Homework 3 Writeup

Tylman Michael
CSE 546 Machine Learning

2/18/2023

1 Cross-Validation Optimal Parameters and Testing

1.1 Prelude

For this homework, I set out to better understand the GridSearchCV functionality of sklearn and how to utilize it for deeper analysis. My reasoning was that the burden of proof for professional/research applications of a GridSearchCV must be higher than the burden of proof for conclusions required for a class. If that's the case, and GridSearchCV was likely made for use in a professional/research setting like the rest of sklearn, then it must be possible to do quality analysis using this function. So, for every item listed in here, I am only doing operations on either the GridSearchCV, or a single estimator pulled from the GridSearchCV.

I will also list a single table with the best results (with the param lists cut) for every model for ease of reference in Table 1. The most important rows to recognize are the last few, where best_final_test is the accuracy of the best model on the test set, best_cv_test and best_cv_train are the average test accuracies of the best model across the CV splits for test and train respectively. A couple of those values are available earlier in the table as well, but I included them at the all together for ease of use.

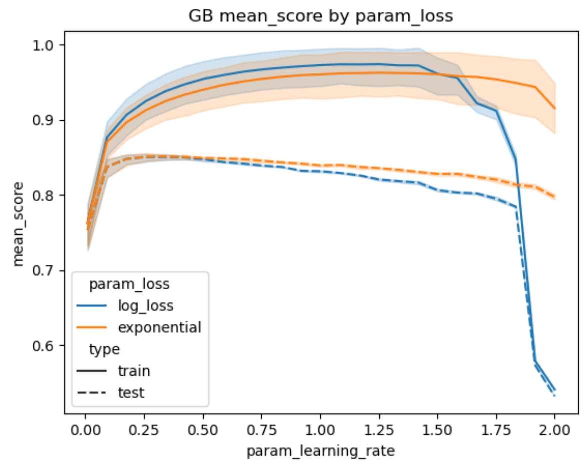
1.2 Gradient Boosted

For the Gradient Boosted (GB) classifier, I gridsearched combinations in the learning rate, loss function, and number of estimators. For the learning rate, I did a linspace of (.01, 2, 25). For the estimators I went from 100 to 700 in steps of 100. And for the losses I chose log loss or exponential options. The resulting performances is shown in Figure 1

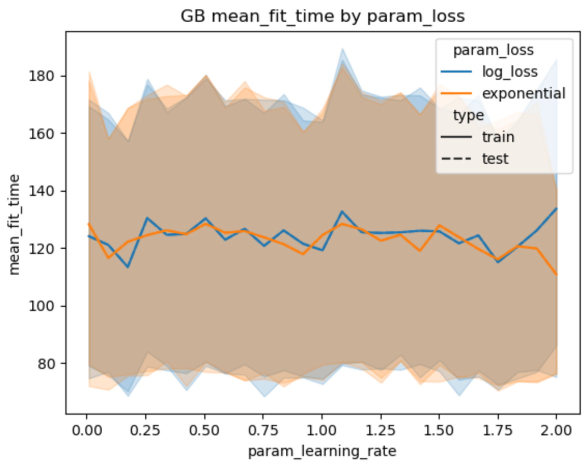
The shaded regions in the two plots show that learning rate had a much stronger influence on the performance of the model than the number of estimators. Interestingly, the training performance of the two loss functions actually inverts when we group by estimators or learning rate. In both cases; however, the exponential bests the log loss function in the test case.

	gb	knn	rf	nb
Index	47	33	249	8
mean_fit_time	208.3647250533104	0.4263809323310852	75.5146564245224	0.8436596393585205
std_fit_time	3.0957062399871327	0.0592272222690093	4.017492669496358	0.03875597162913965
mean_score_time	0.3202803134918213	13.931629002094269	5.75344318151474	0.09340763092041016
std_score_time	0.007948119349096923	0.13358312729265626	0.21164291916351288	0.011340124984171198
params	{'learning_rate': 0.25875, 'loss': 'exponentia...	{'n_neighbors': 12, 'weights': 'uniform'}	{'max_depth': 70, 'max_features': 'log2', 'n_e...	{'alpha': 0.21161616161616165, 'fit_prior': True}
split0_test_score	0.85504	0.65984	0.8448	0.82928
split1_test_score	0.85888	0.66496	0.84832	0.83392
split2_test_score	0.85296	0.66048	0.84208	0.82896
split3_test_score	0.85904	0.64512	0.84944	0.83776
mean_test_score	0.8564799999999999	0.6576	0.84616	0.83248
std_test_score	0.002587353860607377	0.007470475219154384	0.0029120439557121756	0.003625686141959861
rank_test_score	1	1	1	1
split0_train_score	0.94736	0.7285333333333334	0.9999466666666667	0.83824
split1_train_score	0.9471466666666667	0.7330666666666666	0.9999466666666667	0.8366933333333333
split2_train_score	0.9459733333333333	0.74576	1.0	0.83744
split3_train_score	0.9439466666666667	0.73072	0.9999466666666667	0.83616
mean_train_score	0.9461066666666667	0.7345200000000001	0.9999600000000001	0.8371333333333333
std_train_score	0.0013542361520634087	0.006684496158192375	2.3094010767592102e-05	0.0007841768508017322
best_final_test	0.86208	0.66304	0.84412	0.83236
best_cv_test	0.8564799999999999	0.6576	0.84616	0.83248
best_cv_train	0.9461066666666667	0.7345200000000001	0.9999600000000001	0.8371333333333333

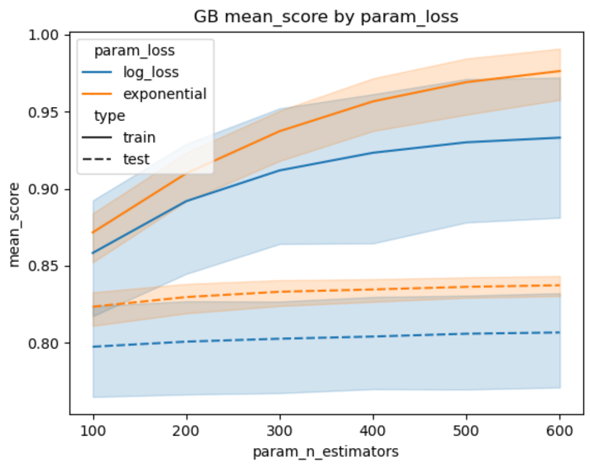
Table 1: Best Results and MetaData



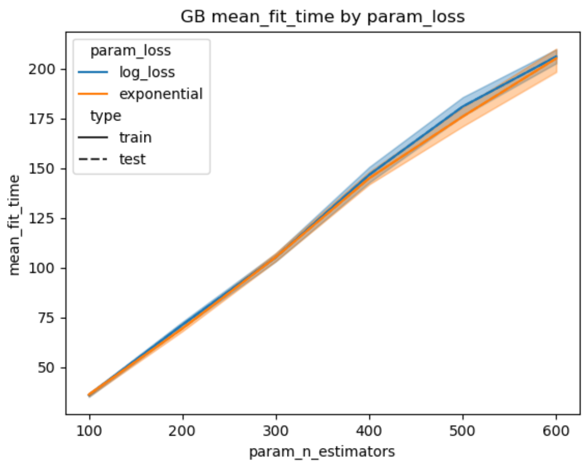
(a) Performance VS learning Rate



(b) Time VS learning Rate

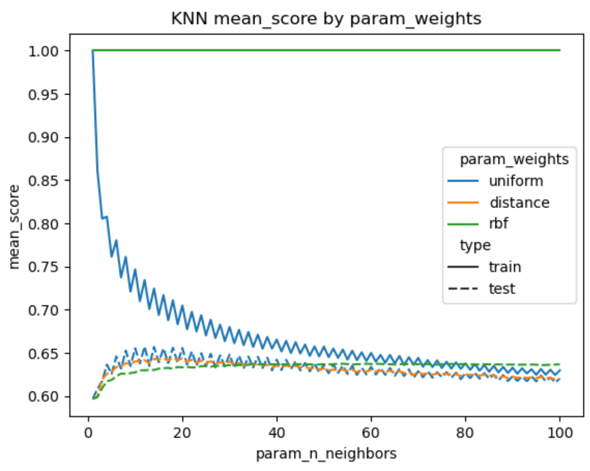


(c) Performance VS Estimators

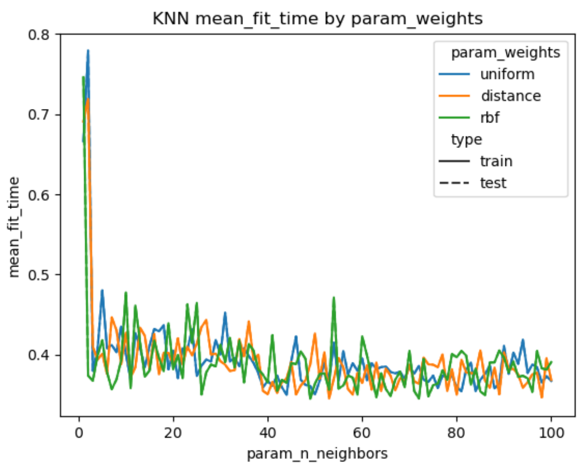


(d) Time VS Estimators

Figure 1: GB Performance and Time



(a) KNN Performance VS N Neighbors



(b) KNN Time VS N Neighbors

Figure 2: RF Performance

The best parameters for all of the parameters were: 'learning_rate': 0.25875, 'loss': 'exponential', 'n_estimators': 600 which gave the results we can see in Table 1. The final test accuracy was very slighly higher than the average cross validation test performance, but by only a small margin. This model appears to have generalized very well, and shows only a little bit of overfitting at this stage. Interestingly, the choice in loss function appears to have directly helped overfitting by lowering the training score and raising the testing score.

Another interesting phenomenenon is that the testing score appears to be much more stable than the training score for both loss functions, with the exponential loss function being more stable than the log loss function when we group by the number of estimators.

The GB model performed quite well, but I have to acknowledge the training time. I will cover this in more detail in the conclusion where I decide the best model, but the GB model trained exceptionally slow when compared to the other models. As expected, the learning rate had no discernable effect on the training time of the GB model, and the training time scaled linearly with the number of estimators used. However, the slope of this line is the most significant by far of any of the other models.

1.3 KNN

For the KNN classifier, I gridsearched combinations in the Distance metric and the number of neighbors. The number of neighbors varied from 1 to 100, and the Distance metric could be 'uniform','distance' or a custom Radial Basis function. I did not also loop over the possible parameters for the rbf, since including it was a bonus in it's own right. The resulting performance and time complexity is shown in Figure 2.

This plot shows that the distance and radial basis functions drastically increased training accuracy to be 100% at all values, but did not have a dramatic impact on test performance. This leads me to the conclusion that the non-uniform weighting increases the complexity of our models which opens us up to overfitting.

Interestingly, the uniform parameter showed a stairstep behavior where the even number neighbor counts outperformed the next and previous odd-number amount of neighbors. That's something that caught me by surprise, and I can't quite explain. This behavior worked in the favor of the uniform distribution, allowing it to peak higher than the other weights. So, the best parameter for the weight was the uniform option, which gave better test scores and showed less overfitting.

The best parameter for n_neighbors was less intersting. We can see that it peaks pretty early in the low 10s, with the best parameter being achieved at n_neighbors = 12, which gave us our final results shown in 1.

The best KNN model generalized well, with the accuracy on the test group being quite close to it's average test performance on the validation splits during training. It did perform noticeably higher on the training set, but not to a degree which would be indicative of oppressive over-fitting. All in all, I'd say the KNN model performed quite well, but trails behind the GB model in performance.

The time complexity for the KNN model is uninteresting, but I included the plot here for consistency with the other models. Since the KNN model is so simple, we don't have many conclusions to make regarding the time complexity, except when compared to other models.

1.4 Random Forest

The Random Forest (RF) classifier was gridsearched over the options: max depth, number of estimators, and max features. The maximum depth went from a range of 10 to 100 with steps of 10, and I also appended a 'None' option which did not limit the max depth at all. The number of estimators used a range of 100 to 1000 with a stepsize of 50. Finally, the number of features was selected from the sqrt and log2 options.

We can see from Figure 3 that the log2 feature selection outperformed the sqrt feature selection at almost every step. Interestingly, we can see much more consistency when we plot the score vs max depth instead of the estimators, even though the max depth has more items it is averaging over. I believe that is evidence to suggest the maximum depth has a stronger influence on the performance than the number of estimators.

The best parameters for the RF classifer were: 'max_depth': 70, 'max_features': 'log2', 'n_estimators': 850, which makes sense given the positive trend we are seeing in both of these plots. Unfortunately, if you look at Figure 3: (a) you'll see that the testing accuracy actually plateaus much faster than the training accuracy, and that this selection is well into the region where we would consider it to be maximum overfitting of the model since the training accuracy is 1.

However, even though the training accuracy is 1 while the validation accuracy is only .846, we do see that the random Forest still generalized well to give us a final test accuracy of .844. While we might be tempted to say this is a victory, if we look at the greater context of all our models, we can see that this is the only model which showed noticeable loss in performance when transitioning to the test set. Two of our models actually increased performance during the switch, and the other model only lost .00012 performance, which is entirely neglible. While the loss of performance for the RF is quite small, and likely neglible, I think it isn't an accident that the model which shows the most signs of classic overfitting is the model which shows the largest drop in performance by a wide margin.

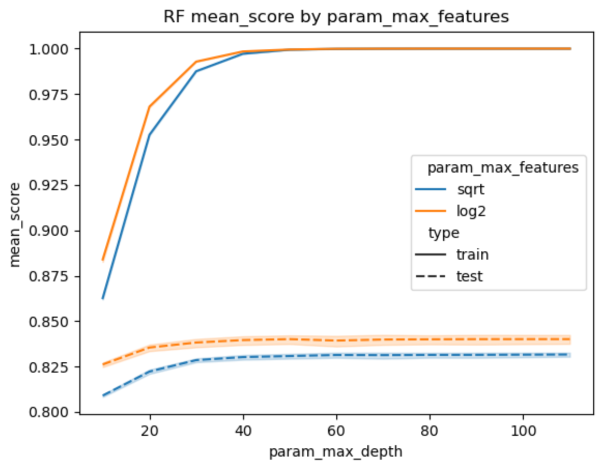
The time complexity of the RF model is very satisfying. The time complexity grew logarithmically with max depth, and linearly with the number of estimators. For both of these, the log2 feature selection is better than the sqrt feature selection, which is as expected. I want to point out that the absolute worst performing time complexity of the RF with a number of estimators of 800 is still better than the GB classifier with 600 estimators.

1.5 Naive Bayes

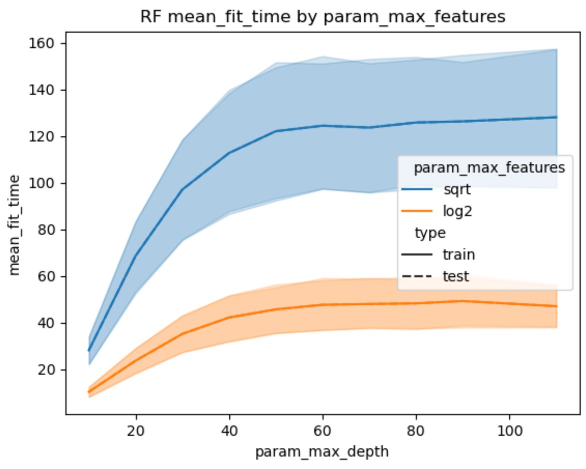
Before we continue, I must declare my bias towards the Naive Bayes (NB) classifier. I find this model to be mathematically pleasing, easy to understand, elegant, and efficient. If I have two classifiers performing similarly, and one of them is a NB classifier, I will almost always pick the NB classifier.

I used a more simple gridsearch for the NB classifier which altered the fit_prior option between False and True, and ranging alpha from .01 to 5. Interestingly, the fit_prior option appeared to do nothing for this dataset, and no option significantly effected the training time. From those two things, the available analysis of the NB classifier is quite dry compared to the others.

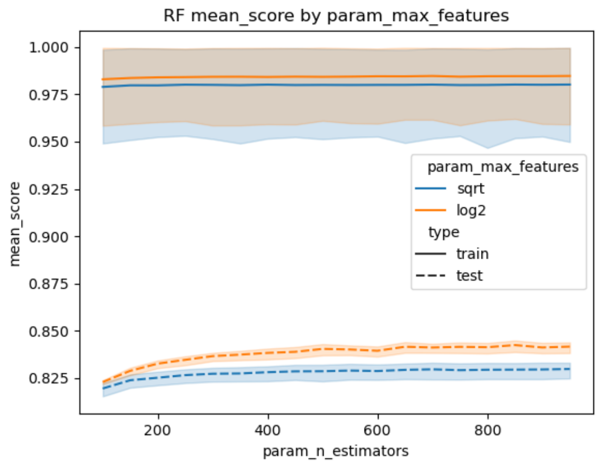
The best parameters for the NB classifier are: 'alpha': 0.2116, 'fit_prior': True. The final performance of the NB classifier showed absolutely zero overfitting, and nearly perfect generalization. The CV training accuracy was .837, CV test accuracy was .832, and the final tets accuracy was .832. Furthermore, it did all of this while being trained in a time comparable to the KNN classifier. This result is beautiful and evidence of why I have such a strong bias toward the NB classifier.



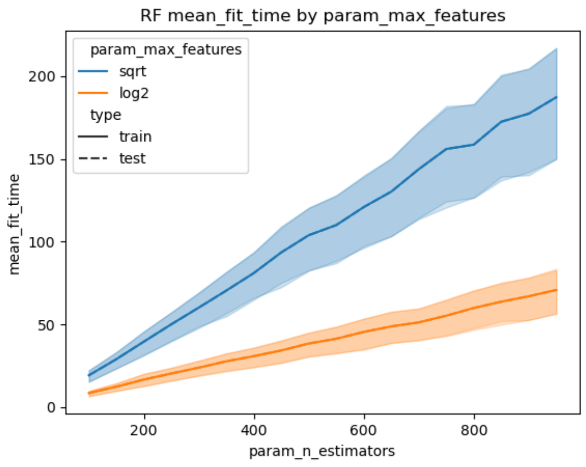
(a) Performance VS Max Depth



(b) Time VS Max Depth

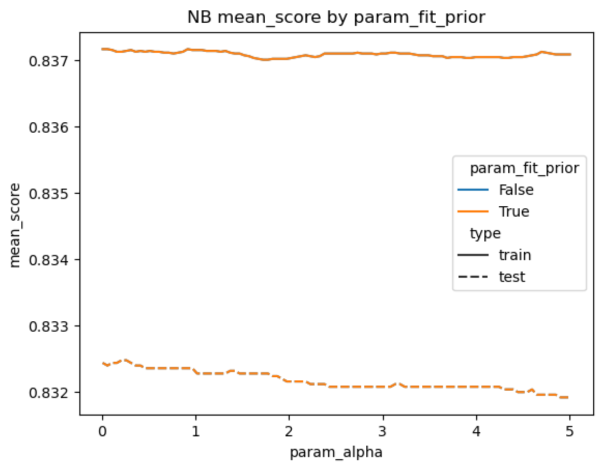


(c) Performance VS Estimators

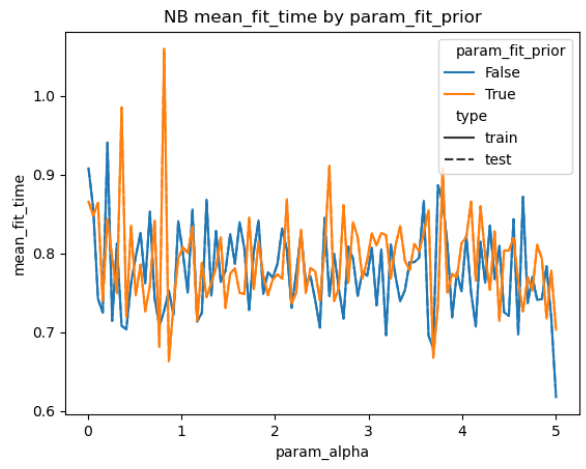


(d) Time VS Estimators

Figure 3: RF Performance and Time

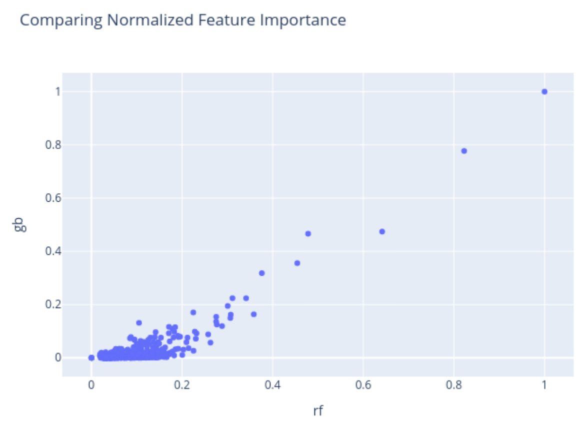


(a) Performance VS Alpha

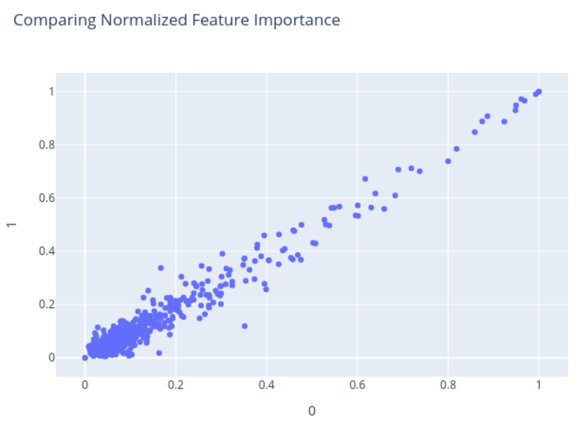


(b) Time VS Alpha

Figure 4: RF Performance



(a) Tree feature importance comparison



(b) NB feature likelihood comparison

Figure 5: Feature comparison

2 Feature Importance and Missclassifications

I decided it would be most useful to compare the feature importance between the RF and GB classifiers, and the positive/negative importances of the NB classifier. I'll include these results with an interactive .html plot in my submission if you'd like a closer look.

For the tree-based items, they both agree that Feature 78 is the most important feature by far, with several other features sharing similar importance. The GB model was much more likely to crush feature importance down to zero than the random forest, causing our scatter plot to be skewed below the line y=x.

For the NB classifier, there is one major outlier of Feature 3. Feature 3 the least important feature of any of the models, scoring a 0 importance fot the GB and RF models. All models agree feature 3 is apparently worthless, by a significant margin compared to the other features. The major reason why Feature 3 confuses me so much is that the readme states they took the 1000 most frequently used words, and looking at the likelihood plot of the NB classifier enforces the idea that they were sorted at least weakly by their usage. The interactive .html file shows that the items with the highest agreement between positive and negative samples are the earlier features. This explains why the trees do not show these items to be necessarily important, but it is odd to me that feature 3 appears to be entirely unused.

I did not attempt to analyze missed samples from these Three Models. since the data is obtuse to human comprehension. I.E. I have no idea what feature 78 is supposed to be, or why it was so important. Likewise, I can't tell you meaningfully why one sample was missclassified over another without the conclusion being "it didn't fit the learned distribution," which we already know. I have no way of providing a meaningful discussion of *why* it didn't fit the distribution without knowing what the features should be.

For these same reasons, I decided to look at the missclassified samples as a whole and try to understand the behavior. So, in Figure 6 I took the average distance from all neighbors for each sample, and sorted them from shortest distance to longest distance. The resulting plot shows a very interesting behavior. It seems like we don't start seeing a significant amount of misses until over double the distance of the very first missed item. Likewise, the largest outliers see a significant increase in the rate of growth of their average distance. I believe this is a sign of a healthy model where the most significant offenders are more likely to be outliers than well-clustered samples.

3 Conclusion and Final Model Selection

My selection for the best model is the Naive Bayes classifier. I understand that the Random Forest and Gradient Boosted classifiers have better performance, but both of them come at a cost. The GB

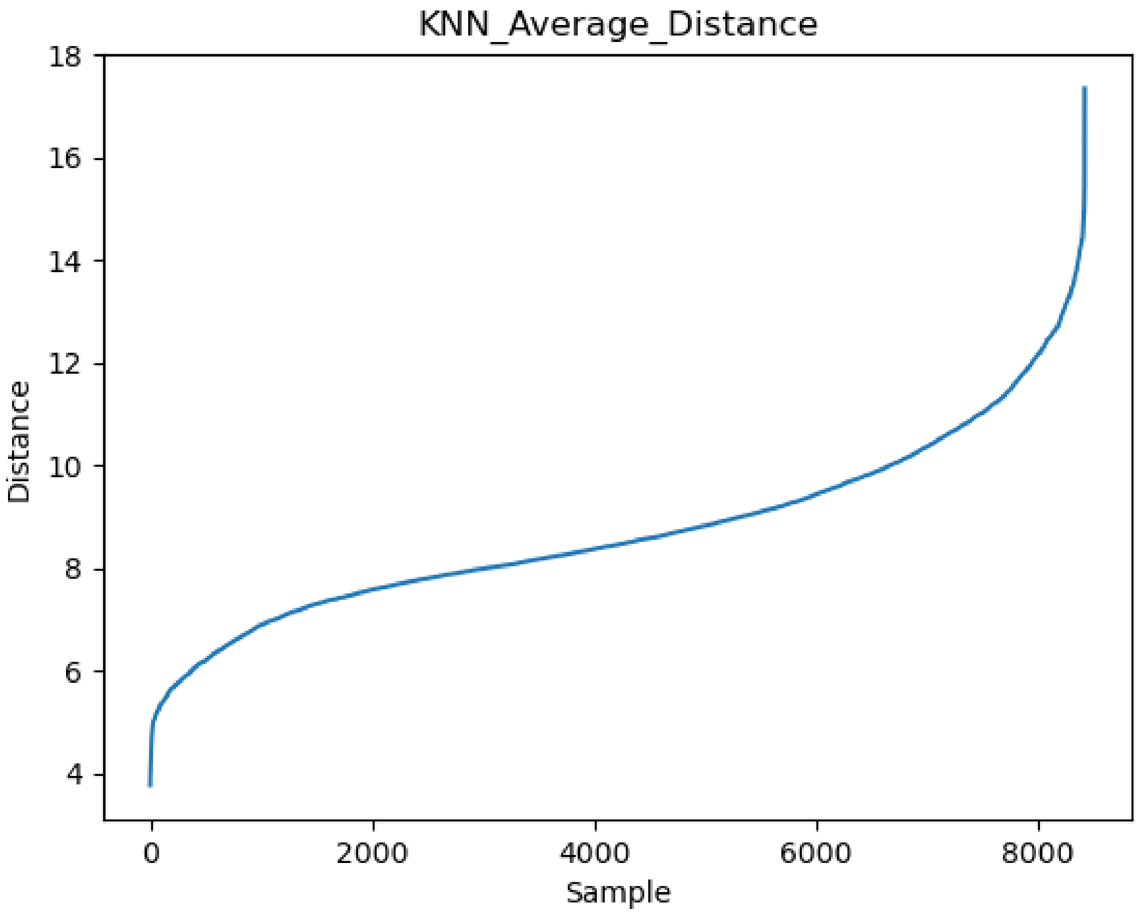


Figure 6: KNN Missed Average Distances

classifier trained in about 250x the time that the NB classifier took, while showing a much larger amount of overfitting, all for a 3% increase in accuracy. Likewise, the RF classifier had even more overfitting than the GB, and it had less performance than the GB classifier. It's saving grace is that it only took 100x longer to train than the NB classifier, but still not worth it to take that much longer to train to gain a whopping .012% better performance.

The explainability of the NB classifier is also comparable to the explainability of both of the other competitive models. KNN does beat the NB classifier in explainability, but it's the worst in terms of performance by a far enough margin to not be in the running for best model in my book.

Overall, if I had to implement one of these four models for a project, especially if that project needed quick runtime, I would choose the NB classifier.