Predicting Baseball Salaries

Greg Eastman

4/23/2021

Baseball franchises are both sports teams and businesses, as such they are trying to get the most value for their players. There are a lot of factors in salary negotiation, and since we do not have access to confidential sales data, we focus just on athlete performance. With data from the Sean Lahman Baseball Database[1], we build 4 statistical models. In the order that we present them we make a random forest, Ada boosted tree, gradient boosted tree, and a linear regression. The best of these models is then used for two tasks, finding the mot important salary predictors and the players that most outperformed their paychecks. We then synthesize this information to gain insights into what aspects of a player's game can be leveraged to best increase their pay.

Introduction

Baseball and statistics have a rich history, and since the Moneyball era in the early 2000's that relationship has only deepened. Analysts are now a core part of any franchise. They deliver data driven insights factoring into player development, recruitment, sales, and so much more. To get an outsider perspective on this work we will take a predictive economic approach.

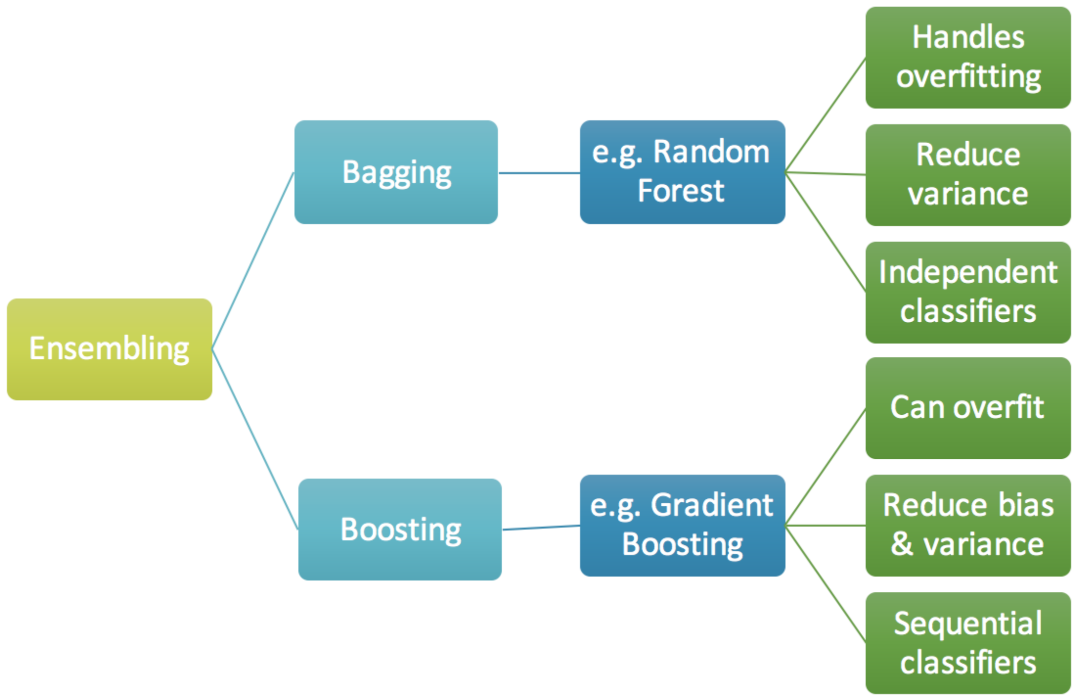
Franchises may be teams, but before that they are businesses whose product is baseball media. This allows us to assume that they will financially follow the same patterns as any other corporate entity. Therefore, they will overall not pay a player more then they will get in revenue. At the very least, if an assumption of perfect competition is true, they will pay the exact amount that athlete brings them in value. Specifically, this assumption stems from the fact players can negotiate contracts, so we can assume that if a player could get a better offer, they would take it. This should mean that overall, the MLB pays players their worth. Unfortunately, this is a rather simple assumption, which we will relax.

There are two major factors that create a less than perfectly competitive market, new franchises cannot easily enter the market, and rookie contracts disallow negotiation for 3-4 years[2]. These factors imply that players have weaker bargaining power than the franchises. While it is true that the corporations compete for players, there are many players, while on the other hand there a limited number of teams. So, a franchise can choose to pass up one player for another, but a player may not have that option with a team. Additionally, rookies will be forced into contracts that may severely undervalue them, because they cannot renegotiate when they perform well. Together these imply that salaries may underpay but should not overpay players. Mathematically we can denote this relationship as $\boldsymbol{Player\,Salary} <= \boldsymbol{Player\,Value}$**.** This begs the question, who were the most underpaid baseball players? We will compare multiple methodologies to get the best estimate of a player's salary and use then use the best one to find the 5 most underpaid players.

Methodology and Models

This work compared three tree-based models and a linear regression to predict a player's salary. We will go through the way that each of the models works. This next section assumes that the reader has familiarity with decision trees. For more resources outside of this work this link[3] may be useful. Additionally, we will not walk through the math for each model, as doing so would be superfluous to the purpose of this work. Before going into the first model, a random forest, the diagram below will provide some important information.

4



The overall methods this work uses is called Ensembling, shown as the left most yellow block. These strategies are ideologically aligned with the saying "many heads are better than one". These techniques work by creating many weak decision trees and conglomerating their predictions. This helps reduce variance because having more models making guesses allows any individual overcompensation to be corrected. Although bagging and boosting do this differently. In bagging, the trees are grown independent from each other, while boosting does it sequentially. Both approaches have benefits and deficits we will discuss further.

Random Forest

A random forest is a bagging technique that grows decision trees from different pieces of data to avoid over fitting. This is a powerful predictive technique with proven performance[5]. It works by first randomly sampling the training data and building a model on each of the created subsets. This helps prevent overtraining because no single model will be able to capture and overpredict on the entire data. Instead, any overtraining on one subset gets minimized by the other trees that learned from different data. Additionally, in random forest when a tree considers splitting a node, it randomly subsets a feature. This random element helps it to try new splits that may lead to otherwise unlearnable insights. It also helps the trees avoid converging on just one feature. Since each tree is made independently from each other on different subsets of data, and with random feature splits, the model is robust.

The benefits of this technique are two-fold, as shown in the diagram. First, its classifiers are independent, so the models do not affect one another. This benefit leads into the next, handling over fitting. By grouping a set of different trees, grown on different data, the model can avoid becoming too dependent on trends specific to the data. The models have all generated with random splits so when it groups them together, even if one model over fit in a certain way, it is compensated for by the large number of other independently grown trees. This allows researchers to feel more comfortable when training a model because they do not have to worry a large performance loss on the testing data. Next, we will go into boosting models.

Ada Boost

Ada boost is the original boosting technique[6], and it works by sequentially growing trees one on top of another. This technique by growing new trees on the areas that the previous missed one missed. They all link together to form the final full model. To do this we make two sets of weights, the first is for the data and the others are for the trees. The weighting of the data is a major key to this process's success. After each link in the chain of trees the data is given a new distribution. Values that were misclassified in previous steps get a higher weight. This makes them more likely to appear later and to become more influential in the loss minimization of future trees. The weights for each tree are calculated with a function that creates a weighted average. The more useful trees get a higher weight, and the total sum of the weights is always equal to one. This approach has several advantages and disadvantages.

The advantage of this technique is that it generally converges faster than a forest and that it reduces bias, but it is inflexible and prone to overfitting. The sequential nature of the model means that there is less randomization that needs to be done. The guesses are more intentional, this means that the model should get to the desired value quicker than a random forest.

Additionally, by constantly improving on missed guesses instead of just the overall error, the model helps to reduce bias in the error term. This makes its importance weights very useful because it is more robust to incorrectly signing an effect. Although, the models have an issue with overfitting. Since they are all built on the same data, and more so they are built on each other, they can quickly start learning trends that are sample specific. This is usually addressed by tuning the learning rate and number of samples. Although, the model still struggles with the fact that it can only use a specific loss function. This means that it may struggle with modeling different types of data because it is more rigid in its construction. The next technique addresses this issue.

Gradient Boost

Gradient boosting is like ada boost, the only major difference is that instead of using the ada boost function to minimize error, it uses gradient descent. Through this method the weights on the data and the trees are trained. After each iteration, the model will reduce the loss by walking down its gradient towards the minima. This method is more flexible for several reasons. First, in ada boost the weights are additively bounded, which is not the case here, as the weights are chosen iteratively without a built-in constraint. Additionally, you can perform this methodology over any loss function instead of just one. This helps to predict different types of data and allows researchers to stay updated with newer techniques. The last model is one of the most classic statistical models.

Linear Regression

Linear regression is a staple of the statistical world. It works by finding the line through the data that minimizes the sum of squared errors. The math for this can be found here[7], and is one of the most classic statistical minimization problems. The model has a strict set of assumptions that are not usually closely but not fully met. Although, its easy interpretability makes it a powerful tool. The understandability has made it an essential tool for causal research in disciplines ranging from chemistry to biology to sociology to economics. Although, we will not go into how to interpret this model, as this work is purely predictive. For the sake of completeness, we will discuss the assumptions behind this linear model.

Linear regression has four major assumptions, linear relationship, no multicollinearity, homoskedasticity, and normality. The linear relationship requirement is often the laxest of all. It is often investigate by forming scatter plots and the met by transforming the data so that the relationship becomes more linear. The tradeoff is that complex transformations to get linearity may affect interpretability. The next assumption is that the model must not be multicollinear. In other words, no two predictors should be highly correlated. This is usually tested for with variance inflation factors, a VIF above 3 is usually cause for concern, above 5 is a problem, and above 10 is a total violation. Omitting a highly correlated is the prescribed cure. The next assumption is homoskedasticity, which is tested by looking at a residual vs. fit plot. If the data has a pattern, such as a cone around the y=0 line, then there is an issue, if it is randomly distributed then the assumption is met. Finally, the normality assumption is tested with a QQ-plot. The pattern of data should follow the normal print on the plot, if it has a large deviation form the line its assumption is not met. This also fixed by transforming data to get a normal curve. With all the models covered, we will move onto the experiment.

Experiment

We took a 3-step approach to answer the research questions by finding the best model, looking at the important variables, then isolating the most underpaid players. Although, before going into the specifics of the experiment we will go over the data.

Data

We are using data from the Sean Lahman database. This database was assembled by sports reporter who has been documenting baseball statistics since 1996. We used only data from that year and beyond because the earlier years are filled with missing salaries and other values. Additionally, for a more robust analysis we got inflation data from the Federal Reserve[8] and used that to adjust the salary data to the monetary value of 2020. The salaries were then logged to account for the natural skew in the data. Next, we removed all players that had less than 50 games and all pitchers. We wanted to keep only the starters and not substitutes, and pitchers are not paid for fielding and hitting ability. The final data cleaning step was to bin all continuous variables so that we can see what is more impactful on salaries, over or under performance. With the data covered we will go over the results of the models.

Model Results

The four models were measured on mean squared error and R^2, which in the end both told the same story, the gradient boost was the best model.

Model	rMSE	R^2
Random Forest	.810	.637
Ada Boost	.850	.600
Gradient Boost	.791	.654
Linear Regression	.815	.633

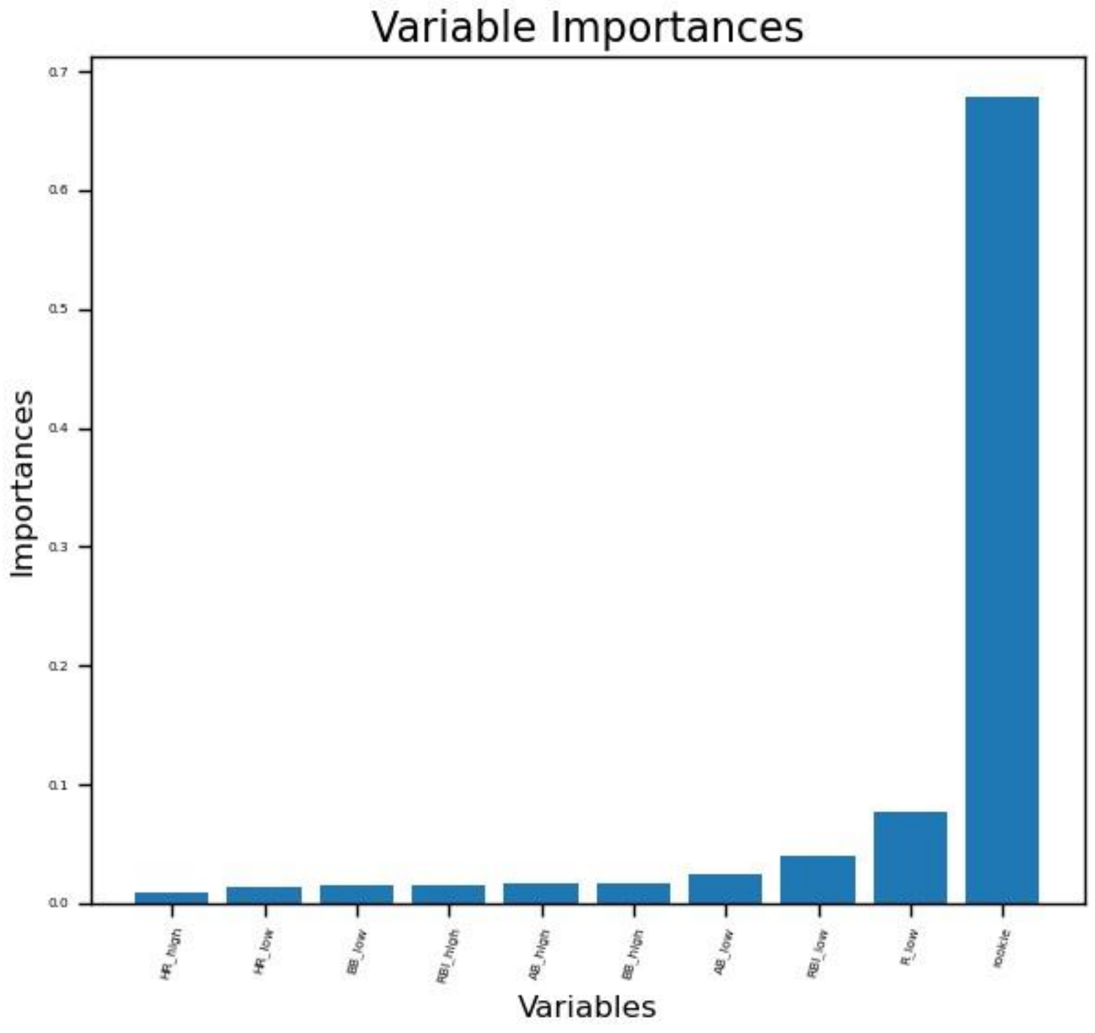
We will break down the table one model at a time by comparing them to the basic multiple linear regression, beginning with the random forest. The forest outperformed the control model by half a percentage point in both rMSE and R^2. This improvement over a regression shows that a random forest is better at handling predictions in a space that may not be linear. Additionally, this is the second-best model overall, itias only overshadowed by the gradient boost.

The gradient model had by far the best performance, it explains over 65% of the variation and has the lowest rMSE. The variance explanation, or R^2 is a common metric for model performance, as explaining more of the variation in the data is better. Although, it is a measure relative to a null model, and not in absolute terms. This is where rMSE comes in, as it measures the standard deviation of the error term in the units of log salary. So having an rMSE of less .8 is strong and seeing it outperformed the regression by over 2 percentage points in both categories. All this together means that this is the model that the next sections will use for their analysis. Although, before going into the next part of the experiment, there is one other tree-based method to still discuss.

The ada boost model had the worst performance overall. This is surprising because it is more complex than the regression, and since it is not making a simple linearity assumption, it should better predict the data. Although, we see in the table that this is not the case because the regression beat it by 3 percentage points in both rMSE and R^2. This could stem from the complexity of factors going into salaries, and that the simple function to refine the tree is just not quite up to that task. Meanwhile a simpler linear model may be more successful due to the large number of inputs and data causing it to split the difference better but not guess on specific points as well. Since the regression was just a baseline model, we will not go over it much and will instead analyze what is most important in a player's salary.

Importance Weights

The importance weights, or the weights given to how useful each variable is in predicting the log of salary, were those generated by the gradient boost model. This model was chosen due to its superior performance. The chart on the next page will highlight the variables that best determine salary.



The first thing to note is that the rookie variable is the most important, it holds more weight than the next 9 combined. This makes sense as rookies sign before they get into the major league for well under the million-dollar contracts they grow into. Since these players cannot renegotiate or go to another team there is no bargaining power. This goes back to the idea discussed earlier that the league holds more power in the negotiations than the players. Regardless, it is not surprising that being a rookie is the largest determinant of salary, but the magnitude is still notable. The next finding involves multiple stats.

The most important determiners of player worth are runs, at bats, walks, RBI's, and homeruns. The first stat to talk about is runs. Scoring a lot is not important in the model, but not scoring is meaningful. This likely means that teams do not pay well for those who do not deliver, but also don't give major bonuses to those who over-deliver. This is shown by the "R_low" bar.

Next, having low RBIs seems to do the same as runs, except that having a lot of RBI's does help to increase pay. Next, at bats seem to determine salary both high and low. Although this is an important variable to have, its weights present an endogeneity scenario because it is highly likely that teams want to play those that they pay more, and not the other way around. Then we are left with walks and homeruns. Both are extremely important and having lots of homers and base on balls drastically changes pay. With this all covered, we will move onto what is not in the chart.

There is a surprising non-finding to report, that the player's team was not a significant determinant of pay. It is shocking that teams are not a major factor, because baseball is well known for having some extremely wealthy and "extremely poor" teams. The Yankees have historically had a mountain of money that they spend on their roster, and the Oakland A's have been quite the opposite. Yet neither team seems to matter in the model. This implies that the league is at a strong equilibrium in player evaluation, and that salary caps may be working to keep things more even across franchises. With all this information covered we will take a quick look at the most underpaid players since 1996.

The Five Most Underpaid Players

The five most underpaid players were all undervalued by roughly the same amount, these athletes in order of profit for their franchise were, Jorge Pasada (2000), Tino Martinez (1996), Scott Brosius (1998), Bernie Williams (1996), and Ryan Ludwig (2008). Each of these players were all stars in these years, and most of them for the first time. Additionally, three of them (Martinez, Williams, and Ludwig) were MVPs in those years and had not been previously. This implies that they all performed unexpectedly well, and when looking at the stats those were the best years those players had. Furthermore, it is interesting to note that even though the data goes to 2019 all the misevaluations were around the early 2000s, with only one in the latter half of the decade. The time around 2001 is significant, because that is when Moneyball started its rise, and statistics instead of scouts began choosing players. Since the contracts have gotten closer to the value a player brings as time has passed, we can see an implication that statistics has made players and teams into more informed and accurate actors when signing contracts.

Conclusion

Baseball is one of the most statistically driven sports, and there is much more to learn about the game with each study. In this work we looked at several different modeling strategies to predict the value of a player. We found that gradient boosting performed the best. Then using this finding, we investigated the most important determinants of a player’s success. We found that being a rookie has a massive impact, and that underperforming is more impactful on salary than overperforming. The next thing of note was that teams did not play a huge role in a player’s pay, the stats all mattered more. Finally, we found that as time went on, player pay got closer to the value their performance brought their teams each year.

These findings could be important for players trying to take that next step in their baseball career. Teams seem to pay most for players that hit on par with other pros rather than for defensive excellence, or other overachieving offensive traits. The teams should not be a factor in determining where a player wants to go, because franchises are not impactfully different from each other on an athlete’s salary. Outside of this advice, we find that statistics have made baseball more economically efficient for teams and players by making the pay more precise to a player’s ability.

Going forward, other work could build on this by taking a different modeling approach. Networks may be able to lend a performance boost, or clustering methods could lend insights on what traits the top paid players exhibit. Also, Bayesian statistics may be able to utilize priors to better leverage the information in this data. Finally, combining the database with sales and other metrics for player valuation could help to increase model accuracy and refine the findings of this paper.

References

[1] "Download Lahman's Baseball Database." SeanLahman.com. February 16, 2021. Accessed April 25, 2021. http://www.seanlahman.com/baseball-archive/statistics/.
[2] "Service Time: Glossary." MLB.com. Accessed April 25, 2021. https://www.mlb.com/glossary/transactions/service-time#:~:text=Upon reaching six years of,his free-agent seasons).
[3] "Decision Tree Algorithm, Explained." KDnuggets. Accessed April 25, 2021. https://www.kdnuggets.com/2020/01/decision-tree-algorithm-explained.html.
[4] Grover, Prince. "Gradient Boosting from Scratch." Medium. August 01, 2019. Accessed April 25, 2021. https://medium.com/mlreview/gradient-boosting-from-scratch-1e317ae4587d.
[5] Fawagreh, Khaled, Mohamed Medhat Gaber, and Eyad Elyan. "Random forests: from early developments to recent advancements." *Systems Science & Control Engineering: An Open Access Journal* 2, no. 1 (2014): 602-609.
[6] Freund, Yoav, and Robert E. Schapire. "A decision-theoretic generalization of on-line learning and an application to boosting." *Journal of computer and system sciences* 55, no. 1 (1997): 119-139.
[7] "5.4 - A Matrix Formulation of the Multiple Regression Model." 5.4 - A Matrix Formulation of the Multiple Regression Model | STAT 462. Accessed April 25, 2021. https://online.stat.psu.edu/stat462/node/132/.
[8] "Inflation, Consumer Prices for the United States." FRED. March 03, 2020. Accessed April 25, 2021. https://fred.stlouisfed.org/series/FPCPITOTLZGUSA.