Homework-3

Tree Classifiers

1. sklearn.tree.DecisionTreeClassifier

- Using GridSearchCV to get the best parameter for Decision TreeClassifier.
- Parameters used for GridSearchCV are
    - "criterion": ["gini", "entropy"],
    - "splitter" : ["best", "random"],
    - "max_features" : [None, "auto", "sqrt", "log2", len(X_Train[0])],
    - "max_depth": [None, 80, 100, 120, 150, 200],
    - "class_weight" : [None, "balanced"]

Clauses	Samples	class_weight	criterion	max_depth	max_features	splitter	Accuracy	F1_Score
300	100	balanced	gini	200	500	best	0.585	0.6175
500	100	None	entropy	120	None	best	0.665	0.6528
1000	100	balanced	gini	200	None	best	0.7	0.7170
1500	100	balanced	entropy	None	sqrt	random	0.835	0.8406
1800	100	None	entropy	100	None	best	0.915	0.9202
300	1000	balanced	entropy	150	None	best	0.6755	0.6823
500	1000	balanced	entropy	100	500	best	0.688	0.6914
1000	1000	None	entropy	150	None	best	0.7935	0.8044
1500	1000	balanced	gini	150	500	best	0.924	0.9260
1800	1000	None	entropy	200	500	best	0.977	0.9772
300	5000	balanced	gini	120	None	best	0.7614	0.7591
500	5000	balanced	gini	100	None	best	0.7749	0.7777
1000	5000	balanced	entropy	200	500	best	0.8405	0.8436
1500	5000	balanced	entropy	200	500	best	0.9603	0.9715
1800	5000	None	entropy	200	500	best	0.981	0.9812

2. sklearn.ensemble.BaggingClassifier

- Using the best parameters obtained from Above DecisionTreeClassifier.
- n_estimators=15, random_state=False,bootstrap_features=True, warm_start=True

Clauses	Samples	Accuracy	F1_Score
300	100	0.69	0.7102803738
500	100	0.72	0.7307692308
1000	100	0.9	0.9029126214
1500	100	0.98	0.98
1800	100	0.98	0.98
300	1000	0.7915	0.794277257
500	1000	0.8345	0.8365432099
1000	1000	0.9305	0.9312221672

1500	1000	0.988	0.9880239521
1800	1000	0.998	0.997995992
300	5000	0.8723	0.8737518537
500	5000	0.8946	0.8942828485
1000	5000	0.9661	0.9662249676
1500	5000	0.9947	0.9946973487
1800	5000	0.999	0.9989991994
	Average	0.9026133333	0.9053320681

- n_estimators=15, random_state=True,bootstrap_features=True, warm_start=True

Clauses	Samples	Accuracy	F1_Score
300	100	0.7	0.7087378641
500	100	0.675	0.6948356808
1000	100	0.855	0.8585365854
1500	100	0.955	0.9552238806
1800	100	0.985	0.9847715736
300	1000	0.7875	0.7892910263
500	1000	0.828	0.8308751229
1000	1000	0.9345	0.9353724716
1500	1000	0.9875	0.9874812218
1800	1000	0.9955	0.9954932399
300	5000	0.8827	0.8834111917
500	5000	0.8949	0.895950896
1000	5000	0.9647	0.9648581384
1500	5000	0.9918	0.9918065548
1800	5000	0.9989	0.99889989
	Average	0.8957333333	0.8983696892

- **(BEST PARAMETERS) n_estimators=20, random_state=True,bootstrap_features=True, warm_start=True**

Clauses	Samples	Accuracy	F1_Score
300	100	0.695	0.6839378238
500	100	0.73	0.7096774194
1000	100	0.9	0.8958333333
1500	100	0.975	0.9751243781
1800	100	0.985	0.9849246231
300	1000	0.793	0.7825630252
500	1000	0.8565	0.8543886352
1000	1000	0.941	0.941
1500	1000	0.99	0.9899699097

1800	1000	0.995	0.994994995
300	5000	0.8974	0.8952419849
500	5000	0.9093	0.9082448154
1000	5000	0.9719	0.9718803162
1500	5000	0.9936	0.993597439
1800	5000	0.9988	0.9987995198
	Average	0.9087666667	0.9053452145

- n_estimators=10, random_state=True, bootstrap_features=True, warm_start=True, reducing estimators reduced accuracy

Clauses	Samples	Accuracy	F1_Score
300	100	0.665	0.612716763
500	100	0.68	0.6235294118
1000	100	0.85	0.8469387755
1500	100	0.955	0.9552238806
1800	100	0.98	0.9797979798
300	1000	0.749	0.7235682819
500	1000	0.809	0.7997903564
1000	1000	0.9135	0.9129340715
1500	1000	0.985	0.9849094567
1800	1000	0.9925	0.9924736578
300	5000	0.8376	0.8275276126
500	5000	0.867	0.8614294645
1000	5000	0.9495	0.9489331581
1500	5000	0.989	0.9889779559
1800	5000	0.9985	0.9984992496
	Average	0.8813733333	0.8704833384

3. sklearn.ensemble. GradientBoostingClassifier

a) loss: *deviance, learning_rate=0.1, n_estimators=50*

Clauses	Samples	Accuracy	F1_Score
300	100	0.835	0.8374384236
500	100	0.865	0.8682926829
1000	100	0.925	0.9253731343
1500	100	0.99	0.99
1800	100	0.99	0.99
300	1000	0.937	0.9391891892
500	1000	0.948	0.948969578
1000	1000	0.978	0.9780876494
1500	1000	0.997	0.997

1800	1000	0.998	0.998003992
300	5000	0.9476	0.9494208494
500	5000	0.9553	0.9563433929
1000	5000	0.9881	0.988165092
1500	5000	0.9979	0.997901469
1800	5000	0.9997	0.99970009
	Average	14.3516	14.36388554

b) loss: *deviance, learning_rate=0.1, n_estimators=100*

Clauses	Samples	Accuracy	F1_Score
300	100	0.85	0.8514851485
500	100	0.87	0.8737864078
1000	100	0.945	0.9447236181
1500	100	1	1
1800	100	0.99	0.99
300	1000	0.9725	0.9732099367
500	1000	0.9845	0.9846458643
1000	1000	0.9935	0.9935355545
1500	1000	1	1
1800	1000	1	1
300	5000	0.9828	0.9830808578
500	5000	0.9866	0.9867379256
1000	5000	0.9972	0.9972055888
1500	5000	0.9999	0.99990001
1800	5000	0.9999	0.99990001
	Average	14.5719	14.57821092

c) (BEST Parameters) loss: *deviance, learning_rate=0.1, n_estimators=150*

Clauses	Samples	Accuracy	F1_Score
300	100	0.845	0.8487804878
500	100	0.88	0.8834951456
1000	100	0.955	0.9547738693
1500	100	1	1
1800	100	0.995	0.9950248756
300	1000	0.988	0.9881422925
500	1000	0.9905	0.9905707196
1000	1000	0.9955	0.9955156951
1500	1000	1	1
1800	1000	1	1
300	5000	0.9916	0.9916699722
500	5000	0.9932	0.9932405567

1000	5000	0.9992	0.9992006395
1500	5000	1	1
1800	5000	1	1
	Average	**14.633**	**14.64041425**

4. sklearn.ensemble. RandomForestClassifier

a) n_estimators=100, criterion=gini

Clauses	Samples	Accuracy	F1_Score
300	100	0.76	0.7692307692
500	100	0.85	0.8469387755
1000	100	0.98	0.9801980198
1500	100	1	1
1800	100	1	1
300	1000	0.8395	0.8377968671
500	1000	0.929	0.9288577154
1000	1000	0.989	0.9889779559
1500	1000	0.9995	0.9994997499
1800	1000	1	1
300	5000	0.8979	0.89797142
500	5000	0.9452	0.9454834859
1000	5000	0.9923	0.9923038481
1500	5000	0.9996	0.99959992
1800	5000	1	1
	Average	0.9454666667	0.9457905685

b) n_estimators=150, criterion=gini

Clauses	Samples	Accuracy	F1_Score
300	100	0.765	0.7834101382
500	100	0.905	0.9025641026
1000	100	0.975	0.9748743719
1500	100	1	1
1800	100	1	1
300	1000	0.8655	0.8650275966
500	1000	0.9395	0.9395906141
1000	1000	0.995	0.995004995
1500	1000	1	1
1800	1000	1	1
300	5000	0.9143	0.9150391593
500	5000	0.9541	0.9543419875

1000	5000	0.9937	0.9937031484
1500	5000	0.9999	0.99989999
1800	5000	1	1
	Average	0.9538	0.9548970736

c) *n_estimators=150, criterion='entropy'*

Clauses	Samples	Accuracy	F1_Score
300	100	0.695	0.6871794872
500	100	0.83	0.8316831683
1000	100	0.955	0.9547738693
1500	100	0.995	0.9949748744
1800	100	1	1
300	1000	0.78	0.7738951696
500	1000	0.8935	0.8929110106
1000	1000	0.9795	0.9795307039
1500	1000	0.9985	0.9984977466
1800	1000	1	1
300	5000	0.8589	0.8579482533
500	5000	0.9261	0.9258999298
1000	5000	0.99	0.9899879856
1500	5000	0.9996	0.99959992
1800	5000	1	1
	Average	0.92674	0.9257921412

d) (BEST PARAMETER) n_estimators=200, criterion=gini'

Clauses	Samples	Accuracy	F1_Score
300	100	0.785	0.7902439024
500	100	0.92	0.9191919192
1000	100	0.99	0.99
1500	100	1	1
1800	100	1	1
300	1000	0.8645	0.8656420426
500	1000	0.9475	0.9477351916
1000	1000	0.993	0.993
1500	1000	0.9985	0.9984977466
1800	1000	1	1
300	5000	0.9207	0.9218642231
500	5000	0.9598	0.9599122457
1000	5000	0.9952	0.9951980792
1500	5000	0.9999	0.99989999

1800	5000	1	1
	Average	**0.9582733333**	**0.9587456894**

Questions

1. GradientBoostingClassifier performs the best among the 4 classifiers. In Gradient boosting the while training errors of each classifiers are fed to next classifiers in other words they current each other's errors since they are better than other classifiers.
2. Increasing Training data increase accuracy in all classifiers
3. Increasing Features data increase accuracy in all classifiers

MNIST Dataset

- GradientBoostingClassifier takes forever to compute due to huge data size, reducing the estimators to 50 gets the result but with low accuracy `80.72%`.

But RandomforestClassifier gives the best accuracy with `97.02`%, with BaggingClassifier close second with `94.83% accuracy, and` DecisionTreeClassifier with `87.8%`.

K-means clustering on images

Questions

- Compressed Images shown below

K = 2

K = 5

K = 10

K = 15

K = 20

K = 2

K = 5

K = 10

K = 15

K = 20

- K values of 2 gives high compression but very lossy.

Koala					
Slno	K	KB	Mean	Variance	Compression
1	2	252	252	0	3.015873016
2	5	382	382.2	1.7	1.989528796
3	10	399	396.4	3.8	1.904761905
4	15	398	398	0.5	1.909547739
5	20	396	395.4	1.8	1.919191919

Penguins					
Slno	K	KB	Mean	Variance	Compression
1	2	149	149	0	5.120805369
2	5	224	223.4	1.2	3.40625
3	10	279	276.6	2.8	2.734767025
4	15	291	288.6	29.3	2.621993127
5	20	288	286.2	13.7	2.649305556

All the images were run 5 times and mean and variance was obtained. Images are attached.

- Yes, there is a trade-off, as the k value increases degree of compression normalises i.e., there won't be much change in degree. I would choose k = 15 as much of the detail is preserved as well as degree of the compression is reasonable.