Project 2 Report:

Experiment 1: Decision Tree Classifier Parameters:

Table 3: Decision Tree Classifier Parameters

Dataset	Criterion	Max_Depth	Max_features	Min_samples Leaf	min_samples_split
c300_d100	Entropy	6	None	0.01	2
c300_d1000	Gini	6	None	1	0.05
c300_d5000	Entropy	8	None	5	2
c500_d100	Gini	6	sqrt	0.01	0.01
c500_d1000	Gini	6	None	0.01	0.01
c500_d5000	Log_loss	9	None	10	2
c1000_d100	Entropy	9	None	0.01	0.02
c1000_d1000	Log_loss	8	None	5	2
c1000_d5000	Entropy	10	None	5	2
c1500_d100	Entropy	11	sqrt	0.01	0.1
c1500_d1000	Entropy	7	None	1	0.01
c1500_d5000	Log_loss	10	none	1	2
c1800_d100	Gini	6	None	5	2
c1800_d1000	Log_loss	13	None	1	2
c1800_d5000	Entropy	9	None	5	2

Experiment 2: Bagging with Decision Trees (Parameters):

Table 4: Bagging Parameters

Dataset	n_estimators	max_samples
c300_d100	40	0.15
c300_d1000	50	0.2
c300_d5000	50	0.25
c500_d100	50	0.25
c500_d1000	50	0.15
c500_d5000	50	0.05
c1000_d100	50	0.1
c1000_d1000	30	0.05
c1000_d5000	50	0.2
c1500_d100	40	0.25
c1500_d1000	40	0.05
c1500_d5000	50	0.1
c1800_d100	40	0.25
c1800_d1000	50	0.05
c1800_d5000	50	0.25

Experiment 3: Random Forest (Parameters)

Table 5: Parameters for Random Forest Classifier

Dataset	N_Estimators	Criterion	Max_depth	Min_Samples_Split	Min_Samples_Leaf	Max_features	Max_samples
c300_d100	150	log_loss	15	0.1	0.01	log2	0.9
c300_d1000	150	log_loss	5	0.05	0.01	sqrt	0.9
c300_d5000	150	Entropy	15	0.01	0.01	log2	0.9
c500_d100	150	Entropy	5	0.01	0.01	sqrt	0.75
c500_d1000	150	gini	10	0.01	0.01	log2	0.9
c500_d5000	150	gini	15	0.01	0.01	log2	0.75
c1000_d100	150	gini	15	0.05	0.05	log2	0.5
c1000_d1000	150	gini	10	0.01	0.01	log2	0.5
c1000_d5000	150	Entropy	10	0.01	0.01	log2	0.75
c1500_d100	50	gini	5	0.01	0.01	sqrt	0.5
c1500_d1000	50	gini	5	0.01	0.01	log2	0.75
c1500_d5000	150	Entropy	5	0.01	0.05	log2	0.5
c1800_d100	50	gini	5	0.01	0.01	sqrt	0.5
c1800_d1000	50	gini	5	0.01	0.01	sqrt	0.5
c1800_d5000	50	gini	5	0.01	0.01	log2	0.5

Experiment 4: Gradient Boosting

Table 6: Parameters for Gradient Boosting

Dataset	Criterion	Learning rate	Loss	max_depth	max_features	n_estimators
c300_d100	squared error	0.05	log_loss	3	sqrt	150
c300_d1000	friedman_mse	0.25	log_loss	3	None	150
c300_d5000	friedman_mse	0.25	log_loss	3	None	150
c500_d100	friedman_mse	0.05	log_loss	3	log2	150
c500_d1000	friedman_mse	0.25	log_loss	3	None	150
c500_d5000	friedman_mse	0.25	log_loss	3	none	150
c1000_d100	friedman_mse	0.25	log_loss	3	sqrt	100
c1000_d1000	friedman_mse	0.25	exponential	3	log2	150
c1000_d5000	friedman_mse	0.25	log_loss	3	None	150
c1500_d100	friedman_mse	0.01	log_loss	3	sqrt	100
c1500_d1000	friedman_mse	0.01	log_loss	3	log2	100
c1500_d5000	friedman_mse	0.01	exponential	3	log2	150
c1800_d100	friedman_mse	0.01	log_loss	1	log2	100
c1800_d1000	friedman_mse	0.01	log_loss	1	log2	150
c1800_d5000	friedman_mse	0.01	log_loss	3	log2	50

Part 5: Accuracy, F1 Scores, and Analysis

Table 1: Classification Accuracy

Dataset	Decision Tree	Bagging	Random Forest	Gradient Boosting
c300_d100	0.635	0.675	0.79	0.86
c300_d1000	0.673	0.835	0.859	0.99
c300_d5000	0.78	0.898	0.9	0.997
c500_d100	0.59	0.82	0.89	0.89
c500_d1000	0.707	0.881	0.957	0.997
c500_d5000	0.793	0.888	0.949	0.999
c1000_d100	0.735	0.905	0.97	0.96
c1000_d1000	0.805	0.934	0.995	0.998
c1000_d5000	0.865	0.959	0.995	0.999
c1500_d100	0.78	0.97	1	1
c1500_d1000	0.932	0.985	1	1
c1500_d5000	0.955	0.991	0.9998	1
c1800_d100	0.94	0.99	1	0.995
c1800_d1000	0.974	0.995	0.9995	1
c1800_d5000	0.984	0.997	1	1

Table 2: F1 Scores

Dataset	Decision Tree	Bagging	Random Forest	Gradient Boosting
c300_d100	0.651	0.652	0.79	0.863
c300_d1000	0.69	0.833	0.856	0.99
c300_d5000	0.798	0.901	0.9	0.997
c500_d100	0.59	0.818	0.892	0.893
c500_d1000	0.72	0.881	0.958	0.997
c500_d5000	0.804	0.888	0.949	0.999
c1000_d100	0.728	0.911	0.97	0.96
c1000_d1000	0.813	0.935	0.995	0.998
c1000_d5000	0.869	0.959	0.995	0.9999
c1500_d100	0.782	0.971	1	1
c1500_d1000	0.932	0.985	1	1
c1500_d5000	0.956	0.991	0.9998	1
c1800_d100	0.94	0.99	1	0.995
c1800_d1000	0.974	0.995	0.9995	1
c1800_d5000	0.984	0.997	1	1

QUESTIONS:

1. Overall, the gradient boosting and the random forest classifiers had significantly better overall generalization than the decision tree and bagging. The best one in my experiments was the gradient boosting. This is because gradient boosting is a model which minimizes the weight of its inaccurate learners and maximizes the weight of

the more accurate learners. None of the other classification methods do this. Therefore, it is expected that this method would be better

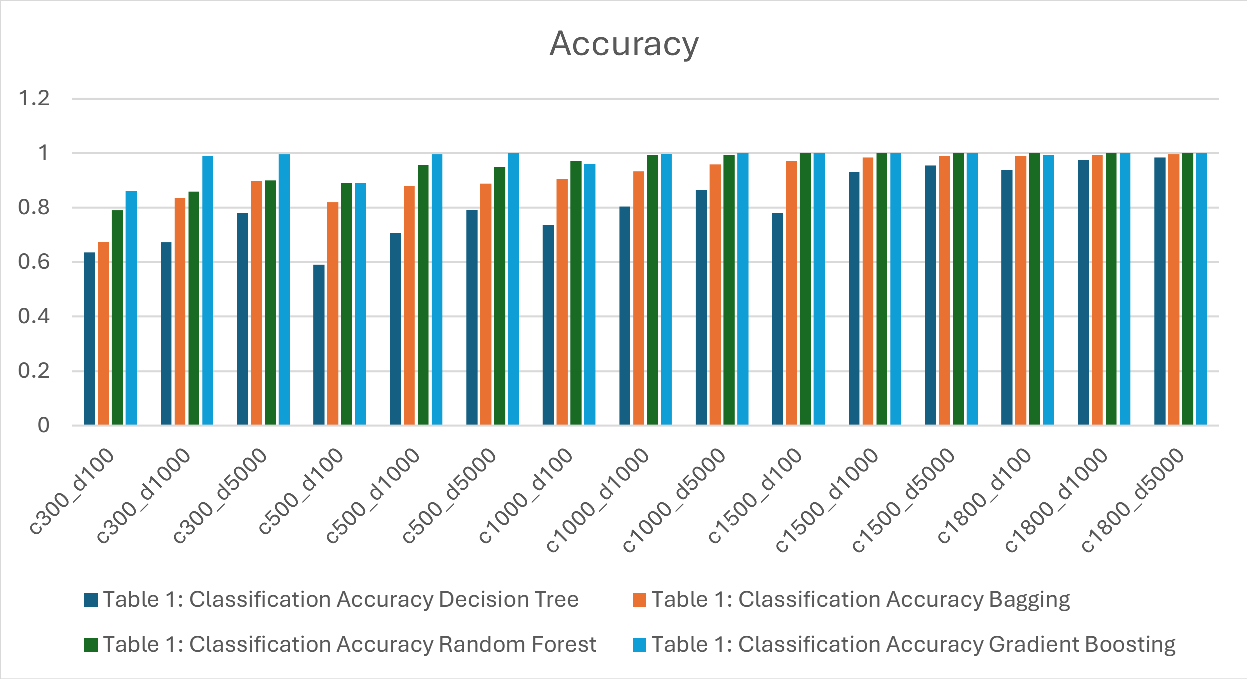
2. Increasing the training data size significantly increases the accuracy and F1 score for each classifier across the board. This is consistent with our understanding of machine learning that increasing the data reduces the error. Since there are more datapoints to train the model, the model has a better understanding of the underlying patterns.
3. Increasing the number of features also significantly increase the accuracy of all of the models. My theory is that, since the data is based on Boolean logic in conjunctive normal form, the output of the samples are constrained by that logic. As you increase the number of clauses, and thereby make that logic more strict, you reduce the variability of the output, and thus it becomes easier to classify.

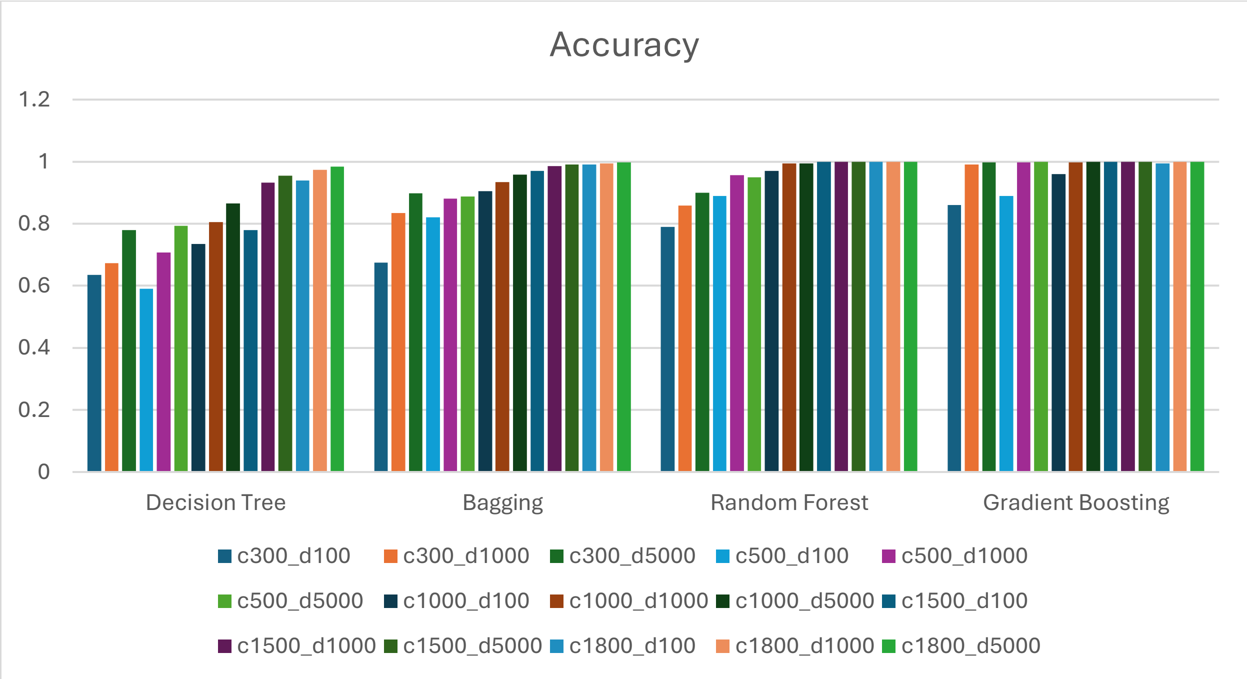
Part 6: MNIST Dataset

Table 7: Accuracy for MNIST Dataset	
Method	Accuracy
Decision Tree	0.8368
Bagging	0.9666
Random Forest	0.9661
Gradient Boosting	0.9728

Table 8: Best Parameters for MNIST Dataset						
Method	Criterion	Max_depth	Max_features	N_estimators	Max_samples	Learning Rate
Decision Tree	Entropy	20	sqrt			
Bagging	Entropy	20	sqrt	100		
Random Forest	Gini	20	sqrt	200	0.75	
Gradient Boosting		5	sqrt	200		0.2

Visualizations





ANALYSIS:

The most accurate model on this dataset is the gradient boosting classifier. However, it was similar in accuracy to the random forest and the bagging method, with each of these three methods differing by less than 1%.

The decision tree classifier is the simplest model, and only creates one estimator for the data. For this reason, we can expect the inferior results. The bagging method, which used 100 estimators instead of just one, improved the accuracy by 13%. This indicates that the model was unstable, and that there was high variance, which bagging is intended to reduce. In the random forest model, we saw a slight decrease in the accuracy by about .05%. This was fascinating because there was twice as many estimators created in my random forest as there was in the bagging method, and both classifiers created models of the same maximum depth (20) and the same number of features (sqrt). The one difference between the two hyperparameter configurations was that the maximum number of samples per estimator was less for the random forest, with the max samples of the bagging being 100% and the max samples of the random forest being 75%. However, I did include 100% max samples in my hyperparameter tuning for the random forest, and yet the grid search I performed chose 75% as the best. This still cannot explain the poorer performance of the random forest since there was twice as many estimators as there was for the bagging. From this, I must conclude that at a certain point, adding more estimators, and thus getting a wider variety from the data, does not help improve accuracy. The gradient boosting method did slightly better. This is consistent with our previous experiments with the first datasets. This was fascinating because the model had the same number of estimators and max features as the random forest, but each estimator had only a max depth of 5. This makes sense because, in gradient boosting, weak learners carry less weight than stronger learners, and so the model can effectively learn what matters and what does not, as opposed to selecting for features and samples randomly. However, the increase in improvement from bagging was less than 1%, and the cost in terms of computing time was noticeably higher. I suspect that if I added more estimators, then the overall accuracy would increase. However, the increased cost in terms of computing time is significant. There does seem to be a tradeoff between computing time and accuracy, in which the cost of slightly higher accuracy is significantly more computing time. My conclusion is that this tradeoff one of the fundamental realities of machine learning.