Homework 3

Machine Learning

A. Part - 1

1. DecisionTreeClassifier

Files	Criterion	Splitter	Max features	Max leaf nodes	Min sample leaf	Min samples split	**Accuracy**	**F1 score**
c300_d100	Gini	Random	Auto	30	1	2	0.590	0.602
c500_d100	Entropy	Random	Sqrt	500	1	0.5	0.640	0.581
c1000_d100	Gini	Random	None	1000	1	2	0.645	0.667
c1500_d100	Entropy	Random	Auto	1500	1	2	0.810	0.822
c1800_d100	Entropy	Random	None	None	1	0.5	0.940	0.942
c300_d1000	Gini	Random	None	30	1	2	0.692	0.715
c500_d1000	Gini	Random	None	50	1	2	0.712	0.715
c1000_d1000	Gini	Random	None	100	1	2	0.820	0.825
c1500_d1000	Entropy	Random	None	150	1	2	0.917	0.919
c1800_d1000	Gini	Best	None	1800	1	0.4	0.966	0.965
c300_d5000	Gini	Best	None	300	1	2	0.788	0.799
c500_d5000	Gini	Best	None	500	1	2	0.796	0.804
c1000_d5000	Gini	Best	None	100	1	2	0.857	0.854
c1500_d5000	Gini	Random	None	150	1	2	0.957	0.956
c1800_d5000	Entropy	Random	None	None	1	2	0.983	0.983

Figure 1 – Hyperparameter tuning and performance metrics on DecisionTreeClassifier

2. BaggingClassifier with DecisionTreeClassifier as base estimator

Train files	Max features	Max samples	N estimators	Bootstrap	Bootstrap features	Warm start	**Accuracy**	**F1 score**
c300_d100	500	100	100	F	T	F	0.770	0.774
c500_d100	50	100	100	F	T	F	0.500	0.667
c1000_d100	50	100	100	F	T	F	0.975	0.975
c1500_d100	50	100	50	T	T	T	0.990	0.990
c1800_d100	1	100	100	F	T	F	0.500	0.667
c300_d1000	50	1000	1000	T	T	F	0.904	0.905
c500_d1000	50	1000	50	F	T	F	0.911	0.911
c1000_d1000	50	1000	50	F	T	F	0.989	0.989
c1500_d1000	50	1000	50	T	F	F	0.999	0.999
c1800_d1000	50	1000	50	F	T	F	0.500	0.667
c300_d5000	50	5000	50	F	T	F	0.857	0.859
c500_d5000	50	5000	50	F	F	T	0.920	0.921
c1000_d5000	50	5000	50	F	T	F	0.994	0.993
c1500_d5000	50	5000	50	F	T	F	0.999	0.999
c1800_d5000	50	5000	50	T	T	T	0.999	0.999

Figure 2 – Hyperparameter tuning and performance metrics in BaggingClassifer with DecisionTreeClassifier as base estimator

3. RandomForestClassifier

Train files	Bootstrap	Criterion	Max depth	Max features	Max leaf nodes	Max samples	N estimators	Warm start	**Accuracy**	**F1 score**
c300_d100	F	G	300	Auto	None	0.4	100	T	0.785	0.786
c500_d100	T	G	1000	Auto	1000	None	100	F	0.865	0.860
c1000_d100	T	G	1000	Auto	1000	None	100	T	0.975	0.975
c1500_d100	T	G	None	Auto	1500	None	100	T	1.0	1.0
c1800_d100	T	G	None	Auto	300	0.4	100	T	1.0	1.0
c300_d1000	F	G	300	Auto	300	None	200	F	0.887	0.887
c500_d1000	F	G	None	Auto	500	0.4	500	F	0.959	0.959
c1000_d1000	F	E	300	Sqrt	None	0.4	500	F	0.994	0.994
c1500_d1000	T	G	300	Auto	None	None	500	F	1.0	1.0
c1800_d1000	F	G	300	Auto	1800	0.4	500	T	1.0	1.0
c300_d5000	F	E	300	Sqrt	None	None	500	F	0.938	0.939
c500_d5000	F	G	None	Auto	500	None	200	F	0.948	0.948
c1000_d5000	F	G	300	Sqrt	None	0.4	100	F	0.993	0.993
c1500_d5000	T	E	300	Auto	1500	0.4	100	T	0.996	0.996
c1800_d5000	F	G	300	Auto	1800	None	100	T	1.0	1.0

Below initials are used in the table –

G – Gini, E – Entropy
T – True, F - False

Figure 3 – Hyperparameter tuning and performance metrics in RandomForestClassifier

4. GradientBoostingClassifier

Train files	Loss	Learning rate	N estimators	Criterion	Max depth	Max features	Max leaf nodes	Warm start	**Accuracy**	**F1 score**
c300_d100	D	0.1	100	MSE	3	None	300	T	0.850	0.851
c500_d100	E	0.1	15	MSE	3	Sqrt	None	T	0.785	0.794
c1000_d100	E	0.5	100	F	3	Sqrt	1000	F	0.990	0.990
c1500_d100	D	0.1	100	MAE	3	Sqrt	None	F	1.0	1.0
c1800_d100	D	0.5	100	MSE	3	None	1800	F	0.995	0.995
c300_d1000	E	0.5	15	MSE	3	None	None	T	0.894	0.897
c500_d1000	E	0.5	15	MSE	3	None	None	T	0.930	0.931
c1000_d1000	E	0.1	1000	MSE	3	Sqrt	None	F	0.998	0.998
c1500_d1000	D	0.5	100	MSE	3	Auto	10	T	1.0	1.0
c1800_d1000	E	0.5	100	MSE	10	None	10	F	1.0	1.0
c300_d5000	D	0.5	100	MSE	3	Sqrt	300	F	0.919	0.920
c500_d5000	D	0.1	100	MSE	3	Sqrt	10	T	0.927	0.928
c1000_d5000	E	0.1	100	MSE	10	Auto	10	F	0.997	0.998
c1500_d5000	E	0.5	100	MSE	3	Auto	10	F	1.0	1.0
c1800_d5000	D	0.1	100	MSE	3	Auto	10	F	1.0	1.0

Below initials are used in the table –

D – Deviance, E – Exponential
F – Friedman_mse, S – Squared_error
T – True, F - False

After c1000_d1000, due to large tuning time, n_estimator was limited by 100.

Figure 4 – Hyperparameter tuning and performance metrics in GradientBoostingClassifier

5. Questions -

a. 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

Theoretical understanding – Boosting should give the best overall performance. Boosting trains on data sequentially to increase accuracy (and in-turn to decrease bias). It uses *weighted examples.* And usually reweighing performs better than resampling.

Empirical observation – With average smaller prediction time, *Gradient Boosting Classifier* achieved the best overall generalization accuracy.

b. What is the impact of increasing the amount of training data on the accuracy/F1 scores of each of the four classifiers?

Theoretical understanding – For each classifier, increasing training data should increase the accuracy – better performance.

Empirical observation – For each classifier, upon increasing the number of examples d, we are seeing improvement in performance.

c. What is the impact of increasing the number of clauses on the accuracy/F1 scores of each of the four classifiers?
Theoretical understanding – For each classifier, increasing the number of clauses should try to fit the underlying *CNF (Conjunctive Normal Form)* better, yielding better performance/improved accuracy.

Empirical observation - For each classifier, we are seeing improved performance with increasing number of clauses c.

6. Evaluate these four classifiers on MNIST dataset

Parameters	DecisionTree	Bagging	RandomForest	GradientBoosting
Criterion	Entropy		Gini	MSE
Splitter	Best			
Max features	None	50	Auto	Auto
Max leaf nodes	None		200	None
Max samples		300	0.4	
N estimators		50	20	20
Bootstrap		True	False	
Bootstrap features		False		
Warm start		True	True	False
Max depth			None	3
Loss				Deviance
Learning rate				0.1
Accuracy	0.884	0.866	0.925	0.889

Grayed out cells are using default values of parameters or not applicable for that classifier
Accuracy of GradientBoosting is acquired in *shorter time* than DecisionTree and Bagging classifiers. Better accuracy could be achieved with more aggressive parameter setting – n_estimators = 200, max_features = None

Figure 5 – Hyperparameter tuning and performance metrics in each of the classifier mentioned above on MNIST dataset

B. Part – 2 (K-Means clustering)

1. Compressed images are being displayed in the source code (Jupyter Notebook) itself.
2. A table for compression ratio is given below –

Image	K (# of clusters)					
		2	5	10	15	20
Koala (763 KB)	First	5.76	5.93	4.58	4.37	4.73
	Second	11.42	4.40	4.59	4.75	4.47
	Third	13.97	4.88	4.40	4.59	4.45
	Average	10.38	5.07	4.52	4.57	4.55
	Variance	11.77	0.4082	0.0076	0.0242	0.0162
Penguin (760 KB)	First	8.60	7.70	6.20	6.80	6.24
	Second	9.09	6.74	7.04	6.86	6.48
	Third	9.07	8.15	6.39	6.23	6.37
	Average	8.92	7.53	6.54	6.63	6.36
	Variance	0.0512	0.3458	0.1295	0.0806	0.0096

Figure 6 – Different compression ratio as per different initialization (indicated by ordinal numbers)

Note – The following definition of Compression ratio is being considered.
Compression ratio = size(original image) / size(compressed image)

3. Yes, there is a trade-off between image quality and degree of compression. Good value for K is highlighted in the table (figure 6) (K = 15).

Appending a substitute table with the following definition of

Compression Ratio = size of reconstructed image / size of an original image

Image	K (# of clusters)					
		2	5	10	15	20
Koala (763 KB)	First	0.1736	0.1686	0.2183	0.2288	0.2114
	Second	0.0875	0.2272	0.2178	0.2105	0.2237
	Third	0.0715	0.2049	0.2272	0.2178	0.2247
	Average	0.1108	0.2002	0.2211	0.2190	0..2199
	Variance	0.0020	0.0005	1.84E-05	5.65E-05	3.65E-05
Penguin (760 KB)	First	0.1162	0.1298	0.1612	0.1470	0.1605
	Second	0.1100	0.1483	0.1420	0.1457	0.1543
	Third	0.1102	0.1226	0.1564	0.1605	0.1569
	Average	0.1121	0.13335	0.1532	0.1510	0.1572
	Variance	8.27E-06	0.0001	6.65E-05	4.47E-05	6.46E-06

Figure 7 – Different compression ratio as per different initialization (indicated by ordinal numbers)