GAUSSIAN NAÏVE BAYES

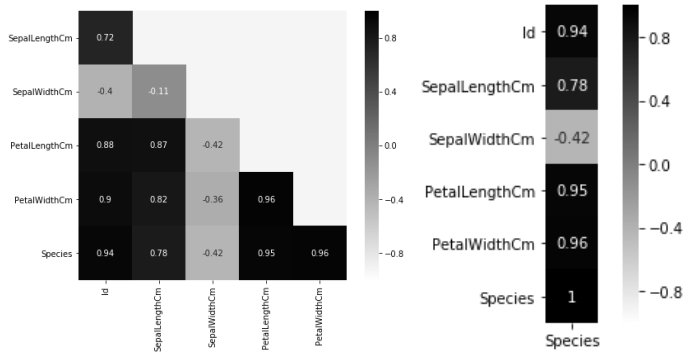
The Code

The model is given a train set and a sample from a test set, with the goal of predicting the class value of the test sample. The train set is split into a dictionary of class value keys, and rows with that class value. This is used to create a dictionary of class value keys, and statistics about the rows with that class value – mean, standard deviation, and number of rows. The probability of each class value is calculated, because the one with the highest probability becomes the predicted value. Laplace smoothing is implemented to avoid the issue of zero probabilities, and log probabilities are implemented to avoid underflow.

There are helper functions to perform math calculations and analyze the results of the model.

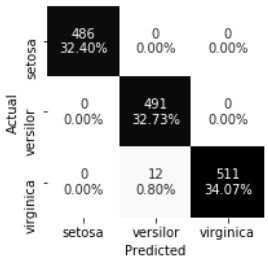
Data Analysis

The correlation matrix of the dataset was analyzed with heatmaps and the analysis function from my personal python package.

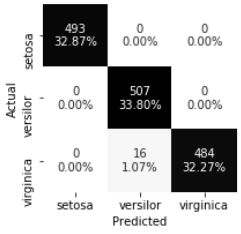


There are so few features, I was hesitant to drop any, but I compared the results of both options.

All features: Accuracy of 99%



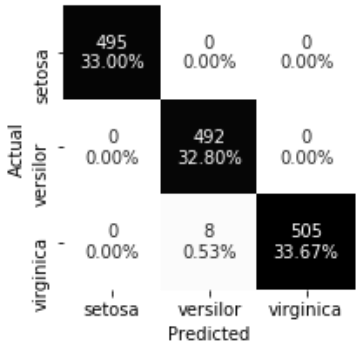
Only high correlation features: Accuracy of 98.93%



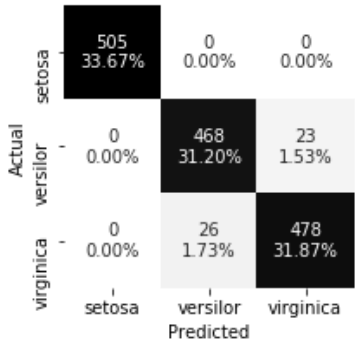
They both had high accuracy, but using all features was consistently better.

I also compared the results of all features, and dropping “Id”, only because I typically see a highly correlated “Id” feature.

All Features: Accuracy of 99.47%



All Features Except Id: Accuracy of 96.73%

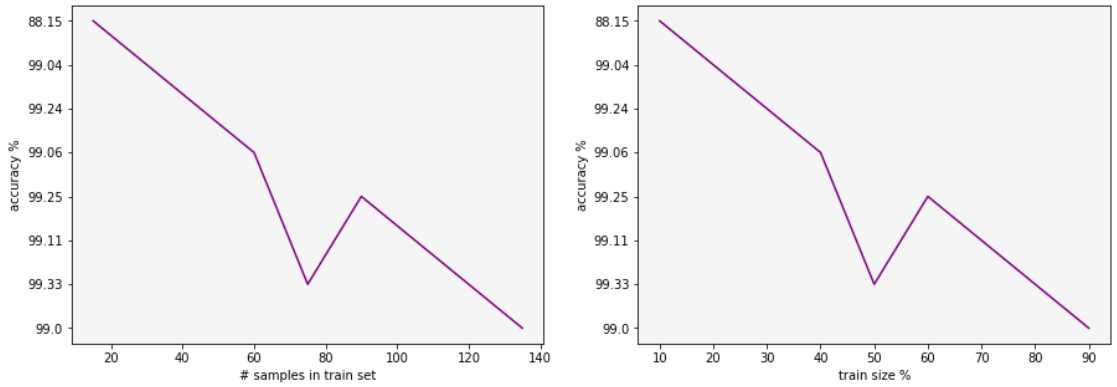


As expected, there was a significant drop in accuracy when “Id” was excluded. I think this is because the dataset is ordered and organized by type already.

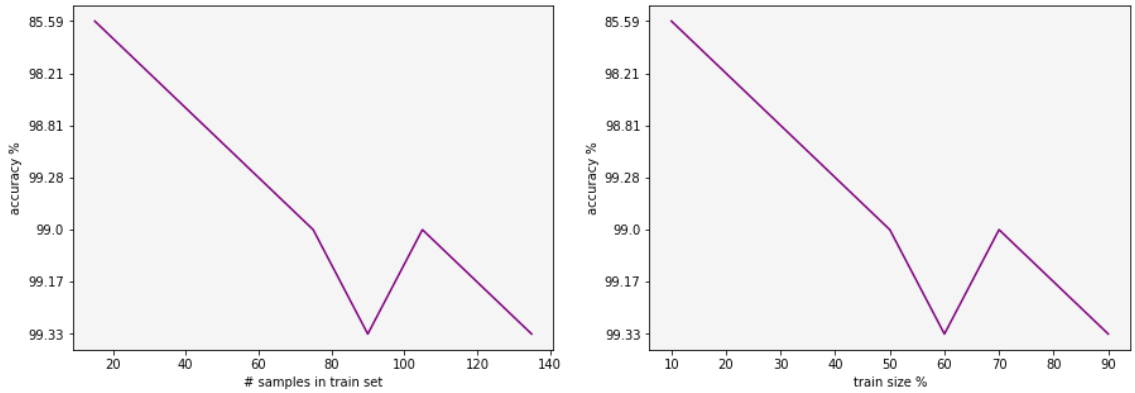
Model Analysis

The accuracy of my model and Sci-kit Learn’s model were evaluated with a range of train/test splits.

My Algorithm



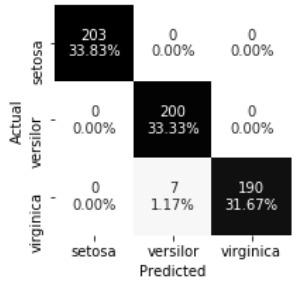
Sklearn Algorithm



Both algorithms had a comparable correlation between accuracy and train size. Accuracy improved when the number of samples in the training set increased - however, this is probably due to overfitting, so a reasonable balance has to be chosen. I think 80% train, 20% test is a fair split.

Final Evaluation

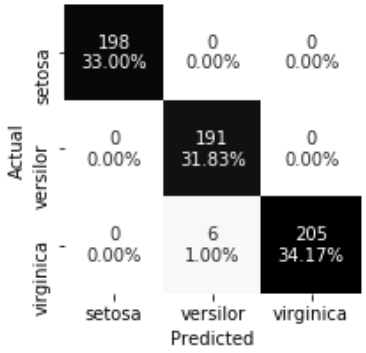
My Algorithm: Accuracy 98.83%



Classification Report:

	precision	recall	f1-score	support
0.0	1.00	1.00	1.00	203
1.0	0.97	1.00	0.98	200
2.0	1.00	0.96	0.98	197
accuracy			0.99	600
macro avg	0.99	0.99	0.99	600
weighted avg	0.99	0.99	0.99	600

Sklearn Algorithm: Accuracy 99%



Classification Report:

	precision	recall	f1-score	support
0	1.00	1.00	1.00	198
1	0.97	1.00	0.98	191
2	1.00	0.97	0.99	211
accuracy			0.99	600
macro avg	0.99	0.99	0.99	600
weighted avg	0.99	0.99	0.99	600

Both algorithms are very accurate, with my algorithm being consistently slightly more accurate. Versicolor was the most incorrectly predicted.

It is hard to truly evaluate overfitting with this small of a dataset. However, the fact that the accuracy was still very high (over 95%) no matter the train/test split, I believe this model is truly accurate.