Salary Prediction Write-up

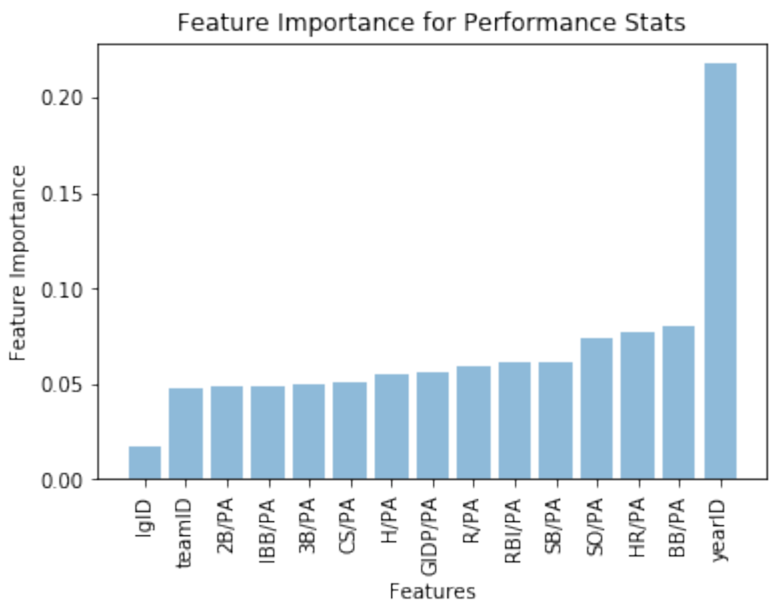
In my Master's program I have been learning about Logistic Regression and Support Vector Machine models, and using ROC curves and AUC values as a way to test them. I wanted to apply these tools to something I am interested in and understand relatively well. As a personal project, I decided to use the classic offensive baseball statistics.

There is some readily available clean data provided by Kaggle that has historical offensive statistics (Batting.csv.zip) and a separate data set with contract details going back to 1985 (Salaries.csv). I acquired both of these from https://www.kaggle.com/open-source-sports/baseball-databank.

I was interested in prediction so I decided to try to see if I could predict if a player would be in the top quartile of salary based on their offensive performance alone.

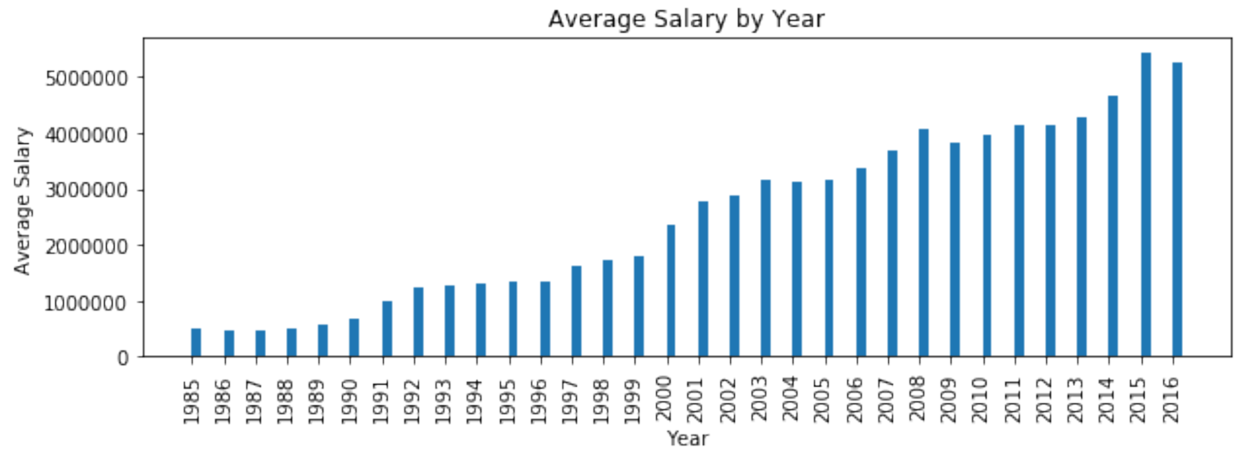
To get rid of pitchers and players whom had few plate appearances, I removed players with less than 70 at bats for a season. I made the purely numbered statistics into rates by dividing by plate appearances. For instance, the “2B” feature became the “2B/PA” feature.

First, I wanted to find out some of the most predictive features. I ran a decision tree classifier and used the feature importances attribute to graph each variables importance.



By far the most predictive variable was year, which was unsurprising considering the effects of inflation and increases in overall baseball revenue since 1985. All other features were not hugely predictive, but league ID (NL or AL), was understandably the least informative.

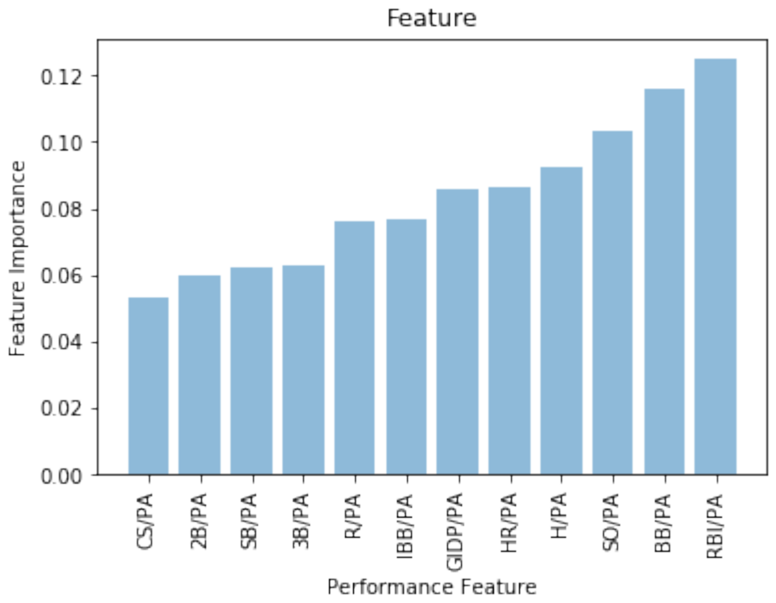
As an aside, I was interested in how much average salary had increased per year, so I made this graph



I was interested to see that there was a small blip in salary after 2008, which may correspond to the economic recession. I was also surprised by the degree to which salaries from the 1980s had quadrupled by the early 2000s, and then continued to grow at a staggering rate.

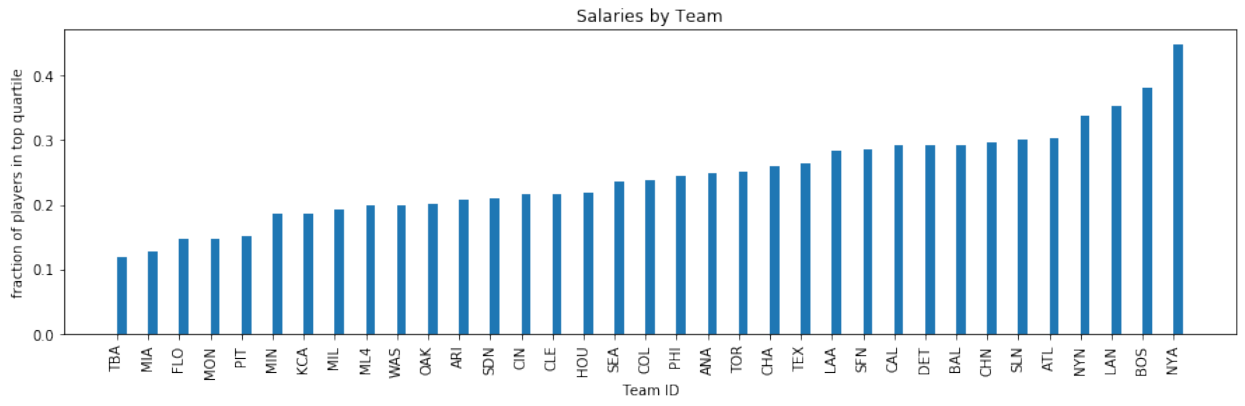
I wanted a way to control for the year, but also to make the salary feature binary so that I could use logistic regression and Support Vector machines. Therefore, I made each salary value 1 if it fell in the top quartile of salaries for a given year. If not, I labeled it a 0. I chose the top quartile because this range would could include elite players, not just all stars. By labeling just the highest paid players for each year, I was able to control for the year because my salary variable would now have an identical distribution for each year. In order to build my model solely on player performance, I dropped the year ID, team ID, and league ID.

Quickly I ran the decision tree classifier again to see which performance features were the most predictive of salary, now that it was controlled for by year.



I was surprised to find that RBI rate was significantly more predictive than hit rate, considering one is highly correlated to the other. Another interesting but perhaps unsurprising result is that the triple rate and the stolen base are almost identically predictive. This could be because both are quite related to how fast the player runs.

Another graph I was interested in was salary by team. I produced this result:

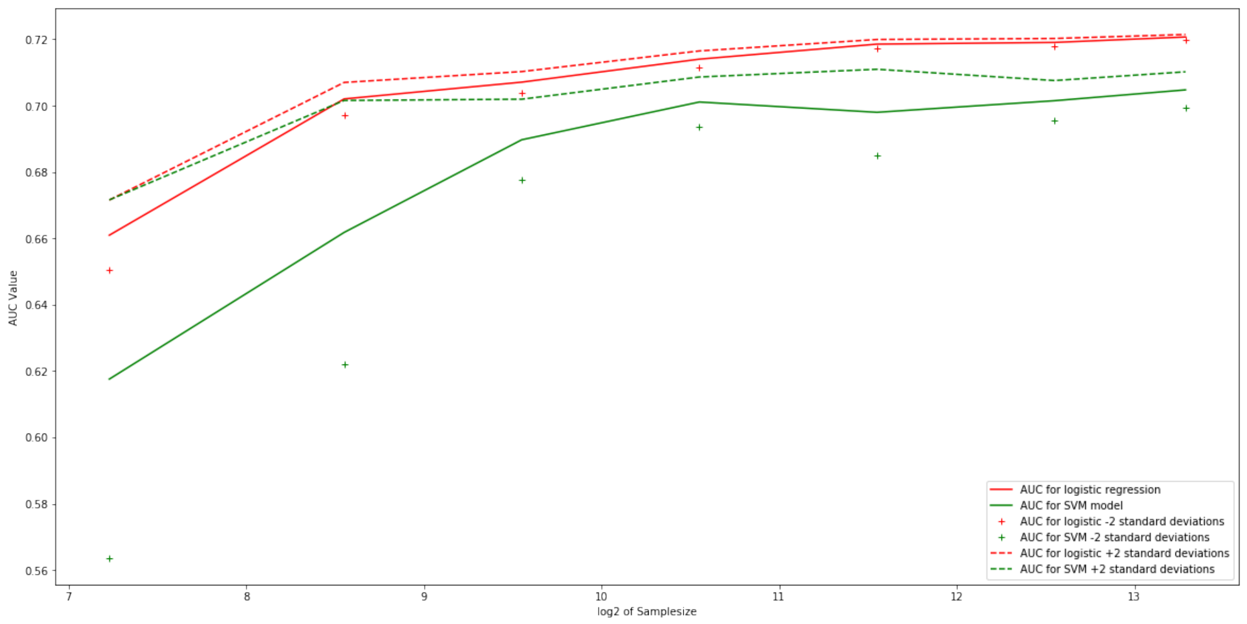


The teams with the highest fraction of players in the top quartile were the Yankees, Red Sox, Dodgers, Mets, and Braves, in that order. This ranking matched my perception of the highest-budget teams in MLB, justifying my assertion that the “top quartile” method is meaningful. Again, I was surprised to see that the Yankees had close to 4 times more top-quartile players than the Tampa Bay Devil Rays.

To move forward I wanted to know the difference in performance between an SVM and Logistic Regression model. I also wanted to see the minimum training set size I would need in order to get optimal performance on my model. I decided to bootstrap with different training set sizes and different model types to get a sense of the accuracies of each approach.

First, I created a function modBootstrapper that took in as its parameters a training set, a testing set, the number of repetitions per step, a sample-size, a target variable and an indicator of whether to use logistic regression or SVM.

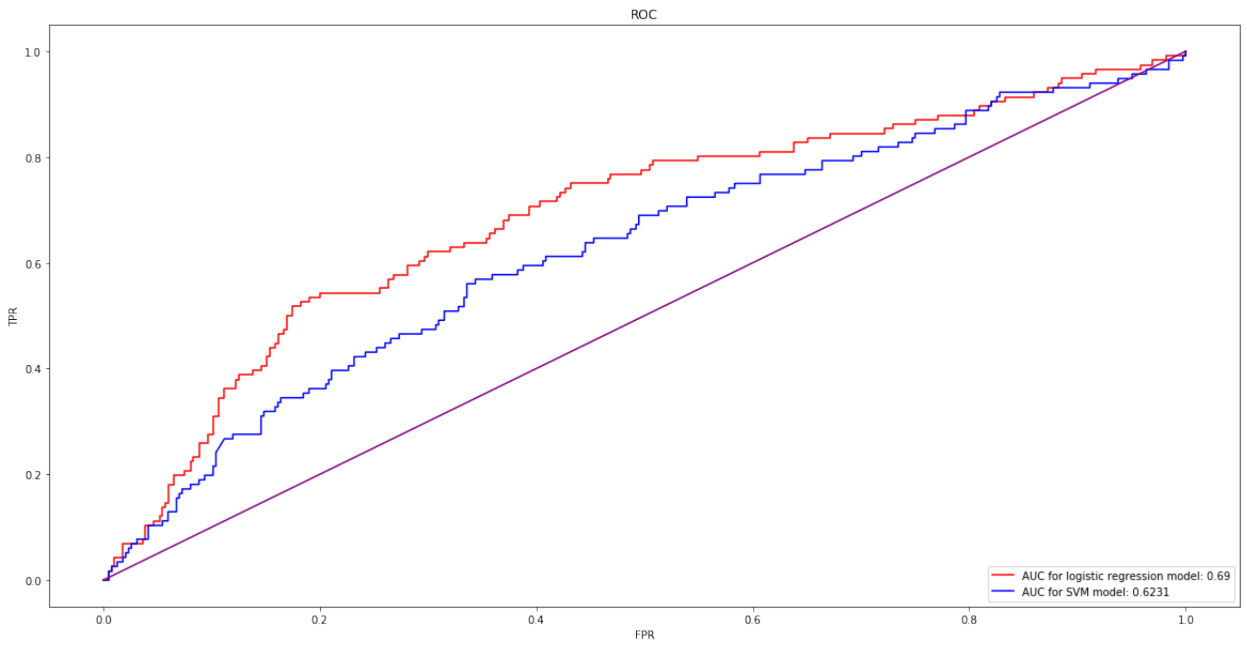
The size of my training data was about 11,000 records, so I made a list of sample-sizes that were exponentially approaching 11,000. (The sample size list was [150, 375, 750, 1500, 3000, 6000, 10000]). For each sample-size I made a training set of that size, and then trained a logistic regression model and a SVM model with it. I used default parameters for the logistic regression. For the SVM model, I used a linear kernel to save computation time. Then I ran my models on my training dataset. Using sklean’s metric.roc_curve function, I calculated the false positive and true positive rates of the prediction. I used the Receiver Operating Characteristic (ROC) curve and associated Area Under Curve (AUC) functions to identify the AUC value for each model. I repeated this process 20 times for each sample size and found the average and standard deviations of the AUCs of each. I made this graph of the bootstrapping process:



The red line represents the AUC value of the Logistic model as the sample-size doubles. The green line represents the AUC value of the SVM model as the sample-size doubles. The logistic model outperforms the SVM model for all sample-sizes, since it has higher AUC values. As we increase our sample-size, the performance of both models increases. However, the AUC values for the logistic model basically plateau when our sample-size gets to around 4000 (~2^12). Therefore I concluded that I would use the logistic regression model, and use a training set size of 4000. Any larger training set, and our accuracy would not greatly improve.

With a training set of size 4000, I built and trained a logistic regression and a SVM model. From our bootstrapping I knew the logistic regression would have a higher AUC than the SVM model, but I computed them both so that I could compare their ROC graphs. This is their

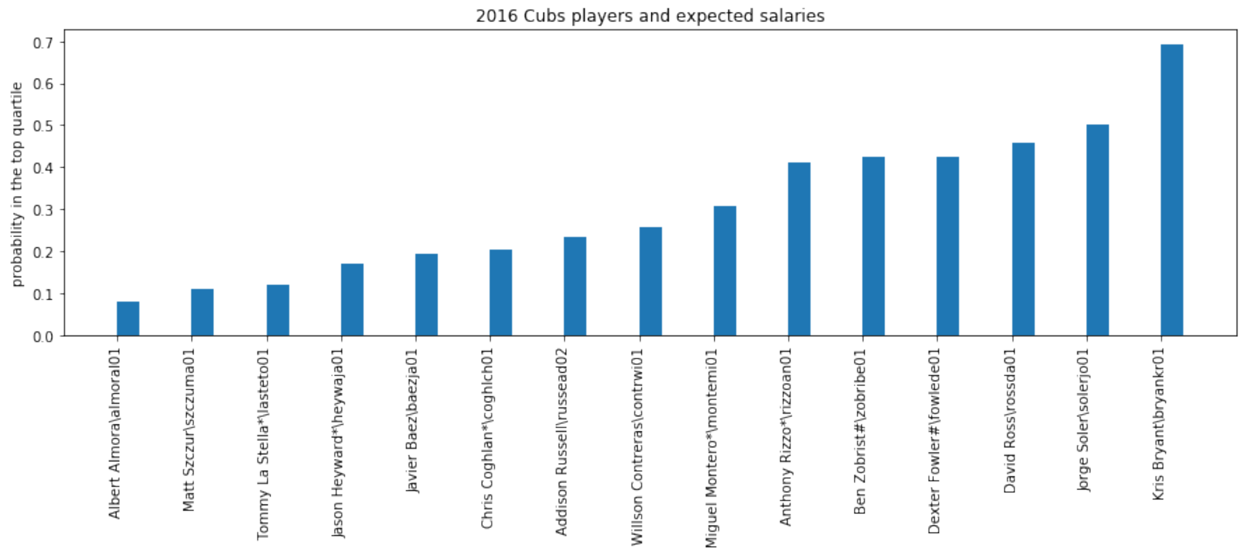
ROC curves:



As my bootstrapping suggested, the logistic regression had a higher AUC value (0.6900) than that of the SVM model (0.6231) so I settled on the logistic model.

Finally, I wanted to test my model on my favorite team, the Chicago Cubs. For obvious reasons, I chose 2016. I got my dataset from https://www.baseball-reference.com/teams/CHC/2016.shtml

For each Cubs player with at least 70 at bats in 2016, I found the probability that he would have a salary in the top quartile of major league baseball. Here were my results:



My model was in some ways expected and in some ways not. It predicted that Kris Bryant was by far the most likely to have a high salary. This seemed like reasonable choice, as he was NL MVP that year. But Jorge Soler came in 2nd place, whom I barely remember. Looking at the raw stats from 2016 he had pretty similar numbers to Miguel Montero, who had a significantly lower chance of making the top quartile according to my model. Soler also did better than Ben Zobrist and Anthony Rizzo, even though both of their batting averages and on base percentages were far better than his. The rest of the players fell in line approximately where I would have expected. The only explanation I have for the Soler outlier is that his caught stealing stat (which was 0) was much better than those of Zobrist or Rizzo, who both were caught stealing as much as they succeeded.