Michael Encinas
CSE 546
Homework 3
October 3 of 2024

I will be applying four classification methods to the movie reviews classification data I was given. The four classification methods are KNN, Multinomial Naïve Bayes, Random Forest, and Gradient Boosted Regression Trees.

First, I will be applying the 4-fold cross validation to the provided training data to train my classifiers and then I will identify the optimal parameters. For the parameters I will vary the distance measure and K for KNN, for Multinomial Naïve Bayes I will be varying the alphas, for Random Forest I will be varying the n depth and n estimators and for Gradient Boosted Regression Trees I will be varying the learning rate, max depth and n_estimators.

For my training sets I will use the original data and then I will be checking afterwards when I normalize the data either with min max scaler or standard scaler if there is a change. If there is I will use that if not, then I will just keep the original data as unscaled. For some methods I do not have to normalize it as it is not applicable.

I will using cross validation on my training set with different parameters and pick the best optimized parameters for each method and then using those parameters to test them against test data set to see what their accuracy is on predicting the target output which I will explain later. I will measure the four methods against each other to pick which one is the best classification method for this dataset.

I will check for over fitting, underfitting, time to train and test as well as explainability throughout my different method outputs. I also will be inspecting feature importance at the end and some misclassified samples to see if I can analyze why some methods are different than others.

The dataset I was provided is obtained from IMDB. This dataset contains 50000 movie reviews. The data is split into 25000 reviews for training and 25000 reviews for testing. Each set will consist of 50 percent of negative reviews and 50 percent of positive reviews.

The features/variables will consist of words that were turned into sequences of integers. Each integer will stand for a word in a dictionary. In this dataset they kept the top 1000 most frequent occurring words in my training data. The target variable output for y_train and y_test will have a list of 0s and 1s. 0 stands for negative review while 1 stand for positive review.

KNN

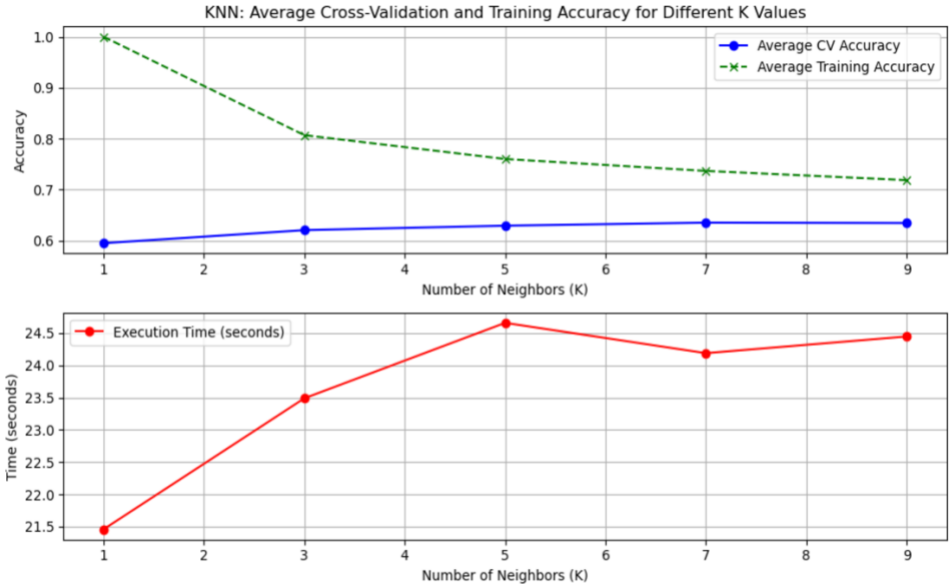
KNN is a neighbor classification method. I will first be varying my distance of K with 1,3,5,7 and 9. I will use 4 K-Fold to get the average CV accuracy and the time taken to run the dataset which will include train time and cv time together in total time.

In figure 1 and 2 you will see my printed output results of my different K parameters that has their cv accuracy score, average training accuracy and time taken. Figure 2 will visualize this.

Figure 1

```
KNN with K=1: Average CV accuracy = 0.5949, Average Training accuracy = 1.0000, Time taken = 21.46 seconds
KNN with K=3: Average CV accuracy = 0.6205, Average Training accuracy = 0.8068, Time taken = 23.49 seconds
KNN with K=5: Average CV accuracy = 0.6292, Average Training accuracy = 0.7601, Time taken = 24.66 seconds
KNN with K=7: Average CV accuracy = 0.6352, Average Training accuracy = 0.7367, Time taken = 24.19 seconds
KNN with K=9: Average CV accuracy = 0.6344, Average Training accuracy = 0.7186, Time taken = 24.45 seconds
```

Figure 2



K = 1, 3 are overfitting due to high training accuracy compared to cv scores. K = 5 and 9 are underfitting due to both low cv scores and training accuracy scores. K = 7 has an average training score of .7367 and a cv score of .6352. This is to me still underfitting but the best generalization out of all the methods. The time taken to run all KNN are lower when the neighbors are 1 or 2, but the times vary between 21 to 24 so it's not really a huge difference between the lowest K and highest K. K = 1 is the fastest as 21.5 and the rest area all between 23-25 seconds. So far based on this I would pick k = 7 based on average cv score, average training score and time taken to run.

I then will be checking Euclidean to see if for my KNN if that has an impact on my results. I will be keeping all the parameters above but now being use the method as Euclidean. I did try Manhattan as well, but the time was so high it was not worth running the program further since just the first parameter I ran took almost 400 seconds to run. Figure 3, Figure 4 and Figure 5 will show the k used, metric used, average cv score, average training score and time taken.

Figure 3

```
KNN with K=1, Metric=euclidean: Average CV accuracy = 0.5949, Average Training accuracy = 1.0000, Time taken = 24.3
8 seconds
KNN with K=3, Metric=euclidean: Average CV accuracy = 0.6205, Average Training accuracy = 0.8068, Time taken = 24.8
4 seconds
KNN with K=5, Metric=euclidean: Average CV accuracy = 0.6292, Average Training accuracy = 0.7601, Time taken = 25.4
9 seconds
KNN with K=7, Metric=euclidean: Average CV accuracy = 0.6352, Average Training accuracy = 0.7367, Time taken = 26.5
1 seconds
KNN with K=9, Metric=euclidean: Average CV accuracy = 0.6344, Average Training accuracy = 0.7186, Time taken = 25.5
5 seconds
```

For figure 3,4 and 5 nothing changes in terms of average cv/training score except the speed is slower while using Euclidean metric. So far, I will be keeping to K = 7 and keeping the metric as normal KNN.

Figure 4

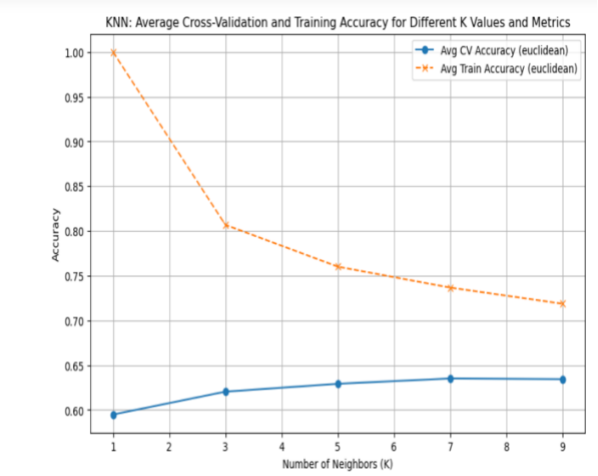
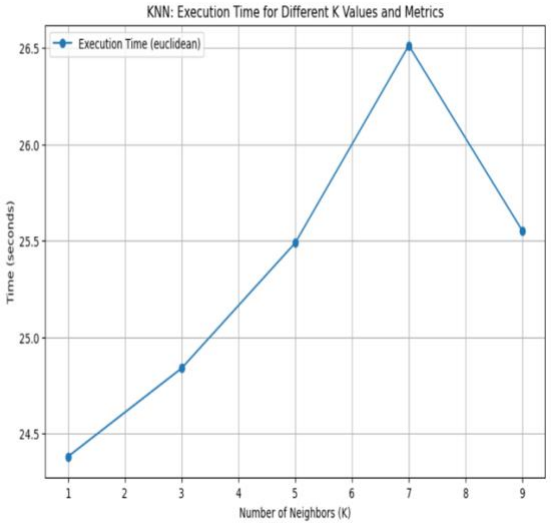


Figure 5



I will be using two normalization methods for KNN which will mix max scaler and standard scaler. In figure 6 and figure 7 I will show the results of my unscaled data, min-max scaled data, and standardized data. I will compare them to each other which will be visualized in Figure 7. The average cv/training score will be shown.

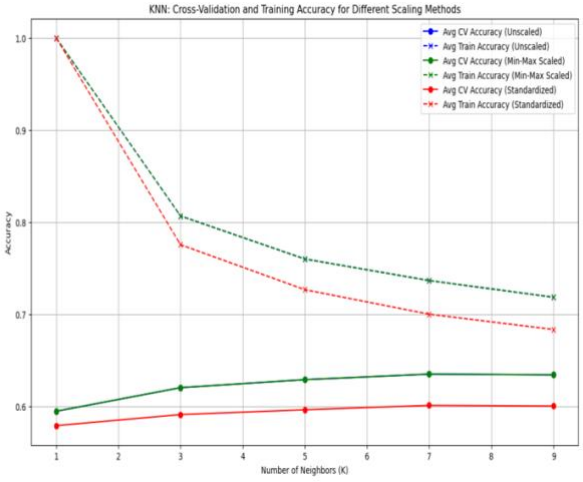
Figure 6

```
Results for unscaled data:
KNN with K=1, Metric=euclidean: Average CV accuracy = 0.5949, Average Training accuracy = 1.0000
KNN with K=3, Metric=euclidean: Average CV accuracy = 0.6205, Average Training accuracy = 0.8068
KNN with K=5, Metric=euclidean: Average CV accuracy = 0.6292, Average Training accuracy = 0.7601
KNN with K=7, Metric=euclidean: Average CV accuracy = 0.6352, Average Training accuracy = 0.7367
KNN with K=9, Metric=euclidean: Average CV accuracy = 0.6344, Average Training accuracy = 0.7186

Results for Min-Max scaled data:
KNN with K=1, Metric=euclidean: Average CV accuracy = 0.5949, Average Training accuracy = 1.0000
KNN with K=3, Metric=euclidean: Average CV accuracy = 0.6205, Average Training accuracy = 0.8068
KNN with K=5, Metric=euclidean: Average CV accuracy = 0.6292, Average Training accuracy = 0.7601
KNN with K=7, Metric=euclidean: Average CV accuracy = 0.6352, Average Training accuracy = 0.7367
KNN with K=9, Metric=euclidean: Average CV accuracy = 0.6344, Average Training accuracy = 0.7186

Results for Standardized data:
KNN with K=1, Metric=euclidean: Average CV accuracy = 0.5793, Average Training accuracy = 1.0000
KNN with K=3, Metric=euclidean: Average CV accuracy = 0.5913, Average Training accuracy = 0.7756
KNN with K=5, Metric=euclidean: Average CV accuracy = 0.5965, Average Training accuracy = 0.7268
KNN with K=7, Metric=euclidean: Average CV accuracy = 0.6013, Average Training accuracy = 0.7002
KNN with K=9, Metric=euclidean: Average CV accuracy = 0.6006, Average Training accuracy = 0.6836
```

Figure 7



In figure 6 I see that for unscaled data and min-max scaled data the scores are the same so nothing changes. However, for standardized data the scores got worst. I believe the data is normalized naturally because all the numbers are on a similar scale naturally being integers. Due to my results above, I will keep the unscaled data and keep K = 7 as my best parameter with no metric.

Multinomial Naïve Bayes

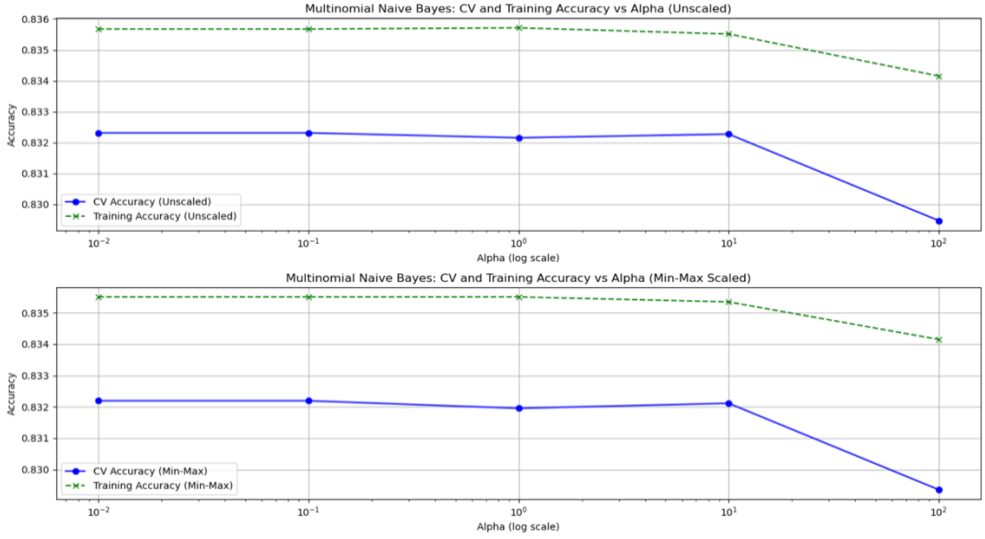
For this method I will be using 4 k-fold as well and change the alphas from .01, .1, 1, 10 and 100. I also we see if unscaled data and normalized data (min-max) changes my scores.

Figure 8 shows my unscaled vs scale results printed out and Figure 9 will visualize the results.

Figure 8

```
Evaluating Multinomial Naive Bayes with unscaled data:
Multinomial Naive Bayes with alpha=0.01: Average CV accuracy = 0.8323, Training accuracy = 0.8357, Time taken = 0.4
1 seconds
Multinomial Naive Bayes with alpha=0.1: Average CV accuracy = 0.8323, Training accuracy = 0.8357, Time taken = 0.40
seconds
Multinomial Naive Bayes with alpha=1: Average CV accuracy = 0.8322, Training accuracy = 0.8357, Time taken = 0.39 s
econds
Multinomial Naive Bayes with alpha=10: Average CV accuracy = 0.8323, Training accuracy = 0.8355, Time taken = 0.40
seconds
Multinomial Naive Bayes with alpha=100: Average CV accuracy = 0.8295, Training accuracy = 0.8342, Time taken = 0.43
seconds
Evaluating Multinomial Naive Bayes with Min-Max scaled data:
Multinomial Naive Bayes with alpha=0.01: Average CV accuracy = 0.8322, Training accuracy = 0.8355, Time taken = 0.4
2 seconds
Multinomial Naive Bayes with alpha=0.1: Average CV accuracy = 0.8322, Training accuracy = 0.8355, Time taken = 0.59
seconds
Multinomial Naive Bayes with alpha=1: Average CV accuracy = 0.8320, Training accuracy = 0.8355, Time taken = 0.43 s
econds
Multinomial Naive Bayes with alpha=10: Average CV accuracy = 0.8321, Training accuracy = 0.8354, Time taken = 0.42
seconds
Multinomial Naive Bayes with alpha=100: Average CV accuracy = 0.8294, Training accuracy = 0.8342, Time taken = 0.44
seconds
```

Figure 9



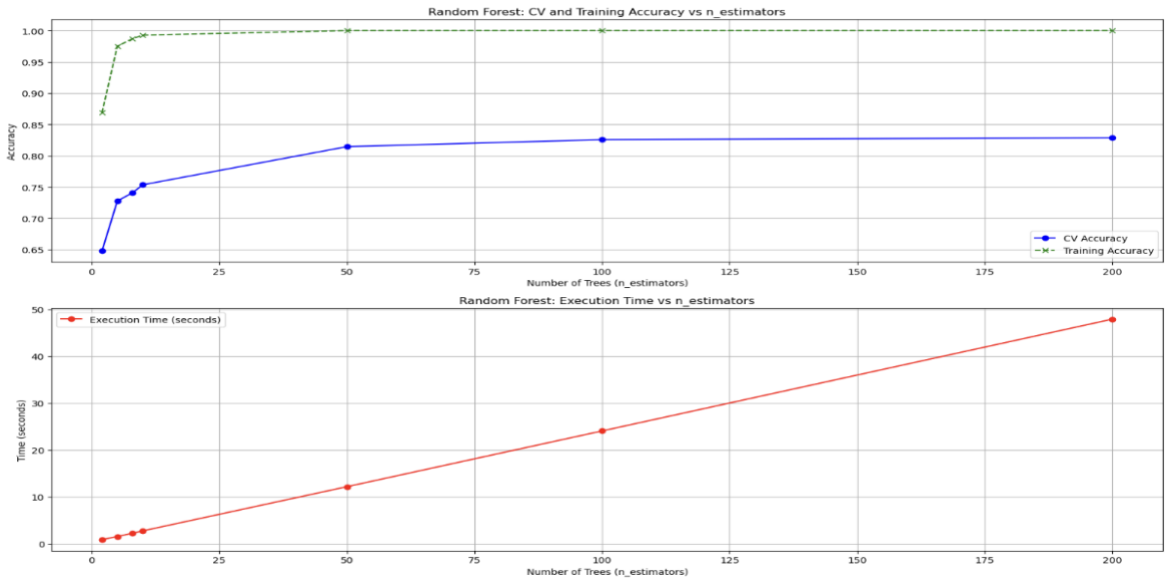
Multinomial Naïve Bayes (MNB) is generalizing well on all alphas and when it comes to time taken the unscaled in a little faster than the min-max scaled. The difference is minor. For this parameter model chose I will be picking the unscaled normalization and be using alpha = 1 just based on time being the fastest.

Random Forest

I will first be doing n_estimators as different values 2, 5, 8, 10, 50, 100, 200. Then I will do max depth at different values 2, 5, 7, 10, 20, 40. I will chose the top 3 of each and combine them to see what my scores are for those. I will still be looking a time, cv and training scores.

In Figure 10 I have cv and training average scores of different N_estimators displayed as well as their execution times.

Figure 10



In figure 10, N_estimators approve when trees are increased. At 10, average training score and cv score are at .9927 with cv score of .7547 with time taken at 2.73. However, I believe 10 is over fitted because its training score is so much higher than its cv score. The rest of the data n_estimator that go past 10 which is 50, 100, and 200 are all overfitting. Underfitting is occurring at 2. I would pick 5 as N_estimator here since even if score is high on training compared to cv score it's not as extreme compared to the others.

For max depth my results are shown below for my different parameters. For Max depth, my accuracy scores are alot better than when compared to n_estimators. Time taken is a lot slower after max depth is past 10. 8 and below have very good times with the lower getting faster. Max depth of 20 and 40 is over fitting. Reference Figure 11 to get my output. Based on time taken and cv/training accuracy score I will select Max depth 5 since the average CV accuracy score of .7910 with average training score of .8038 with 3.98 seconds of taken time is a good generalization.

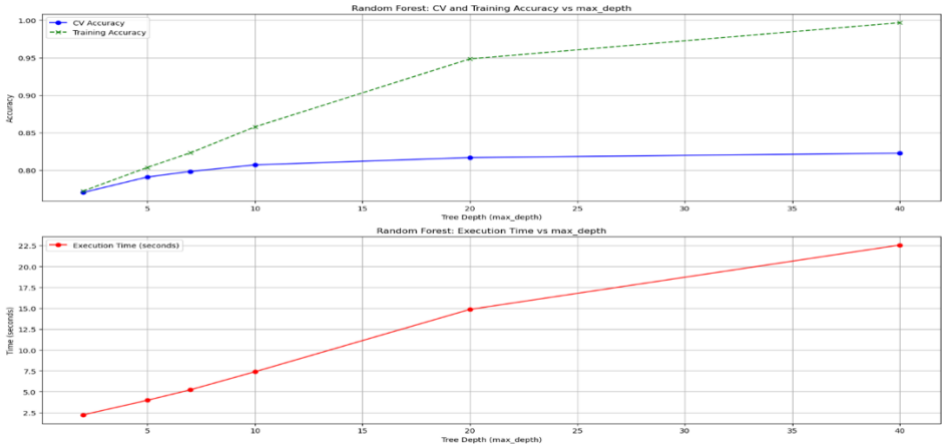


Figure 11

When making my max depth not more than 7 and seeing if my cv/training accuracy score increase with my trees going up to 200. I noticed that with the trees combined with max depth have low scores with n_estimators low but do approve when max_depth is increased. The time is fast when N is lower and max depth is lower but my scores are really underfitting until I get to N estimator equals 100 and above. At N_estimator equals 100 with Max depth equals to 5 I have a average cv accuracy score of .7910 with average training accuracy score of .80. Time taken for these parameters are 3.89. Since this is just about equals with my Max depth of 5 alone parameter, I will take the n_estimator 100 max depth parameter 5 as my chose testing parameter.

Figure 12

```
Random Forest with n_estimators=2 and max_depth=2: Average CV accuracy = 0.6319, Average Training accuracy = 0.633
3, Time taken = 0.49 seconds
Random Forest with n_estimators=2 and max_depth=5: Average CV accuracy = 0.6745, Average Training accuracy = 0.680
5, Time taken = 0.40 seconds
Random Forest with n_estimators=2 and max_depth=7: Average CV accuracy = 0.6988, Average Training accuracy = 0.709
7, Time taken = 0.42 seconds
Random Forest with n_estimators=5 and max_depth=2: Average CV accuracy = 0.6647, Average Training accuracy = 0.664
5, Time taken = 0.42 seconds
Random Forest with n_estimators=5 and max_depth=5: Average CV accuracy = 0.7146, Average Training accuracy = 0.725
3, Time taken = 0.50 seconds
Random Forest with n_estimators=5 and max_depth=7: Average CV accuracy = 0.7359, Average Training accuracy = 0.754
9, Time taken = 0.57 seconds
Random Forest with n_estimators=8 and max_depth=2: Average CV accuracy = 0.6907, Average Training accuracy = 0.690
6, Time taken = 0.47 seconds
Random Forest with n_estimators=8 and max_depth=5: Average CV accuracy = 0.7275, Average Training accuracy = 0.737
0, Time taken = 0.61 seconds
Random Forest with n_estimators=8 and max_depth=7: Average CV accuracy = 0.7491, Average Training accuracy = 0.768
4, Time taken = 0.71 seconds
Random Forest with n_estimators=50 and max_depth=2: Average CV accuracy = 0.7529, Average Training accuracy = 0.754
4, Time taken = 1.23 seconds
Random Forest with n_estimators=50 and max_depth=5: Average CV accuracy = 0.7838, Average Training accuracy = 0.794
2, Time taken = 2.09 seconds
Random Forest with n_estimators=50 and max_depth=7: Average CV accuracy = 0.7927, Average Training accuracy = 0.815
6, Time taken = 2.74 seconds
Random Forest with n_estimators=100 and max_depth=2: Average CV accuracy = 0.7704, Average Training accuracy = 0.77
22, Time taken = 2.15 seconds
Random Forest with n_estimators=100 and max_depth=5: Average CV accuracy = 0.7910, Average Training accuracy = 0.80
38, Time taken = 3.89 seconds
Random Forest with n_estimators=100 and max_depth=7: Average CV accuracy = 0.7984, Average Training accuracy = 0.82
32, Time taken = 5.15 seconds
Random Forest with n_estimators=200 and max_depth=2: Average CV accuracy = 0.7760, Average Training accuracy = 0.78
01, Time taken = 3.95 seconds
Random Forest with n_estimators=200 and max_depth=5: Average CV accuracy = 0.7954, Average Training accuracy = 0.80
81, Time taken = 7.49 seconds
Random Forest with n_estimators=200 and max_depth=7: Average CV accuracy = 0.8011, Average Training accuracy = 0.82
68, Time taken = 10.13 seconds
```

I also took 100 and 200 for n estimator and used the max depth of 2,5 and 7 with max features of 30, 40 and 50 and my results did not approve with a time that was a little faster than my original results above in figure 12. Figure 13 shows a small sample to understand what I mean.

Figure 13

```
Random Forest with n_estimators=100, max_depth=2, max_features=40: Average CV accuracy = 0.7588, Average Training a
ccuracy = 0.7615, Time taken = 2.40 seconds
Random Forest with n_estimators=100, max_depth=2, max_features=50: Average CV accuracy = 0.7598, Average Training a
ccuracy = 0.7618, Time taken = 2.69 seconds
Random Forest with n_estimators=100, max_depth=5, max_features=30: Average CV accuracy = 0.7899, Average Training a
ccuracy = 0.8018, Time taken = 3.78 seconds
Random Forest with n_estimators=100, max_depth=5, max_features=40: Average CV accuracy = 0.7818, Average Training a
ccuracy = 0.7928, Time taken = 4.59 seconds
Random Forest with n_estimators=100, max_depth=5, max_features=50: Average CV accuracy = 0.7756, Average Training a
ccuracy = 0.7864, Time taken = 5.33 seconds
Random Forest with n_estimators=100, max_depth=7, max_features=30: Average CV accuracy = 0.7958, Average Training a
ccuracy = 0.8210, Time taken = 5.13 seconds
Random Forest with n_estimators=100, max_depth=7, max_features=40: Average CV accuracy = 0.7874, Average Training a
ccuracy = 0.8122, Time taken = 6.24 seconds
```

Gradient Boosted Regression Trees

For gradient boosted regression trees I used n estimators as 50, 100 or 200 with learning rates at .01, .1 and .2 as well as max depth of 2, 3 and 5. In figure 14 you get the training score, cv score, and execution times.

Figure 14

Based on the results in figure 14. Underfitting occurs when N = 50 when learning rate is low at .01. However, at .1 and .2 the model parameters are generalizing well. It appears as with 100 and 200 as you go up in learning rate the models perform better however at max depth 5 for learning rate of .2, they seem to be overfitting with high training accuracy and low cv accuracy. Time taken is increased with N = 100 and 200. Even with time being high for 100 and 200, the best parameter that I can find was with n_estimators=200, learning_rate=0.2, max_depth=2: CV accuracy = 0.8391, Training accuracy = 0.8598, Time = 88.27. The time is long but n = 50 only had higher accuracy scores were set at .2. My time is about 40 seconds slower compared to the

lower values but at out of 4 models this was the best generalization that I got and I feel its worth it to pick this specific parameter on this model.

Test Run

Parameters for each model

KNN has k = 7 and unscaled
Multinomial Naïve Bayes has Alpha = 1
Random Forest has N_estimator = 100 and max depth = 5
Gradient Boosted Regression Tree has N_Estimator = 200, learning rate = .2 and max depth = 2

Before I could run the code, I had an issue with columns in Train and Test datasets that were not in the other dataset. Out of 1000 columns in each dataset. I had 55 in train not in test and 55 in test not in train. So, I deleted those columns so I could not get any issues with my code. I did not want to put any substitute number in there to cause some bias and I also know that I can cause a bias by deleting them because It can cause something.

When running my best parameters on my new test data that consist of 25000 new datapoints. I had an output that is visualized in Figure 15 and Figure 16.

Figure 15

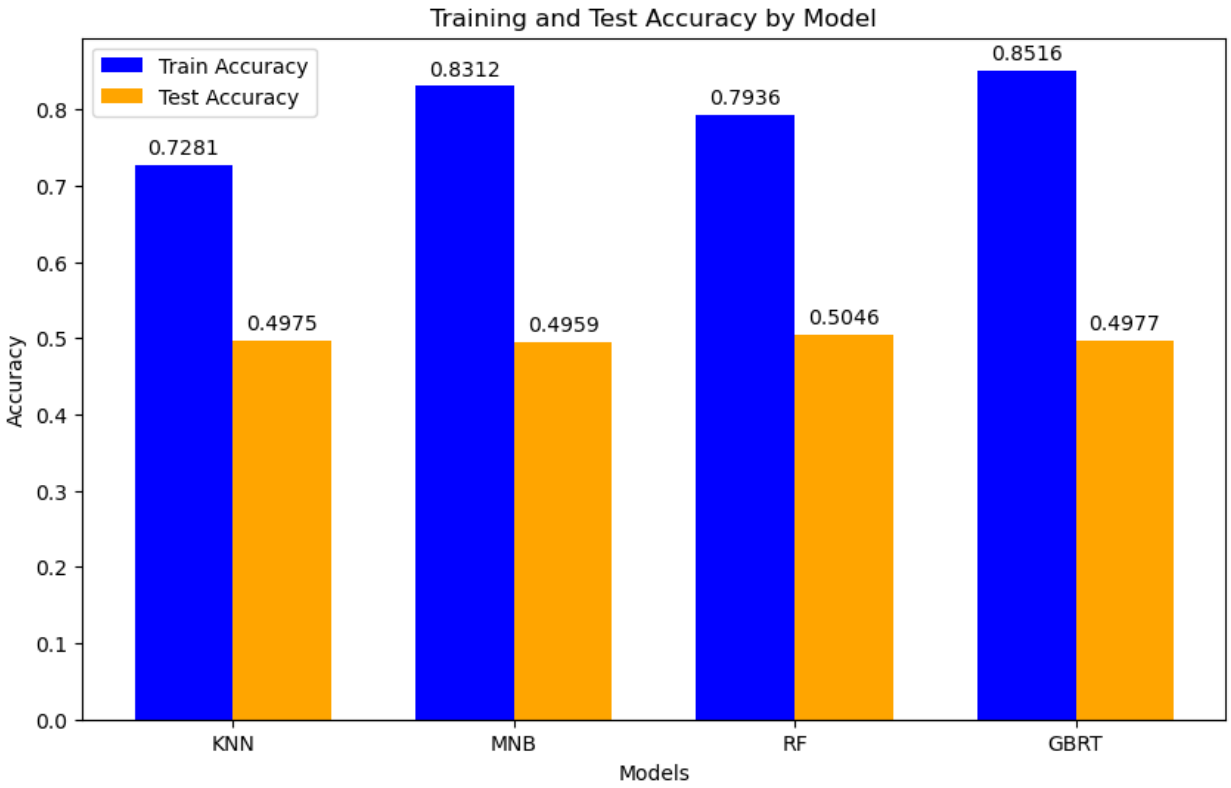
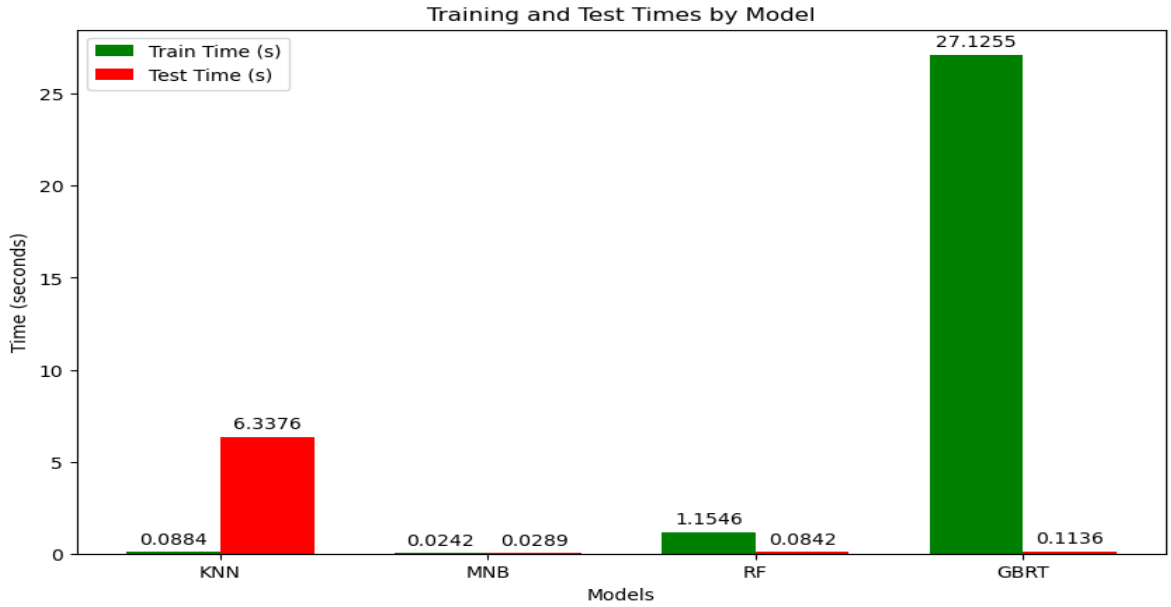


Figure 16



When looking at Figure 15, I have overfitting on all my models. I found this to be odd since I had good generation on my training data but then when I get a new dataset my results were not good. When looking at Figure 16 my times were showing that MNB and RF are the fastest ran models compared to KNN and GBRT. With GBRT taking the most amount of time which I expected since it was the slowest model on my training dataset.

I wondered if I deleted important columns when I tried to keep shared column names between each dataset. I looked at my random forest model and gradient boosted trees on just my train data with those best parameters and I get the top ten features which is shown in figure 17 and figure 18. I then print out the top ten features on my test dataset for random forest model and gradient boosted trees to see if there is a different in important features on those different datasets.

Figure 17

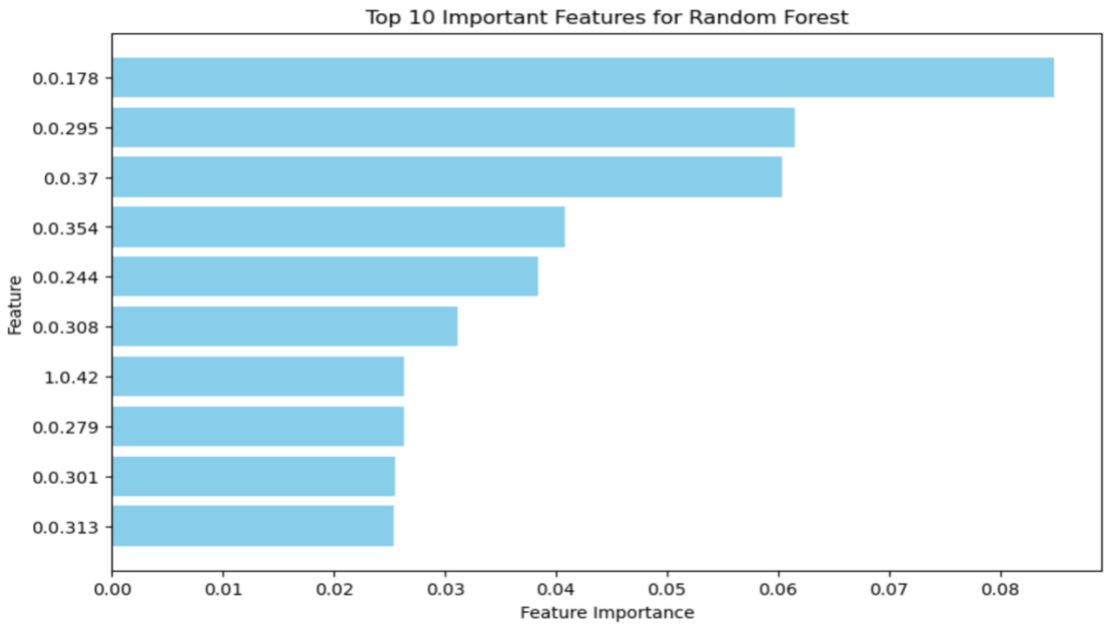
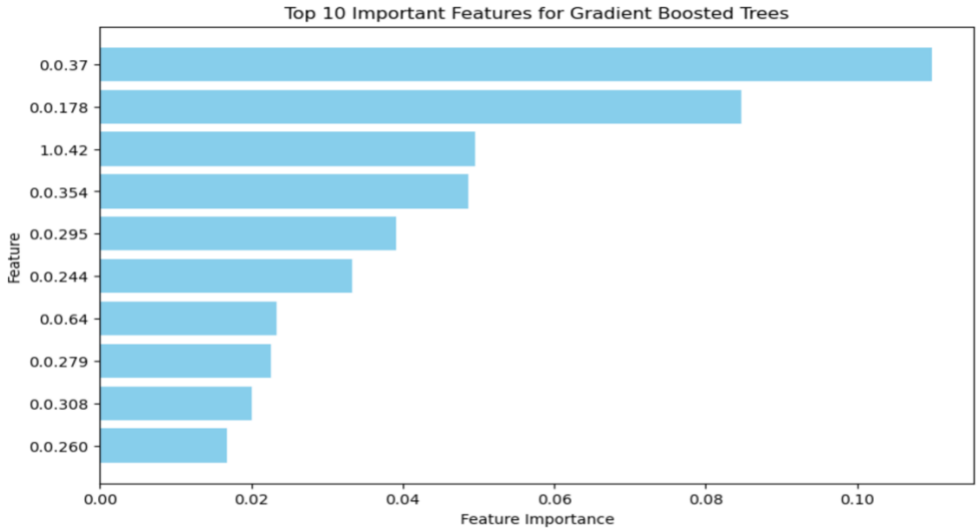


Figure 18



I have the top ten features from these models above in my train dataset which shows that both models do share at least 50 percent of important features.

In figure 19 I printed out the top ten features which is visualized below.

Figure 19

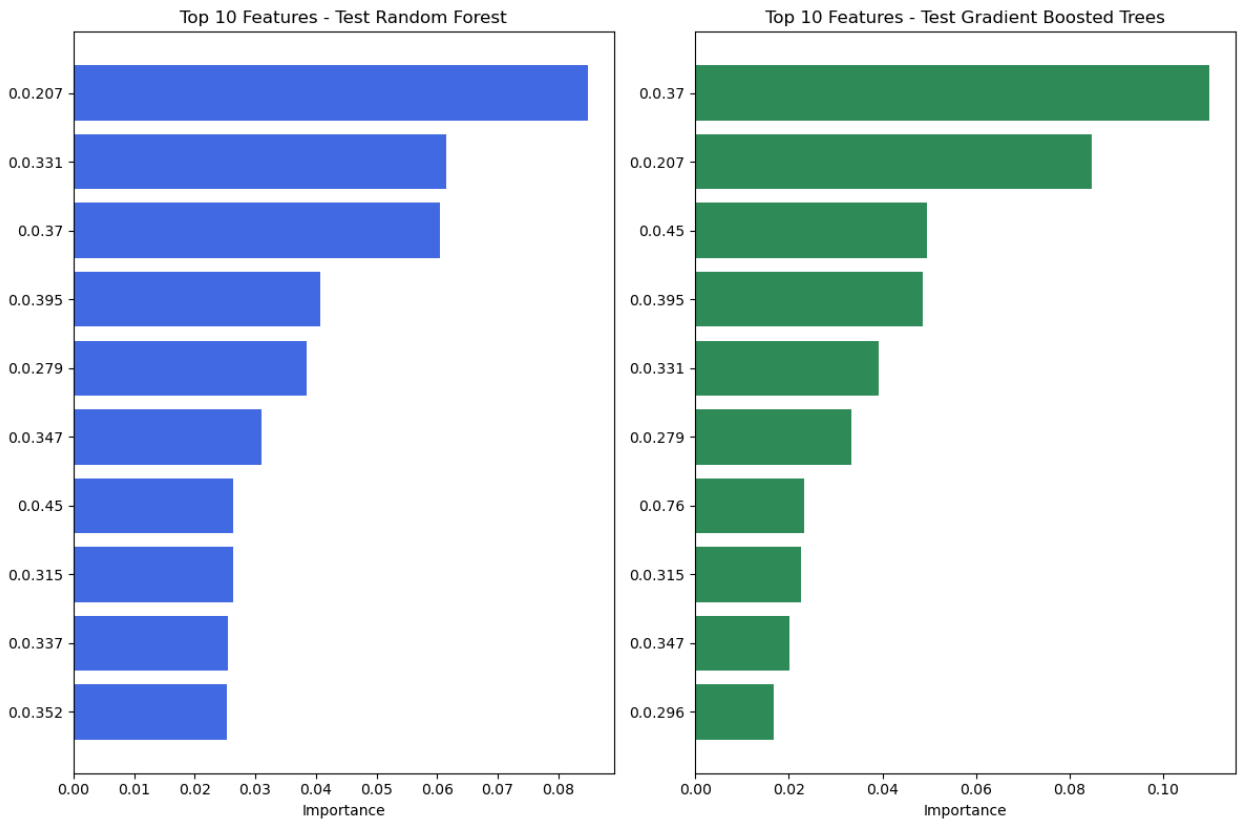


Figure 19 shows that after my model is trained on the train dataset and then ran on the new dataset with the x-test aligned columns then above is the new important features. My results are a lot different on feature importance compared to my train models top features.

I believe that my models are not performing well on the new test dataset. It might be that the new 25 k samples might be very diverse and different than the train dataset so that's why my generalization scores are so low, and my feature importance has changed so dramatically.

I could just train my model on the test datasets but that's bias and my model is supposed to train with the train dataset and hopefully be able to deal with different datasets. Just to show you what the results are if I was to train my test model on my test datasets my output for most important features is a lot different.

Conclusion

Based on the parameters that I chose during cross-validation for my train data, all my models (KNN, MNB, RF and GBRT) performed well on their generalization scores for my training data. However, for my testing dataset my models all performed bad with overfitting occurring.

What this tells me is that the new model does not perform well with the new test dataset. Based on my important features, it appears the important features are different in test dataset compared to train datasets.

Since I cannot train my model with test dataset than maybe n_samples column might perform better for my models. I was surprised by my results, and I thought maybe even have less features might have prevented this overfitting from occurring if my parameters were set up prior to testing on my test dataset.

Link: https://youtu.be/5F0hEFFA340