CS6375 - Project 2 – Report
Shubham Shekhar Jha (sxj220028)

Source Code: I have 2 versions of the source code for this project.
1. Python scripts in the src zip. The projects can be run using the "run.sh" file. [Might take an exceedingly long time to run, thus previous run outputs are present in output.txt and mnist_output.txt]
2. Python notebook on Colab [https://colab.research.google.com/drive/1akgjFx9MLi4eqsTtVyoNoS4i-zLPTZHP]. The .ipynb file downloaded from Colab is also present in the src zip. [The outputs of previous can also be seen here]

Part 5. Accuracy and F1 score for each dataset and classifier

Dataset	DecisionTree	Bagging	Random Forest	Gradient Boosting
c300_d100	accuracy: 0.655	accuracy: 0.675	accuracy: 0.84	accuracy: 0.835
	F1 score: 0.68778	F1 score: 0.67337	F1 score: 0.84466	F1 score: 0.83744
c300_d1000	accuracy: 0.668	accuracy: 0.8405	accuracy: 0.8765	accuracy: 0.991
	F1 score: 0.66191	F1 score: 0.83782	F1 score: 0.87802	F1 score: 0.99108
c300_d5000	accuracy: 0.7588	accuracy: 0.9081	accuracy: 0.9087	accuracy: 0.994
	F1 score: 0.75577	F1 score: 0.91096	F1 score: 0.91059	F1 score: 0.9940
c500_d100	accuracy: 0.65	accuracy: 0.82	accuracy: 0.87	accuracy: 0.9
	F1 score: 0.66667	F1 score: 0.82	F1 score: 0.87	F1 score: 0.90196
c500_d1000	accuracy: 0.685	accuracy: 0.8725	accuracy: 0.947	accuracy: 0.996
	F1 score: 0.67923	F1 score: 0.87179	F1 score: 0.94742	F1 score: 0.99601
c500_d5000	accuracy: 0.7828	accuracy: 0.9216	accuracy: 0.9559	accuracy: 0.9969
	F1 score: 0.78482	F1 score: 0.92169	F1 score: 0.95632	F1 score: 0.9969
c1000_d100	accuracy: 0.7	accuracy: 0.875	accuracy: 0.965	accuracy: 0.985
	F1 score: 0.70874	F1 score: 0.87562	F1 score: 0.96482	F1 score: 0.98507
c1000_d1000	accuracy: 0.786	accuracy: 0.932	accuracy: 0.994	accuracy: 0.9975
	F1 score: 0.79304	F1 score: 0.93274	F1 score: 0.99400	F1 score: 0.99750
c1000_d5000	accuracy: 0.8479	accuracy: 0.9555	accuracy: 0.9965	accuracy: 0.9996
	F1 score: 0.84810	F1 score: 0.95537	F1 score: 0.99650	F1 score: 0.99960
c1500_d100	accuracy: 0.885	accuracy: 0.965	accuracy: 1.0	accuracy: 1.0
	F1 score: 0.88780	F1 score: 0.96447	F1 score: 1.0	F1 score: 1.0
c1500_d1000	accuracy: 0.9085	accuracy: 0.9815	accuracy: 0.9995	accuracy: 1.0
	F1 score: 0.91163	F1 score: 0.98137	F1 score: 0.9995	F1 score: 1.0
c1500_d5000	accuracy: 0.9565	accuracy: 0.9887	accuracy: 0.9998	accuracy: 1.0
	F1 score: 0.95658	F1 score: 0.98868	F1 score: 0.9998	F1 score: 1.0
c1800_d100	accuracy: 0.95	accuracy: 0.97	accuracy: 0.995	accuracy: 0.99
	F1 score: 0.95098	F1 score: 0.96970	F1 score: 0.99497	F1 score: 0.99
c1800_d1000	accuracy: 0.973	accuracy: 0.995	accuracy: 1.0	accuracy: 1.0
	F1 score: 0.97332	F1 score: 0.99499	F1 score: 1.0	F1 score: 1.0
c1800_d5000	accuracy: 0.9837	accuracy: 0.997	accuracy: 1.0	accuracy: 0.9999
	F1 score: 0.98382	F1 score: 0.997	F1 score: 1.0	F1 score: 0.99990

Strategy for tuning hyperparameters: If we use lists of values for params, where each param list has let's say 3 or 4 values and we have 3-5 params changed for each classifier, then we have 3^4 or similar number of parameter combinations for each classifier. I did not use this strategy as I believe this would take a lot of time to iterate over all the models with param combinations.
So, instead I crafted a list of models for each classifier, where each list contains about 14 - 18 models with different param combinations.

The best parameters chosen after tuning the hyperparameters for each dataset and classifier can be found in the **output.txt** file.

Questions

• Which classifier (among the four) yields the best overall generalization accuracy/F1 score? Based on your ML knowledge, why do you think the "classifier" achieved the highest overall accuracy/F1 score.
- Gradient Boosting Ensemble Classifier yields the best overall accuracy and F1 score, although Random Forest Ensemble Classifier comes a close second. With more features, both these classifiers have almost the same performance. I prefer Random Forest Classifier for this dataset as it provides particularly good performance with a lot less training time compared to Gradient Boosting Classifier.
- Gradient Boosting may have better performance than other models because it includes a learning rate which controls the contribution of each weak learner. A lower learning rate prevents overfitting and thus has better generalization. Also, the trees in Gradient Boosting are shallow and pruned whereas each tree in Random Forest can grow independently, and can be deep, thus Gradient Boosting may have better generalization.

• What is the impact of increasing the amount of training data on the accuracy/F1 scores of each of the four classifiers.
- The accuracy and F1 score increase significantly as the number of examples in the dataset grows for Decision Tree and Bagging Classifiers, we can observe this in rows of the table where no. of clauses is the same but dataset size increases, e.g., c1000_d100, c1000_d1000, c1000_d5000. Compare this with Random Forest and Gradient Boosting, where both have superior performance even when dataset size is low, e.g., c1000_d100, c1500_d100. If we consider c300 and c500, the performance gradually increases with increasing dataset size, but for d100, both still have worse performance.

• What is the impact of increasing the number of features on the accuracy/F1 scores of each of the four classifiers.
- The accuracy and F1 score increase significantly as the number of features in the dataset increases for Decision Tree and Bagging Classifiers. Ofcourse they have better performance if the dataset has more examples, but even with smaller dataset, the impact of the number of features can be observed in c300_d100, c500_d100, c1000_d100, c1500_d100, c1800_d100.
Similar behavior is observed for Random Forest and Gradient Boosting but only with smaller number of features, i.e., c300_d100, c500_d100. With 1000 features, we already have excellent performance, but as we reach c1500, c1800 we reach almost 100% accuracy.

Part 7. Accuracy for MNIST dataset and each classifier

DecisionTree	Bagging	Random Forest	Gradient Boosting
0.8875	0.9554	0.9692	0.9619

Questions

• Which classifier among the four yields the best classification accuracy on the MNIST dataset and why?
- Random Forest Ensemble Classifier yields the best performance on the MNIST dataset with 96.92% accuracy, although Gradient Boosting has performance is a close second i.e., 96.19%. Random Forest outperforms Gradient Boosting in the MNIST dataset because Gradient Boosting is sensitive to outliers and noisy data which may be prevalent in the MNIST dataset, where digits have been centered inside 28x28 pixel images. Random Forest is more robust to noise due to its ensemble averaging effect. Furthermore, Gradient Boosting is sensitive to hyperparameter tuning, and with careful tuning it may be able to outperform Random Forest.
- Random Forest also has a drastically low training time compared to Gradient Boosting. Gradient Boosting Classifier took close to 7.5 hours whereas Random Forest took only 10 mins to train & tune hyperparameters. Random Forest also supports parallelization where Gradient Boost cannot be parallelized since the performance of each tree in the ensemble is dependent on the performance of the previous trees.

The best parameters chosen after tuning the hyperparameters for the MNIST dataset and each classifier can be found in the **mnist_output.txt** file.