MATLAB final report

Andrew Moss

May 5, 2020

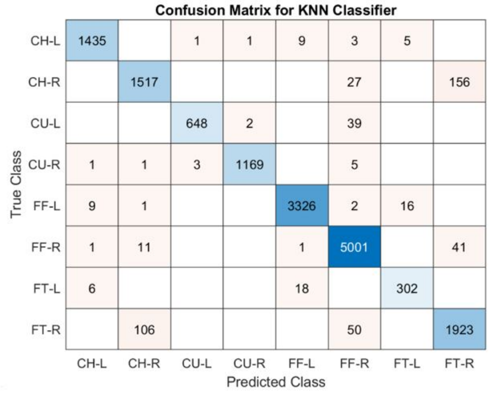
1 Background and Statement of problem

It's not surprising I ended up of the sports analytics field out of college. As long as I've been able to read, I would pick the sports section out of the newspaper to look at the charts and tables. As I grew up, I was always involved in playing and watching sports. At the beginning of March, I was ecstatic to find out I had landed a job as an analytics apprentice with the Texas Rangers. However, I had never done a full project or used publicly available baseball data to try to back up a claim. While I had done some investigations into predicting aspects of football and basketball, it made sense to try to gain some experience working with baseball data.

The problem is decided on was to use Statcast pitch data (publicly available on statcast.com) to try to predict not only the pitch that was thrown by the pitcher, but also which side of the mound the pitcher was throwing the ball from. Statcast data involves physics related figures about the pitch (acceleration, movement, velocity) as well as pose detection for where the pitcher's limbs were when he threw the ball I wanted to practice my Statcast query skills as well as start with a more simplistic model so I can understand how the parameters and predictors are impacting the results. For this model, I queried 4 pitches (so 8 classes in total). They were, for left- and right-handed pitchers, 4-seam fastballs, 2-seam fastballs, curveballs, and changeups. The dataset I chose to try to make this prediction was all Rangers pitches during the 2019 that met the criteria for pitch type. In all, there were 15836 pitches with 1454 left-handed changeups, 1700 right-handed changeups, 689 left-handed curveballs, 1179 right-handed curveballs, 3354 left-handed 4-seam fastballs, 5055 right-handed 4-seam fastballs, 326 left-handed 2-seam fastballs, and 2079 right-handed 2-seam fastballs. The uneven nature of the class sizes would present their own problem throughout the modeling process.

2 Solution Methods and Results

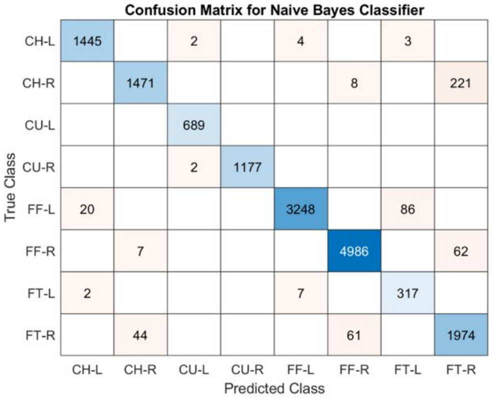
To solve this problem, I trained three different common classifiers, but I wanted to use classifiers from different parts of statistics/machine learning so I could analyze the differences in how each type of model would contend with the inputs. The three models were K-nearest neighbors (a modeless classifier), Naïve Bayes (a Bayesian Model), and a decision tree (supervised learning model that uses the maximum likelihood expectation). The inputs into the model were the following. The x,y, and z coordinates of the release point of the ball; the x and z locations of where the pitch was received by the catcher; the velocity and acceleration of the ball in the x, y, and z directions; the effective speed of the pitch; the release extension by the pitcher; and the spin rate of the ball. After training all three models with these predictors and performing 10-fold cross validation, the k-nearest neighbors model (using 1 neighbors) was 96.75 percent accurate, the Naïve Bayes model was 96.66 percent accurate, and the decision tree model was 97.4 percent accurate.



Confusion Matrix for KNN Classifier

True Class
Predicted Class

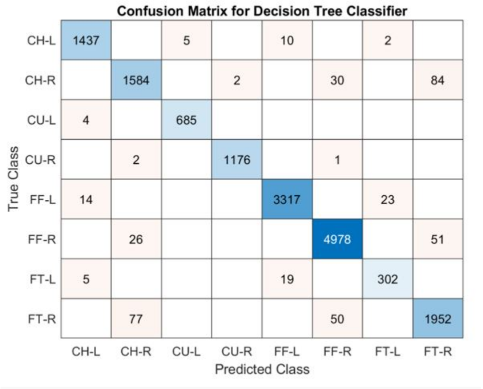
	CH-L	CH-R	CU-L	CU-R	FF-L	FF-R	FT-L	FT-R
CH-L	1435		1	1	9	3	5	
CH-R		1517				27		156
CU-L			648	2		39		
CU-R	1	1	3	1169		5		
FF-L	9	1			3326	2	16	
FF-R	1	11			1	5001		41
FT-L	6				18		302	
FT-R		106				50		1923



Confusion Matrix for Naive Bayes Classifier

True Class
Predicted Class

	CH-L	CH-R	CU-L	CU-R	FF-L	FF-R	FT-L	FT-R
CH-L	1445		2		4		3	
CH-R		1471				8		221
CU-L			689					
CU-R			2	1177				
FF-L	20				3248		86	
FF-R		7				4986		62
FT-L	2				7		317	
FT-R		44				61		1974

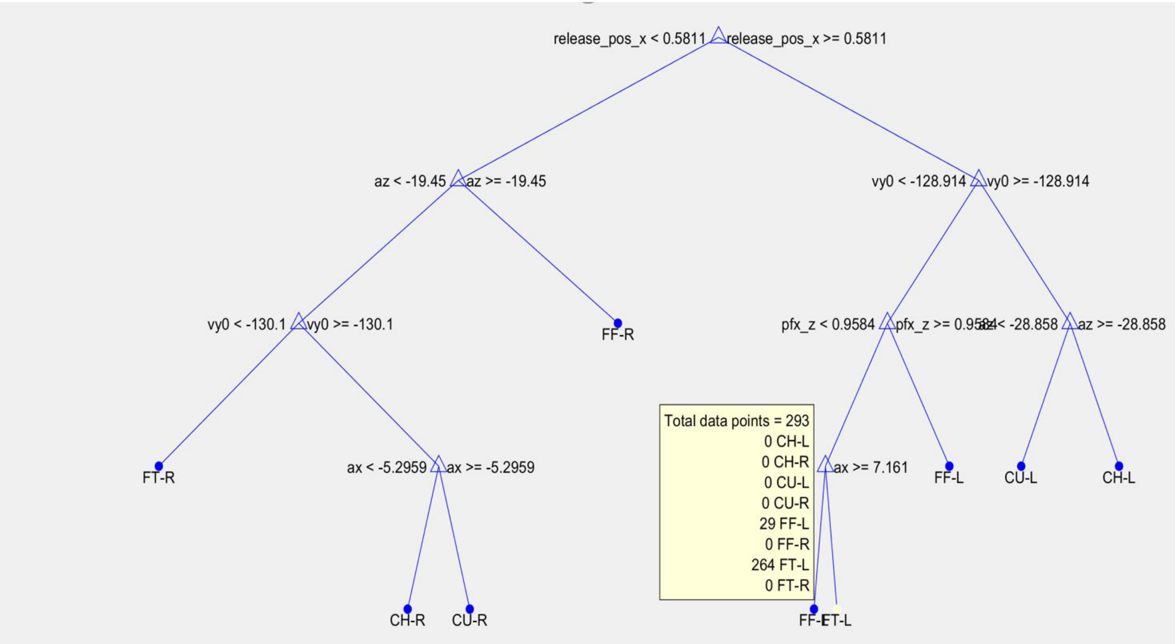


Confusion Matrix for Decision Tree Classifier

True Class
Predicted Class

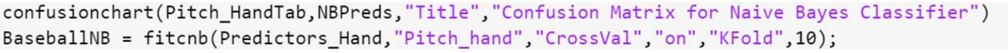
	CH-L	CH-R	CU-L	CU-R	FF-L	FF-R	FT-L	FT-R
CH-L	1437		5		10		2	
CH-R		1584		2		30		84
CU-L	4		685					
CU-R		2		1176		1		
FF-L	14				3317		23	
FF-R		26				4978		51
FT-L	5				19		302	
FT-R		77				50		1952

Even though all of the models were fairly accurate, the nature of the misclassifications made me think that the Naïve Bayes and decision tree were better than the KNN model. In baseball terms, a 4-seam fastball is most likely to look like a changeup or a 2-seam fastball of a pitcher of the same handedness. When KNN misclassified, it would sometimes seem random as to what KNN misclassified the pitch as. When one of the other two models misclassified, it made baseball sense as to why it made that choice for the classification. An abridged version of the decision tree's logic can be seen below.



3 Reflection, additional learnings, and challenges

The MATLAB features I had to learn that was outside the scope of this class was using the machine learning toolkit. I had to learn how to train the different classifiers and use the pairwise parameter inputs. I also had to figure out how to use to built-in cross validation feature when training my classifier. The final MATLAB rookie-related challenge I faced was massaging the strings of predictions and actual class labels in a confusion matrix. The most brainpower-intensive part of this project was analyzing the model through the MATLAB output. In addition to accessing the different objects within the model, I had to think statistically as to what the results I was getting meant in statistical and baseball terms.



```
confusionchart(Pitch_HandTab,NBPreds,"Title","Confusion Matrix for Naive Bayes Classifier")
BaseballNB = fitcnb(Predictors_Hand,"Pitch_hand","CrossVal","on","KFold",10);
```