Implementation Assignment -3 Report

Members:
1) Yichen You
2) Zhe Tian
3) Vijay Tadimeti

Note: We worked through everything together and contributed equally.

General Introduction:

A decision tree is a supervised learning classification algorithm. Decision trees consist of root nodes that represent a population or a sample upon which we perform splitting to divide it on the basis of some conditions. The leaf nodes cannot be further segregated. We use Gini-index to measure the uncertainty and therefore to make each split. After applying the Decision Tree algorithm, we extend our model to Random Forest and Adaboost.

Part -1

Here we implement the decision tree for a training set that contains Optical character recognition data to classify between values of 3 and 5, for the features that are represented by floating point values. We assign labels of +1 to number 3 and -1 to the number 5. For this we create a decision tree that starts at depth of 0 (root node) until a maximum depth of 20.

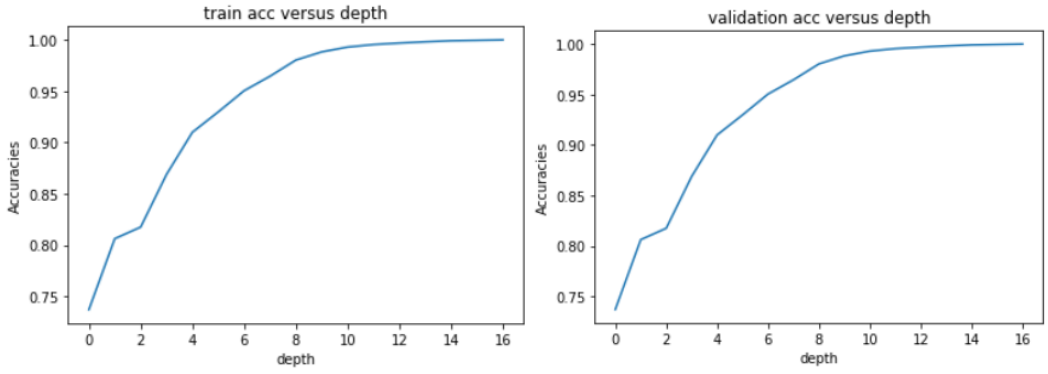
a. We create a decision tree with the required set up
   It should take around 3 minutes on the server. Please be a little patient. There should be two plots saved on the server.

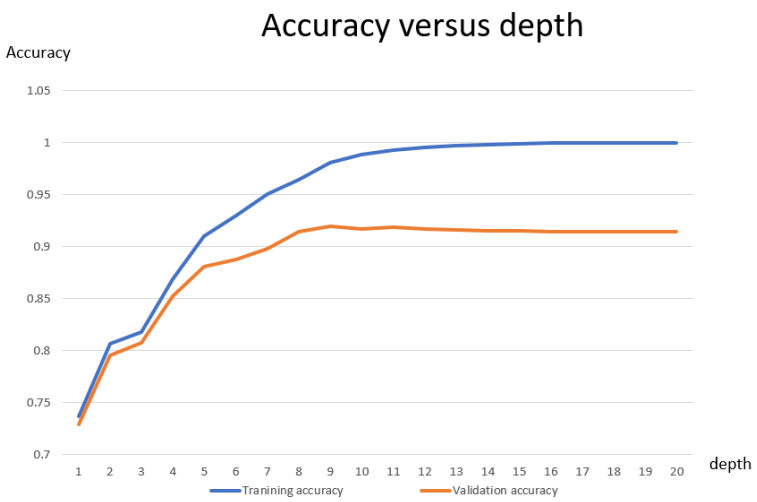
b. See below the computed train and validation accuracy versus depth

	depth	train accuracy
0	1.0	0.737111
1	2.0	0.806260
2	3.0	0.817512
3	4.0	0.868863
4	5.0	0.909984
5	6.0	0.929828
6	7.0	0.950491
7	8.0	0.964607
8	9.0	0.980360
9	10.0	0.988339
10	11.0	0.993044
11	12.0	0.995499
12	13.0	0.996931
13	14.0	0.998159
14	15.0	0.999182
15	16.0	0.999591
16	17.0	1.000000

	depth	validation accuracy
0	1.0	0.909574
1	2.0	0.931874
2	3.0	0.935966
3	4.0	0.950696
4	5.0	0.960311
5	6.0	0.962561
6	7.0	0.966039
7	8.0	0.971563
8	9.0	0.973200
9	10.0	0.972381
10	11.0	0.972995
11	12.0	0.972381
12	13.0	0.972177
13	14.0	0.971768
14	15.0	0.971768
15	16.0	0.971563
16	17.0	0.971563
17	18.0	0.971563
18	19.0	0.971563
19	20.0	0.971563

Plotting the train and validation accuracy versus depth





c. As we can see from the plot, the train accuracy rapidly approaches 100% as the depth increases. The validation accuracy moves closely with the train accuracy for the first few depths but level off at a lower level of accuracy. Also, the train accuracy reaches 100% at depth 17. The validation accuracy becomes stable at 97%.

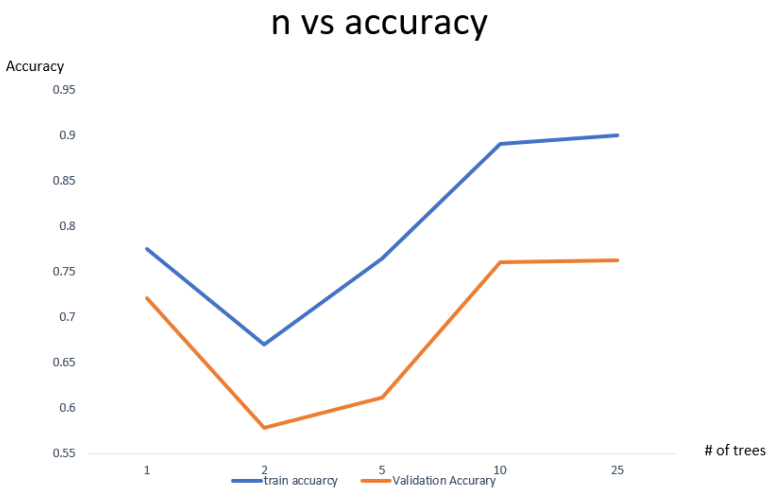
d. According to our computed accuracy, the depth associated with the best validation accuracy is 8. After the depth of 8, the validation accuracy decreases a bit due to overfitting issues.

Part -2

Here we implement the random forest algorithm in this part. Note every time we run through the code, we might get different results due to the randomized nature of random forest. But they should be very similar in the ballpark.

a. We implement the random forest as required
Note: it might take a little longer than the DT.

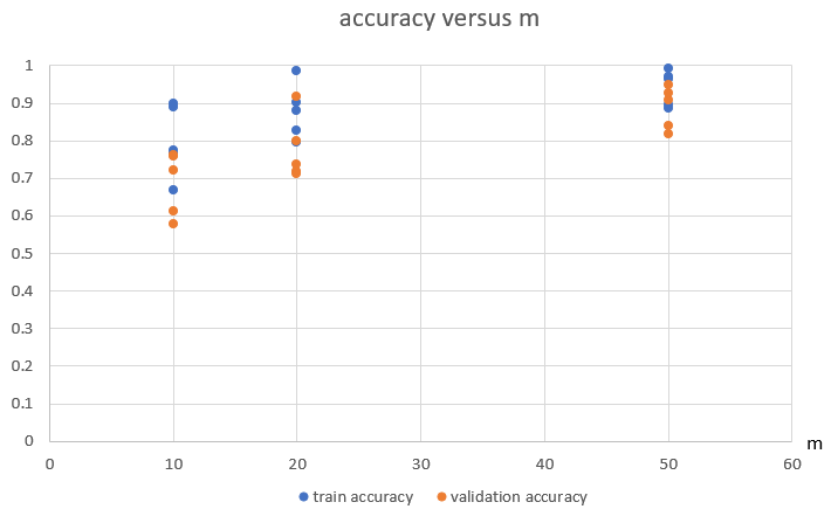
b. For d=9 m=10 and n$\in$ [1,2,5,10,25], we plot the train and validation accuracies versus n

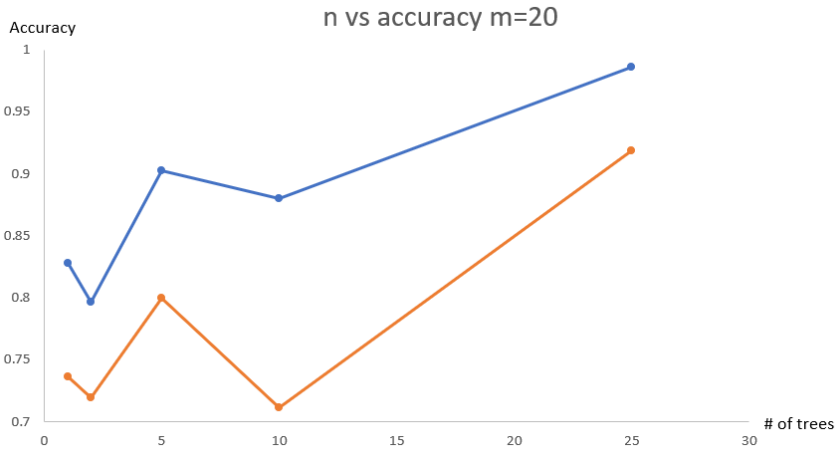


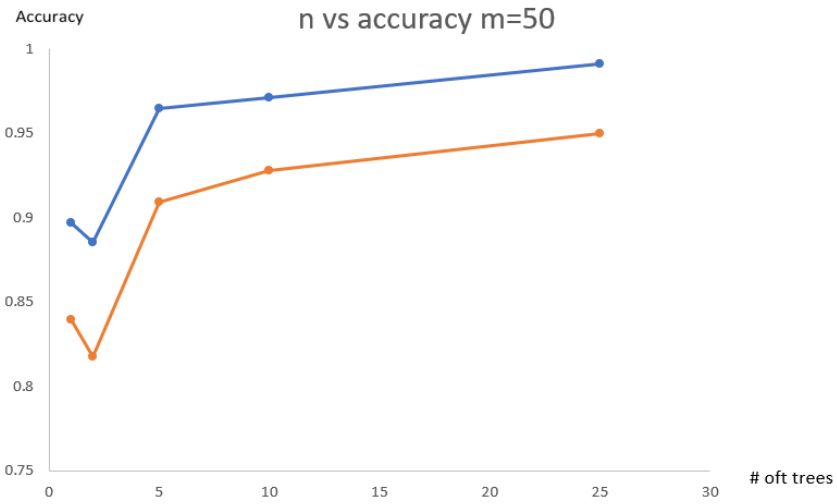
c. The train accuracies are always higher than the validation accuracies. Both train and validation performance get improved with a greater number of trees. In other words, as the number of trees in the forest increase, both train and validation accuracies increase. Also, notice the accuracy actually decreases a little bit when the number of tree changes from n=1 to n=2. After that, the accuracies increase rapidly and then improve at a lower rate. Up to n=25, there's no evidence of overfitting yet because the validation accuracy is still increasing. In generally, adding more trees can improve the train and validation accuracy despite potential downside effects in the beginning.

d. When d=9 we print the train and validation accuracies for m=10, 20 and 50 for comparison purposes.

m	n	train accuracy	validation accuracy
10	1	0.775368249	0.721301412
10	2	0.669599018	0.578268877
10	5	0.764934534	0.611418048
10	10	0.890548282	0.759975445
10	25	0.899959083	0.763044813
20	1	0.82794599	0.73664825
20	2	0.796644845	0.719459791
20	5	0.902414075	0.799877225
20	10	0.880319149	0.711479435
20	25	0.985883797	0.918354819
50	1	0.897094926	0.839779006
50	2	0.885638298	0.817679558
50	5	0.964402619	0.909146716
50	10	0.971358429	0.928176796
50	25	0.991202946	0.94966237







We make a table and a corresponding plot show that how m is related to the train and validation accuracy. We find evidence of positive relationship between m and the train and validation accuracy. In other words, an increase in m leads to an increase in train and validation accuracy.

The number of features m can influence the generalization error in two ways: selecting many features increases the strength of the individual trees whereas reducing the number of features m leads to a lower correlation among the trees increasing the strength of the forest as a whole. Generally speaking, it is not always the case that an increase in m leads to an increase in the train and validation accuracy. But it is the case in most situations.

Part -3

Here we implement Boosting algorithm to the weak learner DT so that we can create a stronger learner. The details are presented as followed.

a. We implement the AdaBoost as required
   Please run the code and be a little patient.

b. run the code for details

c. The computed table is given below

m	train accuracy	validation accuracy
1	0.980360065	0.918968692
5	0.9801657673	0.90300798
10	0.9801657673	0.914238953
20	0.9801657673	0.901206037

d. When we increase the parameter L, both the training and validation accuracy are increasing. The training and validation accuracy are much higher than simply applying the weak learner DT. We can conclude that the Adaboost turns a weak learner into a stronger one. The accuracy is not a strictly increasing or decreasing but fluctuation is shown. And validation accuracy might have seen some evidence of overfitting.