Springboard - DSCT
Capstone Project 2 - Milestone Report 2

David Buehler

December 2019

At this point I have finished the baseline modeling of my data set. This is a multi-class classification problem, and I decided on using Logistic Regression and Ridge Classifier models to model the data set. I decided to use these models for a few reasons, first and foremost this being a classification problem so I needed classification models. Logistic regression being the "base" classification model, then I took a look through scikit-learn's library of models and found the ridge classifier. This model seemed useful for the problem as it gave a different perspective on the data modeling. This model essentially took all the classes, turned them into numeric values then performed a multi-output regression model on the data set. It was useful to perform the modeling with two different classifiers, to see how differently they modeled the data and how accurate they were.

Before actually modeling the data, I needed to split up the data first. However this was not as trivial as just using train_test_split and that being adequate. I needed to keep the class representation of the main data set in the training and testing data for the coming modeling. Fortunately, train_test_split comes with the stratify keyword argument that does just that. I also needed to get all of the noise and features of the data that was not known before the pitch happened. These included all of the "id" columns, the object columns that were just categorical

info, and all other columns that happened after the pitch. This data was also able to be split up by the pitcher too, just by specifying the pitcher's first and last name.

After splitting the data, I needed to do some hyperparameter tuning. For the logistic regression model, the parameter of interest was the C value, and for the ridge classifier, the alpha value needed to be found. Since I was planning on looking at specific pitchers, these parameters needed to be specific for every pitcher. I ran a GridSearchCV to cross-validate and find the best parameter for each model and plugged those parameters into the model that I would eventually be fitting and predicting. Next I wanted to see how accurate these models were. I made another function that took in the best hyperparameters and put those into a logistic regression and ridge classifier model to fit the data and give me an accuracy score of the data. But with classification problems, using accuracy alone to assess the performance of classification models is not enough. I needed to use the classification report and confusion matrix to get a good gauge on how well the model was performing for each class. I was able to build a final function that did just this, that utilized all the splitting data and hyperparameter tuning from before, to fit and predict the data and build confusion matrices and give classification reports.

Results

First thing I did with my functions was run it on the entire data set. All 2.8 million data points, and got some interesting results. The logistic regression model came out to around a 74% accuracy on the data and the ridge classifier came out to around 61% accuracy. However looking deeper into the classification report, the logistic regression model did a decent job of

predicting all of the different types of pitches, except the “other” pitches. It’s f1-score was just 0, and considering there were only 1925 instances of “other” pitches in this testing set, it’s not surprising no predictions were made. Breaking balls had an f1-score of 0.5, with a precision of 0.75 and recall of 0.37. So this model did a good job of predicting fastballs, but with the recall being that low, it does not seem like the model was very sure of these predictions. Fastballs did better at 0.74 precision, 0.93 recall and 0.83 f1-score. This was by far the biggest class so the model was very sure of these predictions. Finally offspeed pitches had a precision of 0.72, a recall of 0.84, and an f1-score of 0.78. This was the third largest class at 87000 instances, so this model did a better job at picking up when offspeed pitches were thrown better than it did for breaking balls. For the ridge classifier classification report, it had a hard time picking up breaking balls, offspeed pitches, and other pitches, with a recall of 0.28, 0.01 and 0 respectively. For the majority class, fastballs, it did well with a 0.93 recall rating. But had the lowest precision rating at only 0.6. Overall the ridge classifier was worse than the logistic regression model, and that was a trend that continued as I started looking into specific pitchers.

After the two general models, I decided to look into specific pitchers around the MLB. I looked at 3 starters: Felix Hernandez, Justin Verlander, and Clayton Kershaw. As well as 2 closers, Edwin Diaz and Aroldis Chapman. What stood out right away was the difference in model accuracies between starters and closers. For starting pitchers, the model accuracies averaged out to about 84%. However for closers, the models were almost perfectly predicting what pitches they would throw. Starting with Hernandez, his logistic regression accuracy was 87% and ridge classifier accuracy at around 70%. Looking at the classification report though, the logistic regression model perfectly predicted all of his offspeed pitches, and did a good job

predicting his fastballs and breaking balls. Fastballs had an f1-score of 0.86 and breaking balls with an f1-score of 0.79. However, the ridge classifier was a similar situation to what we saw with the all pitchers model, not great scores on breaking ball and offspeed pitches, but did a good job predicting fastballs. Hernandez's breaking ball and offspeed pitch f1-scores were 0.35 and 0.67 respectively, with fastballs f1-score being 0.89.

Verlander and Kershaw were a similar story, with their logistic regression accuracy at about 80% and ridge classifier accuracy at about 74%. Even the trends in the classification reports were the same. Higher precision for breaking balls than for fastballs, lower recall for breaking balls than for fastballs, and breaking balls had a lower f1-score than fastballs. The only real difference between the two pitchers were offspeed pitches. Verlander threw a lot more so there was a larger sample size. The logistic regression model was able to predict all of those, whereas Kershaw hardly threw any offspeed pitches, so the model didn't predict any.

Moving on to the closers, this is where things got surprising. For both Edwin Diaz and Aroldis Chapman, the logistic regression model was nearly perfect when making predictions. With model accuracies for both pitchers at 99.5%. Even the ridge classifier did really well, with its accuracy hovering around 94%. There could be a few reasons for this. Both pitchers are closers, which means a much smaller sample size of pitches than starters. Also both pitchers almost exclusively throw fastballs and breaking balls, with both throwing about 2-3 times more fastballs than breaking balls. This would make it easy for models to pick up on trends and make the right predictions for these closers.

Having only looked at 5 pitchers, it's tough to see trends on good pitchers vs. bad pitchers and starters vs. relievers. However, Verlander and Kershaw have been two of the best

pitchers in the MLB since 2015, and Hernandez has been on the downslope of his career since that same year. So this analysis could be the beginning of a trend, but more modeling on more pitchers would need to be done to see it clearly. Also the sharp contrast between starting pitchers and relievers was unexpected. I didn't think relief pitchers would be that predictable, especially two of the better ones since 2015. But again, more pitchers would need to be analyzed to see if this trend continues.

From here I plan on getting into more in-depth modeling, including Random Forest Classifier, gradient boosting, k-nearest neighbors and whatever other models could potentially be good for imbalanced classification problems. After looking at what sci-kit learn has to offer, I plan on getting into some neural networks and potentially deep learning. I think this will give a wide variety of perspectives and models on this problem and will give me the best chance to find a suitable model for predicting pitches.