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. Both models came out to around only 56% accuracy, which isn't very good at all. However looking deeper into the classification report, the logistic regression model did a decent job of predicting fastballs, and that was about it. With a recall of

0.95, precision of 0.57, It's f1-score was 0.71. Considering this was the majority class, by a wide margin, it is unsurprising it did the best on this class. Breaking balls had an f1-score of only 0.14, with a precision of 0.42 and recall of only 0.08. So this model did a good job of predicting fastballs, but that was about it. Both offspeed pitches and other pitches were not predicted at all, with zeroes in every category of the classification report. Ridge classification had a similar trend too, with the only difference being a 0.09 recall for breaking balls and 0.74 recall for fastballs. Otherwise everything else was zeroes. Overall both models did not do well when predicting pitches, 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 pitchers. Starting with Hernandez, his logistic and ridge classification accuracy were both at 47%. Looking at the classification report though, both models did a good job predicting his fastballs, with an f1-score of 0.91 on the logistic regression and 0.90 on the ridge classifier. However, much like the overall data set, it did not do well on any of the other pitches. Though unlike the whole data set, it was actually able to make predictions on the offspeed pitches, and actually did better on those than the breaking balls. The f1-score of 0.19 for the offspeed pitches beat the 0.11 f1-score for the breaking balls.

Verlander and Kershaw were a similar story, though they saw slightly different results. Looking at these two pitchers has made a pattern clear. What pitches the model predicts seems to be a function of how closely related the class representation is. For example, Clayton

Kershawn in his testing set, the number of fastballs and breaking balls he threw were almost even (1260 and 1295 respectively). Both models had an f1-score of 0.61 on both types of pitches. So those were predicted quite close to one another, with the next largest class, offspeed pitches (at 13) was not predicted at all. Now looking at Justin Verlander, his testing set had 1880 fastballs and 1135 breaking balls. His model did much better on fastballs, with an f1-score of 0.73, than it did on breaking balls, f1-score of 0.31. Felix Hernandez had a similar story, with his breaking balls and offspeed number at around the same, and the f1-scores were around the same. This trend extended into closers as well.

Moving on to the closers, we see a similar trend as we saw with the starters. The majority class would always be predicted better than the minority classes. For Aroldis Chapman, his test set hat 754 fastballs, 187 breaking balls, and 40 offspeed pitches. His f1-score on his fastballs was 0.87, and zeroes for breaking balls and offspeed. He threw a disproportionate number of fastballs compared to his breaking balls and offspeed pitches. Edwin Diaz had a smaller split between his fastballs and breaking balls. With 519 fastballs and 266 breaking balls in his testing set. With Diaz, we see an f1-score on his fastballs of 0.79, and breaking balls a 0.36 f1-score. It seems like as the proportions of pitch types get closer to one another, the models have a better time predicting more classes, but if there is a disproportionate number of pitches compared to the others (like Chapman had), the models will have a harder time picking out the minority classes. A table to summarize all of this data is below.

Pitcher	FB Support	BB Support	OS Support	FB f1-score	BB f1-score	OS f1-score
Felix Hernandez				0.91	0.11	0.19
Justin Verlander	1880	1135		0.73	0.31	0
Clayton Kershaw	1260	1295	13	0.61	0.61	0
Edwin Diaz	519	266		0.76	0.36	0
Aroldis Chapman	754	187	40	0.87	0	0

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 majority vs. minority class problem has become apparent. With majority classes seemingly being favored by the models while minority classes are not predicted as well. This is certainly not a common problem, as all imbalanced classification problems suffer from this.

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. In response to how imbalanced this classification problem is, I will also get into under and oversampling the data set with the model that best predicts the original data set.