Homework 3 Report

CSE590-12 Machine Learning

Jonathan Loyd

The following experiments were conducted to determine the usefulness of Multinomial Naïve Bayes, Random Forest, and Gradient Boosted Regression Trees on a given movie ratings dataset. The data used for these experiments were given in 4 different csv files, which were already split into training and testing data for both the y and x axis. The data was split into training and validation data using k-fold cross-validation, where k is equal to 4. After this, the training and validation data is used to find the optimal parameters of the 3 previously mentioned learning models. Once the optimal parameters are found and chosen, those parameters are used to predict the testing data.

The first model validated was Multinomial Naïve Bayes, which predicts the class based on the frequency of something, which in this case is words, appearing in the test samples based on the validation or training samples. This method only has one parameter that will be focused on, which is alpha, or the value that adjusts additive smoothing. A range of $1x10^{-5}$ to $1x10^{5}$ was tested, and the model starts overfitting as alpha increases past a value of 1, but there is close to no change in performance on values around alpha equal to .01 to 1, where performance is best.

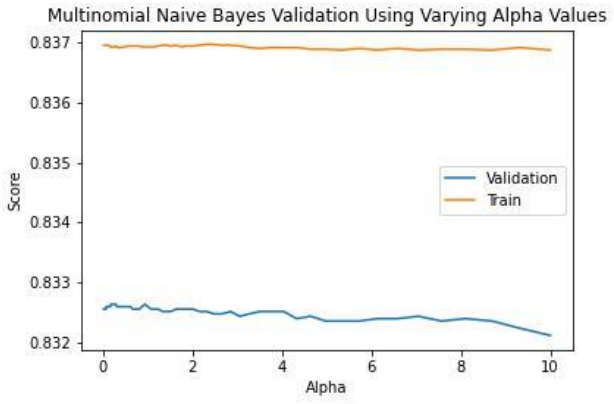

Figure 1: Multinomial Naive Bayes Validation Using Varying Alpha Values

Figure 1 shows the decline of alpha as it increases past 1 and shows the stagnation at lower values. Since the values are so close, an alpha value of 1 was used as the optimal parameter for testing. The training score using an alpha value of 1 was 0.8369 and the validation score was 0.8326. On the testing data, the training score was 0.8357 and the testing score was .8323. The accuracy is similar, so the data used in validation appears to have an accurate representation of the rest of the data used in testing.

Using the feature log probability, the 10 most important features for identifying negative samples are features 1, 2, 4, 6, 5, 7, 8, 14, 9, 11, 12, 15, 13, 21, 18, 19, 16, 20, 23, and 24. The most important features for identifying positive samples are 1, 2, 4, 5, 6, 7, 8, 9, 11, 14, 12, 15, 13, 18, 19, 21, 17, 23, 16, and 10.

The misclassified samples can be analyzed using the most important features. One such sample that was misclassified was sample 3, which was predicted to be class 0, but was true class 1. This sample has all zero values for the negative feature identifiers, which is most common in features that have been classified as negative, which could be why it was misclassified. Another misclassified sample was sample 296, which was predicted to be class 1, but was true class 0. This sample has one values for features 8, 15, 10, and it is more common for samples classified as class 1 to have one values for features 8 and 10 than samples that were classified as class 0. This may explain this sample's misclassification.

The Random Forest model was tested after Multinomial Naïve Bayes. The Random Forest model is an ensemble method that uses multiple different decision trees using a random element to predict on a dataset. The important parameters for Random Forests are n_estimators, max_features, max_depth, and max_leaf_nodes. Due to time constraints, n_estimators was not tested, although it should be noted that in general increasing n_estimators will improve the accuracy of the model, but increase the amount of memory and time required to train the model. max_leaf_nodes was the first parameter tested.

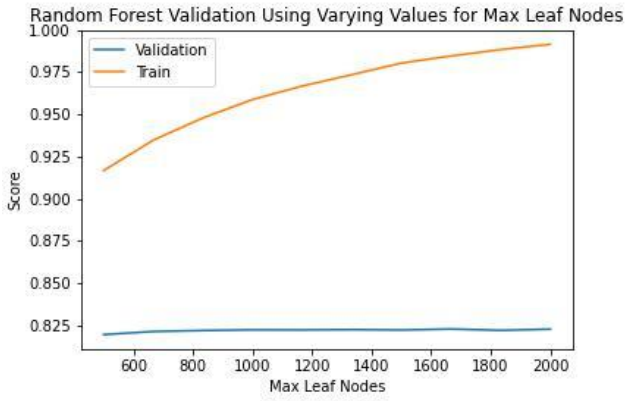

Figure 2: Random Forest Validation Using Varying Values for Max Leaf Nodes

There is minimal change in accuracy for max_leaf_nodes as long as the maximum number of leaf nodes is 500 or above. There is underfitting below 500 leaf nodes, but the difference in performance for more than 500 leaf nodes is around a 0.01 change in accuracy. Of the values tested, 1666 leaf nodes performed slightly better than the others with a value of 0.8229 compared to a low of 0.8196.

max_features was also tested. The values remain steady, only fluctuating from 0.8226 to 0.8254 after there are 10 or more max features.

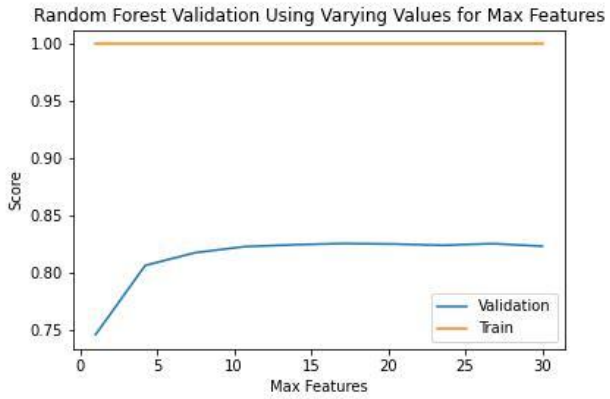

Figure 3: Random Forest Validation Using Varying Values for Max Features

Figure 3 shows the lack of fluctuation in max_features values greater than 10. It also shows that the model performs worse with max_features less than 10. There appears to be underfitting as max_features is closer and equal to 1. The max_features performed best around 15 features, but this parameter was further tested in combination with the next parameter, max_depth.

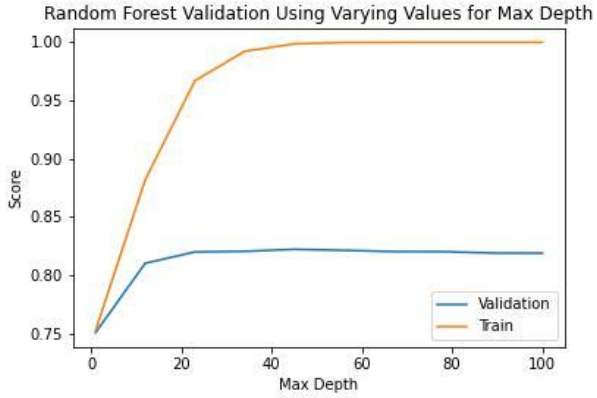

Figure 4: Random Forest Validation Using Varying Values for Max Depth

The max_depth parameter told a similar story as max_features as seen in Figure 4, but the accuracy with varying max_depth values fluctuated more. There is underfitting at lower max_depth values, especially towards a value of 1. The best performance was around a max_depth of 45, but this parameter was further tested in combination with max_features.

Table 1: Further Testing on Random Forest Validation Scores with Max Depth and Max Features

	Features 13	Features 14	Features 15	Features 16	Features 17
Depth 43	0.8298	0.8297	0.8337	0.8287	0.8278
Depth 44	0.8297	0.8282	0.8330	0.8316	0.8268
Depth 45	0.8300	0.8286	0.8323	0.8312	0.8288
Depth 46	0.8294	0.8302	0.8344	0.8296	0.8292
Depth 47	0.8304	0.8297	0.8329	0.8289	0.8287

Further testing was conducted on max_features and max_depth with max_leaf_nodes equal to the default ('None') and with max_leaf_nodes equal to 1666. The performance was much better with max_leaf_nodes equal to 1666, which is the contents of Table 1. In Table 1, the maximum value is highlighted in green, and the minimum value in red. These results show that the optimal parameters for the Random Forest model are max_leaf_nodes equal to 1666, max_features equal to 15, and max_depth equal to 46. Using the optimal parameters on the validation data, the training score for the validation data was 0.9825 using these parameters, and the validation score was 0.8344. Using the optimal parameters again on the testing data, the training score was 0.9701 and the testing score was 0.8318. While the model performed 0.01 worse on the training score for the testing data, the model performed almost exactly as well for the testing scores, which is the goal.

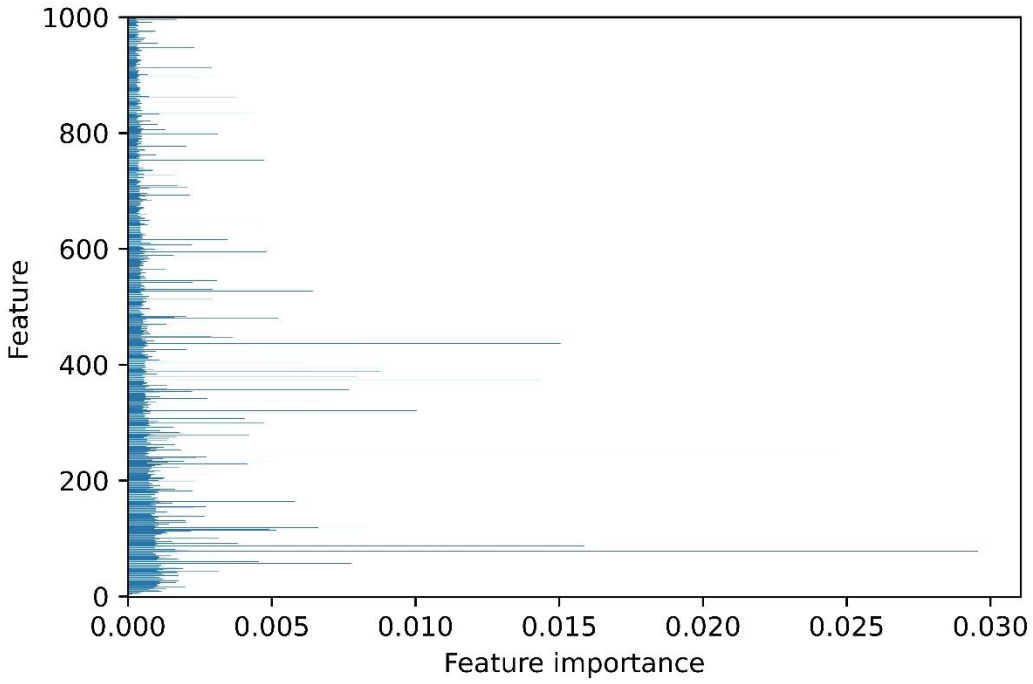

Figure 5: Random Forest Classifier Feature Importance

When testing for feature importance on the Random Forest model with optimal parameters, the most important is features were 78, 249, 87, 437, 373, 321, 394, 389, 118, and

433. Most of the features were nonzero, and therefor had some importance. There were 3 values at 0, which were features 1, 2, and 4, which means these features were not used in classifying samples for this model.

The most important features can be used to analyze misclassified samples. One misclassified sample was sample 7, which was true class 0 but was predicted to be class 1. Sample 7 had values of 0 for features 78, 249, 437 and 373, which was much more common in samples strongly predicted to be class 1, which could explain why it was misclassified as class 1. Another misclassified sample was sample 145, which was classified as class 0, but was true class 1. This might be because it had values of 0 for features 87, 321, and 389, which were more common for samples strongly classified as class 0.

The last model used was Gradient Boosted Regression Trees, which is another ensemble method that is like the Random Forest method, but builds trees in a serial manner, where each tree iteratively improves the performance of the model. This method requires strong pre-pruning due to the amount of memory and time it takes. The main parameters are n_estimators, max_depth, max_leaf_nodes, min_samples_leaf, and learning_rate. Due to time and memory constraints, n_estimators tested minimally, however adding more trees will generally increase the performance of the model because it will add more trees, which has more chances to correct the mistakes of the model on the training set. The number of estimators, n_estimators was left at its default value of 100 to further test other parameters on the dataset because it performed marginally better than lower values, such as 20 or 50, but was not marginally worse than increasing values such as 150. Using 100 estimators also allows the model to run in a reasonable amount of time to allow further testing on other parameters

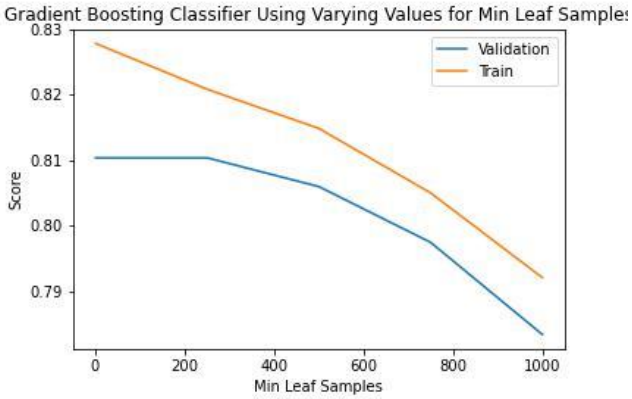

Figure 6: Gradient Boosting Regression Tree Classifier Using Varying Values for Min Leaf Samples

The min_samples_leaf parameter adjusts the minimum number of samples per leaf for each tree. Figure 6 shows that the performance decreases, and there is underfitting as min_samples_leaf increases to larger values, such as 1000. The performance is best for training and validation with a min_samples_leaf value of 1, so it will be used as an optimal parameter for the Gradient Boosted Regression Trees model.

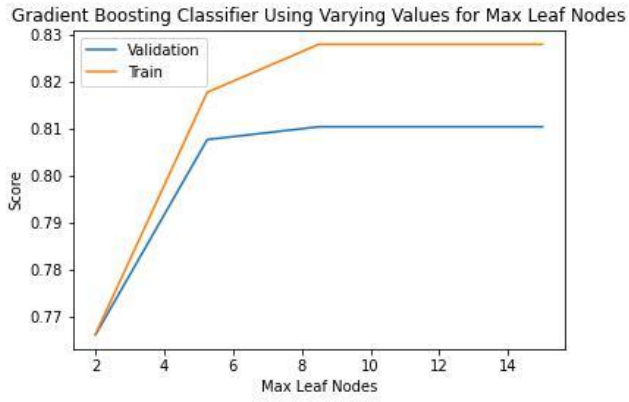

Figure 7: Gradient Boosted Regression Trees Classifier Using Varying Values for Max Leaf Nodes

The max_leaf_nodes parameter adjusts the maximum number of leaf nodes for each tree. There is underfitting at small values, such as max_leaf_nodes equal to 2, as shown in Figure 7. The figure also shows that at around 8 leaf nodes the performance flatlines and does not fluctuate at all. With a default value of max_depth for testing equal to 3, max_leaf_nodes has the best performance at higher values, so the default value of 'None', which refers to no limit on the number of leaf nodes, will be used as the optimal parameter for the Gradient Boosted Regression Trees model.

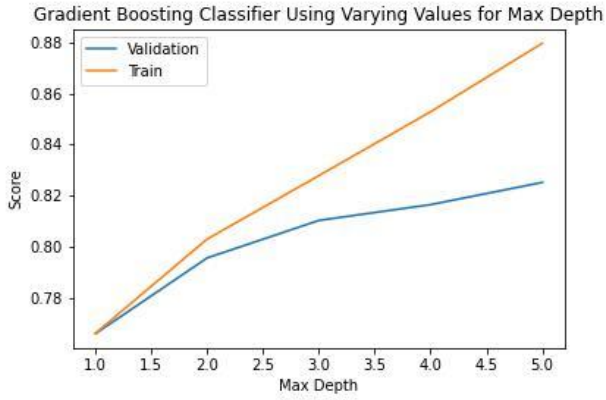

Figure 8: Gradient Boosted Regression Tree Classifier Using Varying Values for Max Depth

The max_depth parameter changes the depth that each tree can be before it is pruned in the model. The range tested was from 1 to 5. Figure 8 shows that there is underfitting in the case of max_depth equal to 1. It also shows that in the range, a value of 5 performed the best. It takes a considerable amount of time to run the model with max_depth equal to a value of 5 compared to a value of 1, or even a value of 3. Although higher values, such as 6, could produce more accurate results than a value of 5, it would consume too many resources to test on the current hardware. Additionally, a range of 1 to 5 is a common range for max_depth, but the value of 5 being the most accurate of those tested could be in part due to the number of estimators, n_estimators, being relatively low for Gradient Boosted Regression Trees. Since the number of

estimators are low, having a larger depth could help increase the complexity, whereas if there were more estimators, a smaller depth could be more optimal.

The learning_rate parameter changes the amount each tree contributes to the classification. The sklearn documentation notes that there is a trade-off between n_estimators and the learning_rate. With a higher value of n_estimators, a lower value for learning_rate may be desirable, but since n_estimators will remain at 100 for testing on this dataset due to time and memory constraints, the optimal learning_rate was found specifically for 100 trees on the dataset. The learning_rate will also depend on the max_depth. The learning rate was tested on a max_depth value of 5, since that was discussed previously to be the best parameter for the resources available.

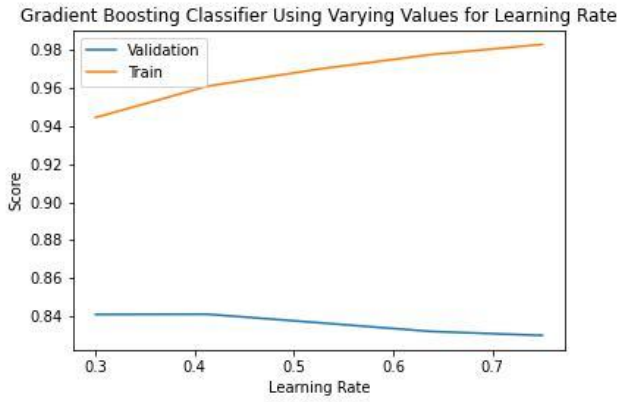

Figure 9: Gradient Boosted Regression Tree Classifier Using Varying Values for Learning Rate

Figure 9 shows some amount of overfitting at larger values for learning_rate such as 0.75, and the model fits best in the range of 0.3 to 0.525, with the best validation score for a learning_rate value of 0.4125. This value is a little out of the norm, where many models have the optimum between 0.1 and 0.3, but this could be due to the testing value for n_estimators being 100, which is relatively small for a Gradient Boosted Regression Trees model.

The parameters found to be optimal given time and memory constraints for the Gradient Boosted Regression Trees model on this dataset are to have min_samples_leaf equal to 1, max_leaf_nodes equal to 'None', max_depth equal to 5, and the learning_rate equal to 0.4125. When fit on the validation data, the model has a training score of 0.9611 and a validation score of 0.8409. The model performs similarly on the testing data, with a training score of 0.9611, and a testing score of .8406. The training score and the testing scores are almost the same, which indicates that this model is performing similarly between the two, and it is unlikely that the model is being overfit to adjust to the training data.

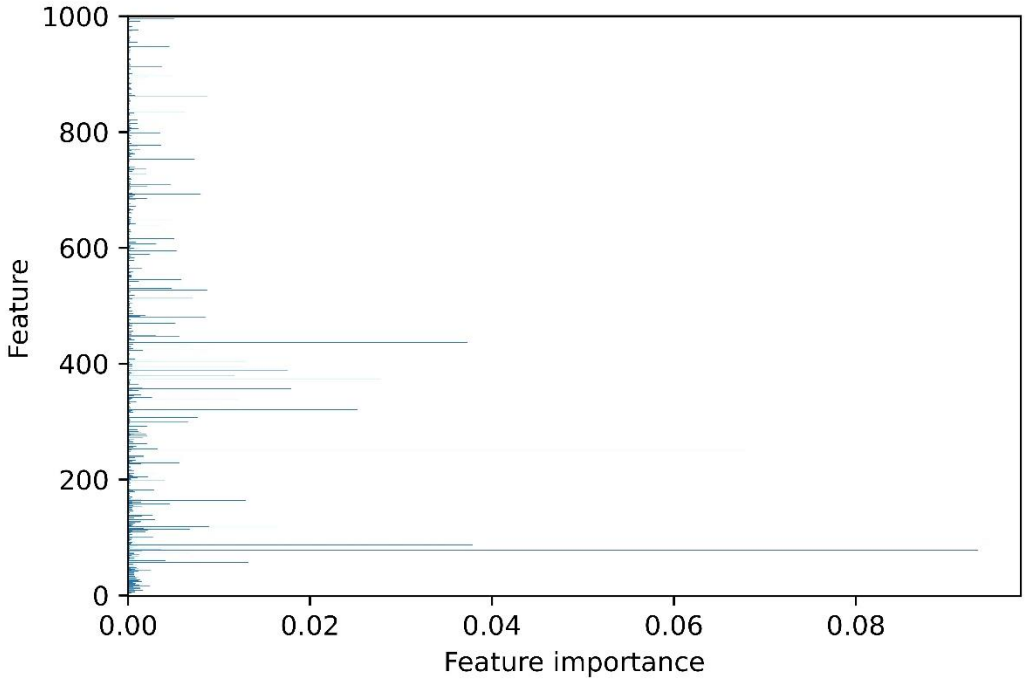

Figure 10: Gradient Boosted Regression Trees Feature Importance

Figure 10 shows the feature importance for the Gradient Boosted Regression Trees model with optimal parameters. The top 10 most relevant features are 78, 249, 87, 437, 373, 321, 357, 389, 118, and 57. These features can be used to analyze misclassified samples. One example for a misclassified sample is sample 2, which was predicted to be class 1, but was true class 0. One reason that it could be misclassified is that it had a value of 0 for feature 78, the most important feature, which was more common for samples classified as class 1. Another misclassified sample was sample 24965, which was classified as 0, but was true class 1. This may be due to feature 57 being 1 for the sample, which was much more common in samples predicted to be class 0.

The test score performance was similar between the 3 models on the dataset, but the Gradient Boosted Regression Tree model performed slightly better. In terms of explain ability, the Random Forest model is the easiest to explain and visualize, with Gradient Boosted Regression Trees being almost as easy to explain and visualize. Random Forest takes marginally less time to train and test than Gradient Boosted Regression Trees, but the Multinomial Naïve Bayes is even faster than both models. These results make Gradient Boosted Regression Trees seem like the best model on the given dataset, because it can be trained and tested much quicker than Gradient Boosted Regression trees, it has similar performance on testing data, and it is the easiest of the models to explain to someone with a nontechnical background.