Predicting MLB Players Salaries

Emmet Donahue, Eric Siegel, Layton Webber

Introduction

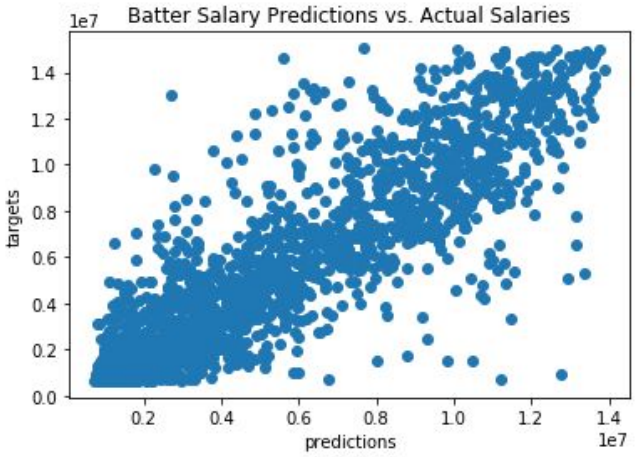
Our data set was compiled by the reporter Sean Lahman, and can be found here. The database contains batting and pitching statistics for the years 1871-2018. In addition, salary data for players is provided from years 1865-2018. These datasets are provided in separate csv files, which we stitched together for our uses. We did this by using year, team, and player name to uniquely identify an entry in both the stats and salaries, and then we combined these two into a new row in our updated dataset. Note, since a players statistics for the previous year are used in salary negotiations, we appended their salary for the following year to their statistics from the previous year. That is, if a player has statistics for year 2000, the salary that appears alongside these stats is their pay in 2001. There is another modification that we make, which is that a player's salary for the previous year is provided as another data point. In the above example, the 2000 statistics would also have a 2000 player salary under the header prev_salary.

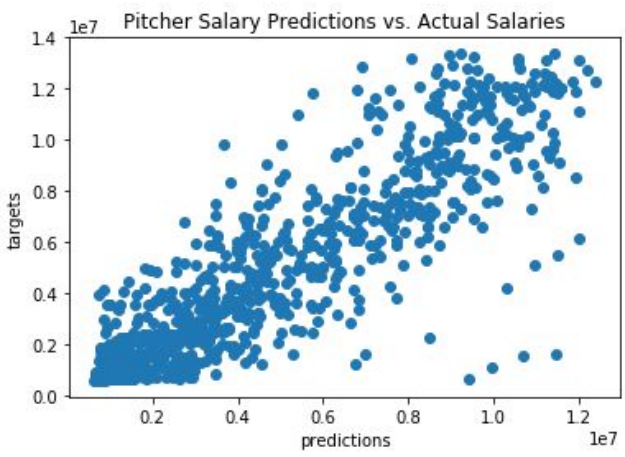
We used various methods to clean the complete dataset before analyzing it. First, since the salary data is of a large range of years, we standardized all salaries to one year to account for inflation. To do this, we retrieved the average MLB salary from every year from 1865-2018, and from this calculated the average rate of inflation from year to year. From these inflation rates, we standardized all salaries to 2018 dollars. Next, we used the standard statistical rule that outliers are values below 1.5*Q1 or above 1.5*Q3 to clean our databases of players who are paid exceptionally high or low. Finally, players who posted no meaningful statistics for a season, such

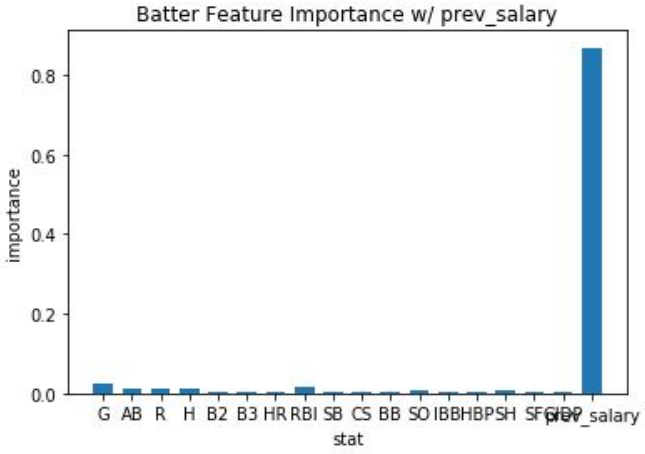
as players with long term injuries and players who were traded, and posted no statistics for the team they were traded from, were dropped from the dataset.

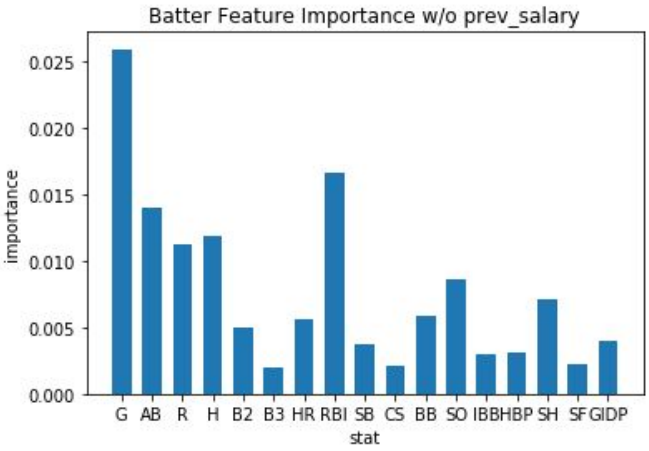
Prediction Method 1: Random Forest Regressor

The first algorithm that we applied to the dataset was the random forest regressor. This algorithm uses the feature variables to construct yes/no decision trees to predict the target variable. The random forest algorithm produces a forest of such trees and makes its predictions by averaging the those of all the decision trees in the forest. This averaging process helps smooth out overfitting in any one tree by balancing it against the others. We choose this algorithm because it can capture nonlinear relationships, which we thought may appear in the data. Additionally, the random forest encodes feature variable importances as part of its model, which gives us insight into which variables are the most important for determining a player's salary. Below we can see the plot of the test targets and our models predictions, showing the semi-accurate predictive power of the regressor for our dataset.

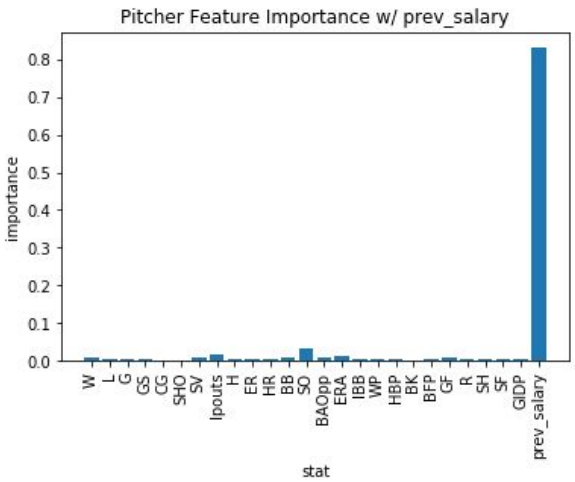


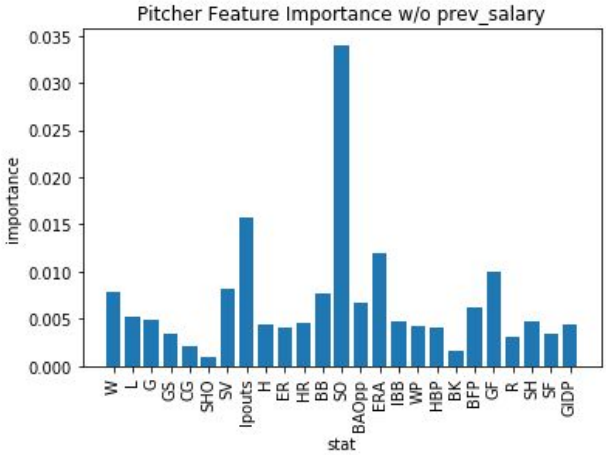






As we can see from the batter feature importance plots, the player's previous year salary is the strongest predictor of their future salary. This could suggest that teams generally have confidence in their proven players regardless of how that player actually performs in a given year. Removing this feature to explore the others a little more, we see that games played is the next strongest feature. This is not too surprising as a teams more valuable players would generally be played more for their skills. The next few stats are again all what one might expect, runs scored due to a player (RBI), at bats (AB), runs scored by a player (R), and hits (H). Overall, the conclusion one can draw is that one's intuition about how we might rate a batter is generally also how that player is evaluated by their team.





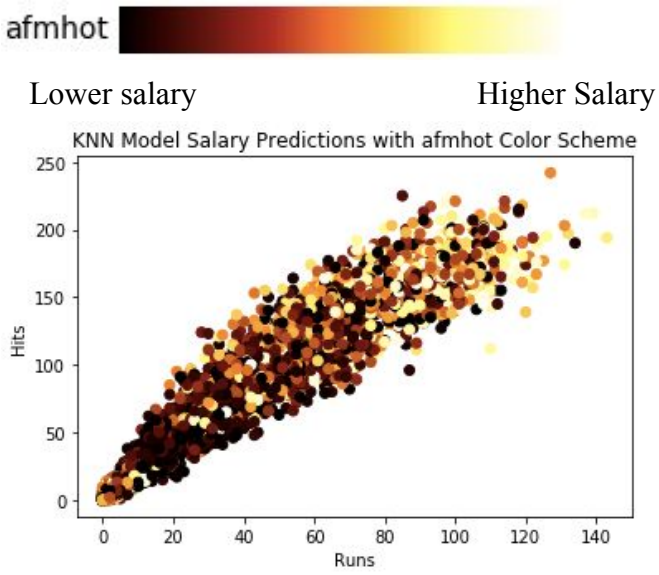
A similar analysis for pitchers reveals largely the same conclusions. Prev_salary is the strongest predictor, and obvious statistics such as strikeouts (SO) and outs pitched (Ipouts) became more apparent once this is removed.

Prediction Method 2: KNN Means Model from sklearn module

After the random forest regressor produced semi-accurate results, we employed the KNN classifier from the sklearn module to both the pitcher and batter datasets to see if we could create a more accurate model. We partitioned the range of salaries in both the pitcher and batter datasets into 500 salary categories, and assigned every player their corresponding salary category as the target of the algorithm. Salaries ranged from about $200,000 to $35,000,000, so every salary category encompassed all salaries in a ~ $70,000 range. We then fit our model to 40% of our data, and tested it on the remaining 60%. This model then predicted players salary category with ~94% accuracy.

Our salary data was right skewed, meaning that more players fell into low salary categories than high salary categories, resulting in our target data not being evenly distributed across all salary categories. In order to evenly distribute target data across all salary categories, we partitioned the salary data into 100 percentiles, and assigned every player their corresponding percentile. Since every percentile had about the same number of players in it, our data was distributed about evenly across all salary categories. When we fit our model to this data, it predicted player's salary categories at only about 70% accuracy. While we are not sure why this partition created a less accurate model, we guessed that the percentile approach eliminates the right skew of the salary distribution, which the previous model may have been using to improve its predictions.

The below scatterplot shows the KNN model's predictions of batters salaries based on the number of runs they scored, and hits they made in a season. As one would expect, the model predicts that players who score more runs and make more hits will be paid more. Visualizations like this, which show a direct correlation between statistics and our model's predictions allow us to determine which variables most influence the KNN model's predictions.



Prediction Method 3: Multiple Linear Regression with Pitcher Data.

Finally, we constructed a multiple regression model to predict pitcher's salary. Regression analysis has existed much longer than machine learning, so comparing it to the KNN model and random forest regressor would allow us to compare older prediction models to newer ones.

The first step in preparing our multiple regression model was subsetting the pitcher statistics from the pitcher csv file that we expected would strongly influence a pitcher's salary. These statistics were hits, runs, earned runs, walks, strikeouts, wins, loses, saves, home runs and earned run average. This was decided on by the significance (p-value) of each variable as a

single regressor as seen below:

```
Variable: H LinregressResult(slope=5.539805425788131e-06, intercept=81.17941273503821, rvalue=0.3000781878170832, pvalue=3.8178450310139584e-106, stderr=2.472017606018855e-07)
Variable: R LinregressResult(slope=2.3747578216098577e-06, intercept=41.18855412323536, rvalue=0.25467293849573863, pvalue=5.539222478676989e-76, stderr=1.2657770425485956e-07)
Variable: ER LinregressResult(slope=2.1529532548388587e-06, intercept=37.57481722710439, rvalue=0.2522405047866131, pvalue=1.5838299440134067e-74, stderr=1.1593821035464675e-07)
Variable: BB LinregressResult(slope=1.2846182129734087e-06, intercept=32.689153057108356, rvalue=0.20058418830813002, pvalue=3.0865329913702066e-47, stderr=8.807284652785147e-08)
Variable: SO LinregressResult(slope=4.069351728067276e-06, intercept=62.995656098732795, rvalue=0.2985617499798774, pvalue=4.818101493042797e-105, stderr=1.8259928118754222e-07)
Variable: W LinregressResult(slope=4.2518434255161266e-07, intercept=4.742636345144697, rvalue=0.31573244824900976, pvalue=6.648107194560038e-118, stderr=1.7936462269958828e-08)
Variable: L LinregressResult(slope=2.9952409939169283e-07, intercept=4.591243706511794, rvalue=0.2666454226894248 7, pvalue=2.2062698341884516e-83, stderr=1.5197219890729253e-08)
Variable: SV LinregressResult(slope=5.4507819935416870e-07, intercept=1.689690740628 79, rvalue=0.2107825605286486, pvalue=4.3699579054596514e-52, stderr=3.548440509392562e-08)
Variable: HR LinregressResult(slope=5.576124173089121e-07, intercept=8.677401012079915, rvalue=0.24401518583174736, pvalue=1.0174902262777824e-69, stderr=3.1107630748238304e-08)
Variable: ERA LinregressResult(slope=-5.922629904233685e-08, intercept=4.279080951323576, rvalue=-0.08143043950979902, pvalue=6.232709518913451e-09, stderr=1.0175719167207154e-08)
Variable: prev_salary LinregressResult(slope=0.6903090948274953, intercept=-95564.20004046336, rvalue=0.8102976263931998, pvalue=0.0, stderr=0.007007972334545201)
```

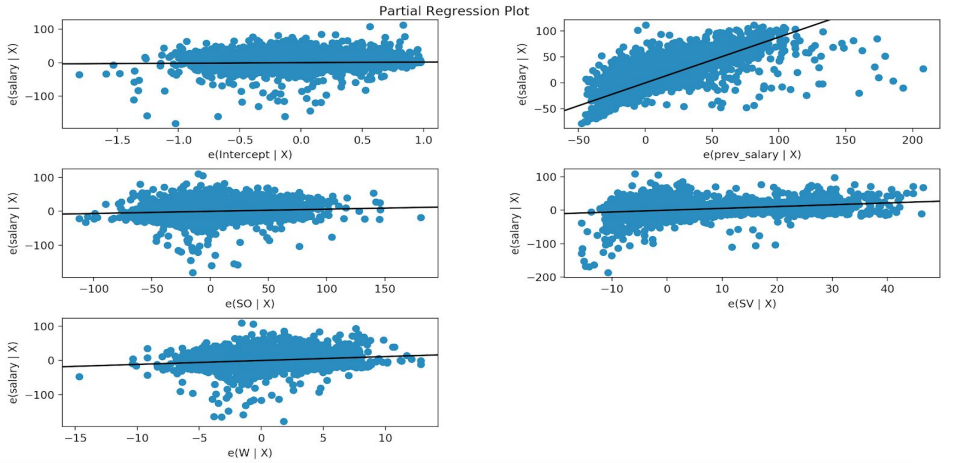
Next, we used a stepwise forward regression algorithm to select the model that explains the most variation (using the statistic adjusted r squared) in predicting salary from our list of predictors. Since the statsmodels.formula.api package did not contain a stepwise forward regression algorithm, we found this one on the internet. Stepwise forward regression starts with a model with no predictors and finds the configuration of predictors that explains the most variation in the model.

The results of the stepwise forward regression algorithm gave us a model for predicting salary. The model was: salary ~ prev_salary + SO + SV + W + H + ER + ERA (The " ~" means "predicted by"). From there, we looked at the summary table of the model and removed variables that either had a high variance inflation factor (greater than 5), which tells us if a variable can be predicted from the presence of other variables, or had a p-value that was insignificant (greater than 0.05). Our final model was: salary ~ prev_salary + SO + SV + W. This is the summary table:

```
[4.341289774492058, 1.0659219033909038, 2.891174067489084, 1.1046362103698522, 3.051489396756]
                            OLS Regression Results
==============================================================================
Dep. Variable:                 salary   R-squared:                       0.714
Model:                            OLS   Adj. R-squared:                  0.714
Method:                 Least Squares   F-statistic:                     3169.
Date:                Sun, 17 Mar 2019   Prob (F-statistic):               0.00
Time:                        15:19:37   Log-Likelihood:                -22168.
No. Observations:                5077   AIC:                         4.435e+04
Df Residuals:                    5072   BIC:                         4.438e+04
Df Model:                           4
Covariance Type:            nonrobust
==============================================================================
                 coef    std err          t      P>|t|      [0.025      0.975]
------------------------------------------------------------------------------
Intercept      1.8407      0.558      3.301      0.001       0.748       2.934
prev_salary    0.8835      0.009     97.121      0.000       0.866       0.901
SO             0.0642      0.009      6.854      0.000       0.046       0.083
SV             0.5420      0.031     17.766      0.000       0.482       0.602
W              1.1438      0.097     11.747      0.000       0.953       1.335
==============================================================================
Omnibus:                     1094.810   Durbin-Watson:                   1.924
Prob(Omnibus):                  0.000   Jarque-Bera (JB):            18505.926
Skew:                          -0.569   Prob(JB):                         0.00
Kurtosis:                      12.284   Cond. No.                         202.
==============================================================================
```

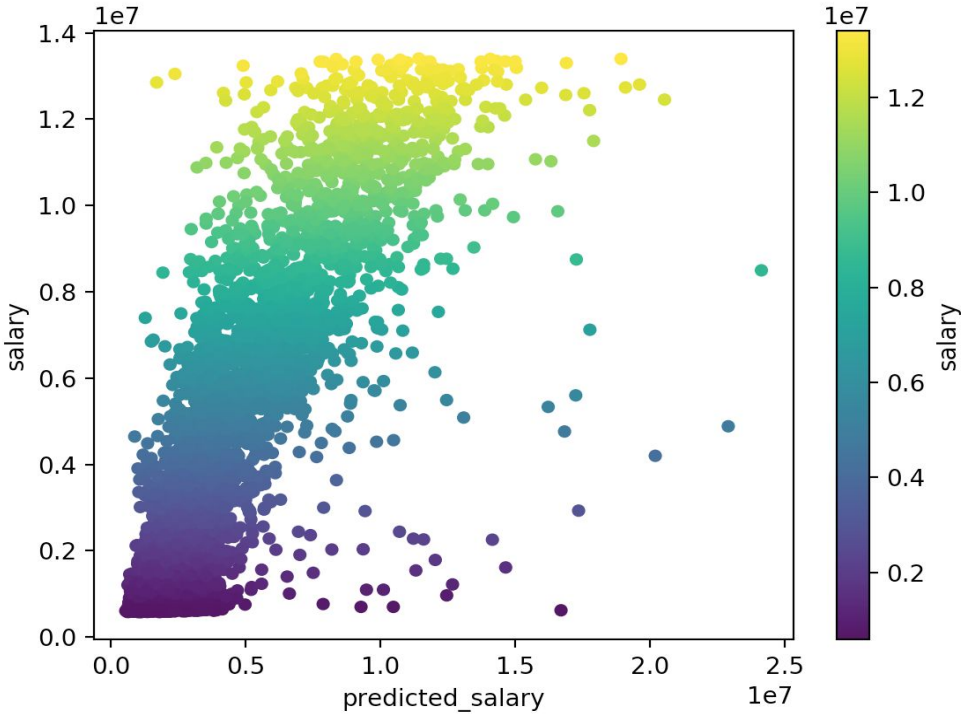
As seen in the picture above, all the variables have a variance inflation factor less than 5 and there p-values are less than 0.05. The adjusted r-squared value for this model is 0.714, which means the model explains 71.4% of the variation in the data predicting salary.

The next step was to examine the slopes of the mode variables' coefficients predicting salary. This was done using a graphics function (plot_partregress_grid()) in the statsmodels.api module. This was the result:



As seen in the Random Forest Regressor method, the strong positive slope of the prev_salary variable indicates its strength as a predictor of salary. The large t-statistic value shown in the previous summary table (97.121) indicates the model heavily relies on the prev_salary variable to explain variation in the salary variable .

The final step in analyzing our model was to see how accurate our final model was at predicting salary. This was accomplished by creating a scatterplot of our model's predicted salaries on the x axis and the actual pitcher's salary. The graph is below:



The colormap helps indicate the player's classification of their actual salaries, with the players making less money as darker points and the players making the most money as lighter points. This scatterplot shows that our model has a general linear trend but that the predictions suffer from non-constant variance as the actual salaries increase. This non-constant variance can be seen from the range of x-values for a single y-value. From this scatterplot, it is clear our multiple regression model did not stack up as well to our KNN model for predicting pitcher salaries.

Appendix A. Statistics Abbreviations

General Stats	Pitching Data	Batter Data
playerID - Player ID code yearID - Year stint - player's stint (order of appearances within a season) teamID - Team lgID - League	W - Wins L - Losses G - Games GS - Games Started CG - Complete Games SHO - Shutouts SV - Saves IPOuts - Outs Pitched (innings pitched x 3) H - Hits ER - Earned Runs HR - Homeruns BB - Walks SO - Strikeouts BAOpp - Opponent's Batting Average ERA - Earned Run Average IBB - Intentional Walks WP - Wild Pitches HBP - Batters Hit By Pitch BK - Balks BFP - Batters faced by Pitcher GF - Games Finished R - Runs Allowed SH - Sacrifices by opposing batters SF - Sacrifice flies by opposing batters GIDP - Grounded into double plays by opposing batter	G - Games AB - At Bats R - Runs H - Hits B2 - Doubles B3 - Triples HR - Homeruns RBI - Runs Batted In SB - Stolen Bases CS - Caught Stealing BB - Base on Balls SO - Strikeouts IBB - Intentional walks HBP - Hit by pitch SH - Sacrifice hits SF - Sacrifice flies GIDP - Grounded into double plays