the 16 stats in the fifth row (ORB%, FTr, and eFG%) are generally viewed as better to have higher values in but the split showed that the lower half of players in those stats had a higher cap percentage than the upper half, suggesting a lot of randomness due to small sample size is present. Turnovers (TOV) was present twice in the fifth row as well with the upper half having a higher average cap percentage than the lower half despite being a negative stat to accumulate, but it may be more complex than that because players that turn the ball over a lot are also usually responsible for creating a high portion of the team's offense. The league leaders in turnovers every year are mostly star caliber players earning maximum contracts. The fourth row is also questionable in its predictive ability to a lesser degree because each of the 16 categories has around 7661/16 = roughly 479 players and one of the eight stat splits (eFG%) that's viewed as a positive stat has a higher cap percentage in the lower 50% than the upper 50%. In general, we expect the higher rows to be more predictive because their splits are based on more data. With that caveat in mind, there are four things that stood out to me from this tree.

The game is still about a bucket: Total points (PTS) was tied for the most appearances in the top four rows (three) and was present in the first and second rows, showing even more predictive ability than the advanced metrics designed to encapsulate total player value. Many times the first stat NBA fans will look at when evaluating a player is his points per game, and not entirely without reason. For most players, scoring is the most significant way in which they contribute offensively. Putting the ball in the basket yourself usually (but not in all situations) reflects a higher value contribution to the offense than passing to the scorer. Also, players that score a high number of points often can do so because they have the ability to create their own shot at a high level, an important aspect of the game not well quantified on its own.

Advanced stats appear more in the right side of the tree: PER, VORP, and USG% in particular all appeared more and in higher levels of the right side of the tree than the left. The first split was points, so this is suggesting that these metrics may be more predictive for players with a requisite amount of raw scoring contribution, and playing time in order to accumulate that raw scoring contribution. This makes sense in that players who are in the lower half of points scored usually play a wider range of minutes (0-24ish per game) because they're deeper bench players, so more raw stats indicative of playing time are more predictive for them, whereas those in the upper half are likely playing a narrower range of minutes (24-36ish per game) because they're usually starters or 6/7th men, so stats that include their efficiency and/or quantify more granular aspects of the game are more predictive for them relatively speaking.

Scoring efficiency measures rarely appear: Only three times to be exact (eFG% twice, FT% once), all in the bottom two rows, and as previously mentioned the two times eFG% appeared the lower half had a higher average cap percentage that the upper half which is counterintuitive. This doesn't necessarily mean scoring efficiency is unimportant, more likely it's just highly role dependent. For example, star players that create their own shots often have lower scoring efficiency measures than role playing big men who get most of their shot attempts off of lobs and dump off passes near the rim, but command much higher salaries. Within each general offensive role (ex. shot creators, pick and roll bigs, catch and shoot specialists, etc), however, scoring efficiency measures, TS% and eFG% in particular because they quantify multiple aspects, are likely much more predictive than they are in the context of this tree.

Defense wins championships?: DWS, DRB, and DRB% appeared a combined eight times in this tree with their offensive counterparts only appearing once. That one offensive counterpart was ORB%, which as mentioned before actually had a higher cap percentage for the lower half than the upper half. DRB and DRB% being more predictive than ORB and ORB% does make basketball sense because defensive rebounds occur much more often than offensive rebounds thus you would expect being in a higher percentile of accumulating defensive rebounds to be more predictive. However, for DWS, which appeared four times on its own with no appearances for OWS or total WS, there isn't an intuitive explanation. It could be that there's truth to the old adage "offense wins games, defense wins championships," but more likely this is mostly due to randomness.

Comparing Models

To test the effectiveness of the three different model types used for this project (linear regression, logistic regression, and decision tree), I will use the most prevalent stats from the tree I built to start when testing models and adjust the stats used from there to find a model with a good balance of parsimony and quality where applicable. The most prevalent factors were PER, DWS, PTS, VORP, DRB, TOV, and USG%. All three of TOV's appearances were in the lowest row of the tree so it won't be used and I will replace DWS and DRB with WS and TRB because it doesn't make basketball sense not to include both offense and defense and the prevalence of defense only metrics over total metrics is likely due to randomness as described in the previous section.

Linear Regression

When deciding on the stats to put into the linear regression model, I used a similar process as I did in homework 4 to ensure a good blend of parsimony (p values) and quality (RMSE). That process was to add as many factors as possible as long as they all have a P value under 0.05 to ensure parsimony, with slight adjustments based on whether the estimate agreed with whether the stat was positive or negative.

The results were significantly different than those of the tree I built. My end model included VORP, TRB, USG%, which were among the most prevalent factors in the tree, but also STL, TOV, AST%, and TOV%, which were not. STL had a negative estimate which may seem surprising, but a player going for a steal often compromises the team's defense if they don't get it. The league leaders in steals are often small guards that aren't viewed as being very valuable defensively. TOV having a positive estimate may also seem surprising, but players that turn the ball over a lot are also usually responsible for creating a high portion of the team's offense. The league leaders in turnovers every year are mostly star caliber players earning maximum contracts. AST% and TOV% are interesting additions because not only do they have great p values and intuitive estimates (AST% is positive and TOV% is negative), they also represent a measure of passing which isn't really represented in the other stats here because AST/TOV ratio, a combination of these two, is often used to judge passers. Apart from this basketball reasoning, all of these stats, not including the intercept, have a p value less than 0.0001.

The calculated RMSE of the training set was 0.06276912. The calculated RMSE of the test set was 0.06264906. Since the model was even more accurate on the test set than the training set, we can reasonably assume that the model isn't overlearning.

Logistic Regression

For the logistic regression model I followed the same process as linear regression for choosing the stats, but the quality was measured in terms of the log loss rather than the RMSE because logistic regression minimizes log loss. Ultimately, I was only able to work 3 factors into the model, PTS, PER, and TRB without the factors not having p values less than 0.05.

The calculated log loss of the training set was 0.2743826. The calculated log loss of the test set was 0.2720853. Since the model was even more accurate on the test set than the training set, we can reasonably assume that the model isn't overlearning.

cTree

The ctree() model is different from the other two in that it decided which factors to put in the tree without any of the splits having a p value above 0.05. Thus the best formula was to let the tree use all the stats and decide which ones were most significant. It produced the tree below.

The tree uses different methods than mine in that it limits the growth based on the p values of each stat split rather than the depth. However, we can see that the factors look much more similar to my tree than those of the linear model.

The calculated RMSE of the training set was 0.06020114. The calculated RMSE of the test set was 0.06089209. Since the model was very nearly as accurate on the test set as it was the training set, we can reasonably assume that the model isn't overlearning.

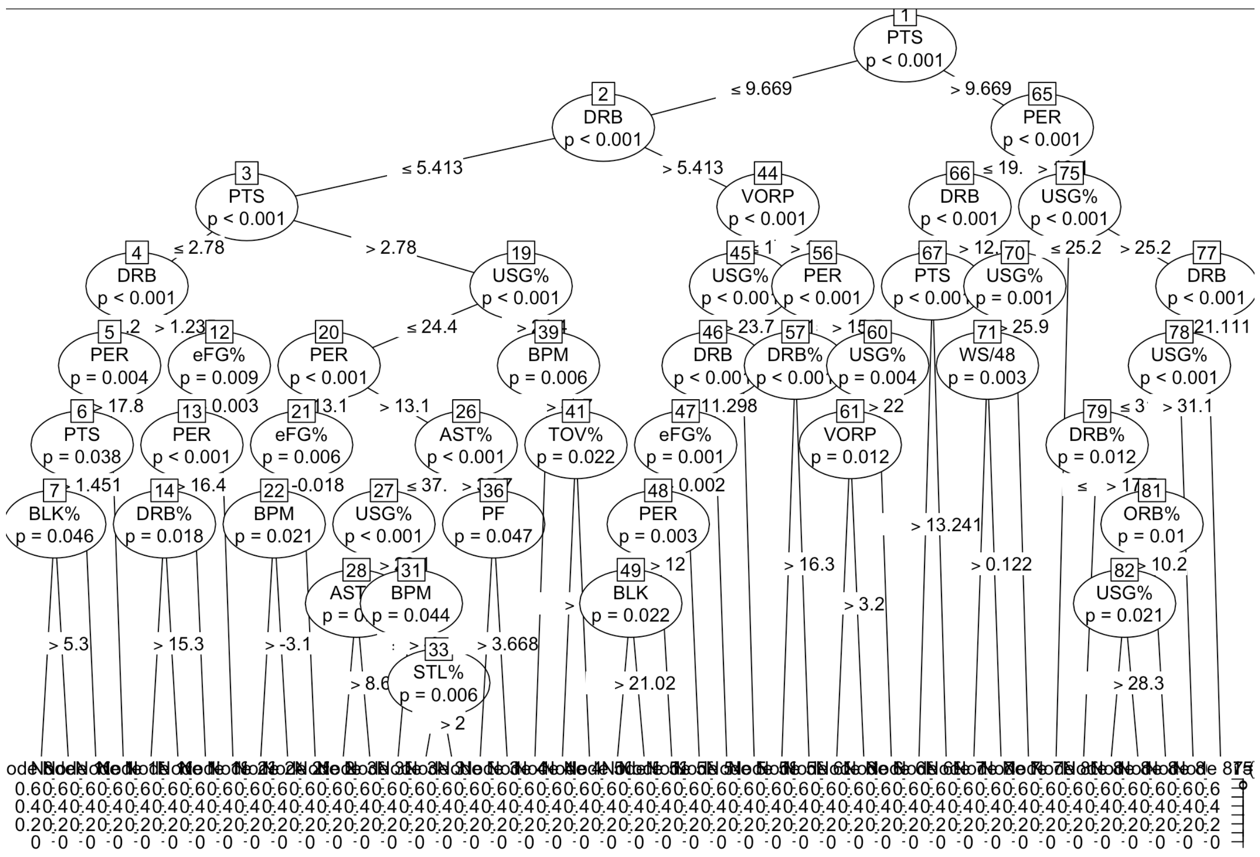

Figure 2: Output From ctree()

Winner

It's fair to say that the logistic regression model isn't the most predictive here because it was much harder to find parsimonious features for it than the linear model, although we can't really compare its accuracy to the other two because its accuracy is in terms of the log loss whereas the others are in terms or RMSE. This is somewhat expected because logistic regression is used much more often for classification problems, such as the one in Homework 5, rather than regression problems because of its shape. Of the remaining two, the ctree model had the lower RMSE on both the training set and the test set, while the linear model was much more parsimonious. So we'll say it's a tie.

Post Project Reflections

I believe I did achieve my goal of analyzing which stats are most predictive of NBA player pay and comparing different models for doing so. Knowing what I know now, the main thing I would've done differently is explore more types of models. My desire to build my own decision tree took a lot of time, and it may have been better spent exploring other models that R already has to find one that makes better predictions. Other things I would've done differently include adding per game stats to the data set by dividing the total stats by the number of games played and also searching for a data set that included per 75 or 100 possession stats, as those eliminate pace as a factor and are considered better for player evaluation than stats that aren't adjusted for pace.

Data Set Links

Player Stats: https://www.kaggle.com/datasets/drgilermo/nba-players-stats?select=Seasons_Stats.csv

Player Salaries: https://www.kaggle.com/datasets/whitefero/nba-player-salary-19902017

League Average Stats: https://www.basketball-reference.com/leagues/NBA_stats_per_game.html

Salary Cap by Year (Used for finding cap_percentage): https://www.spotrac.com/nba/cba/