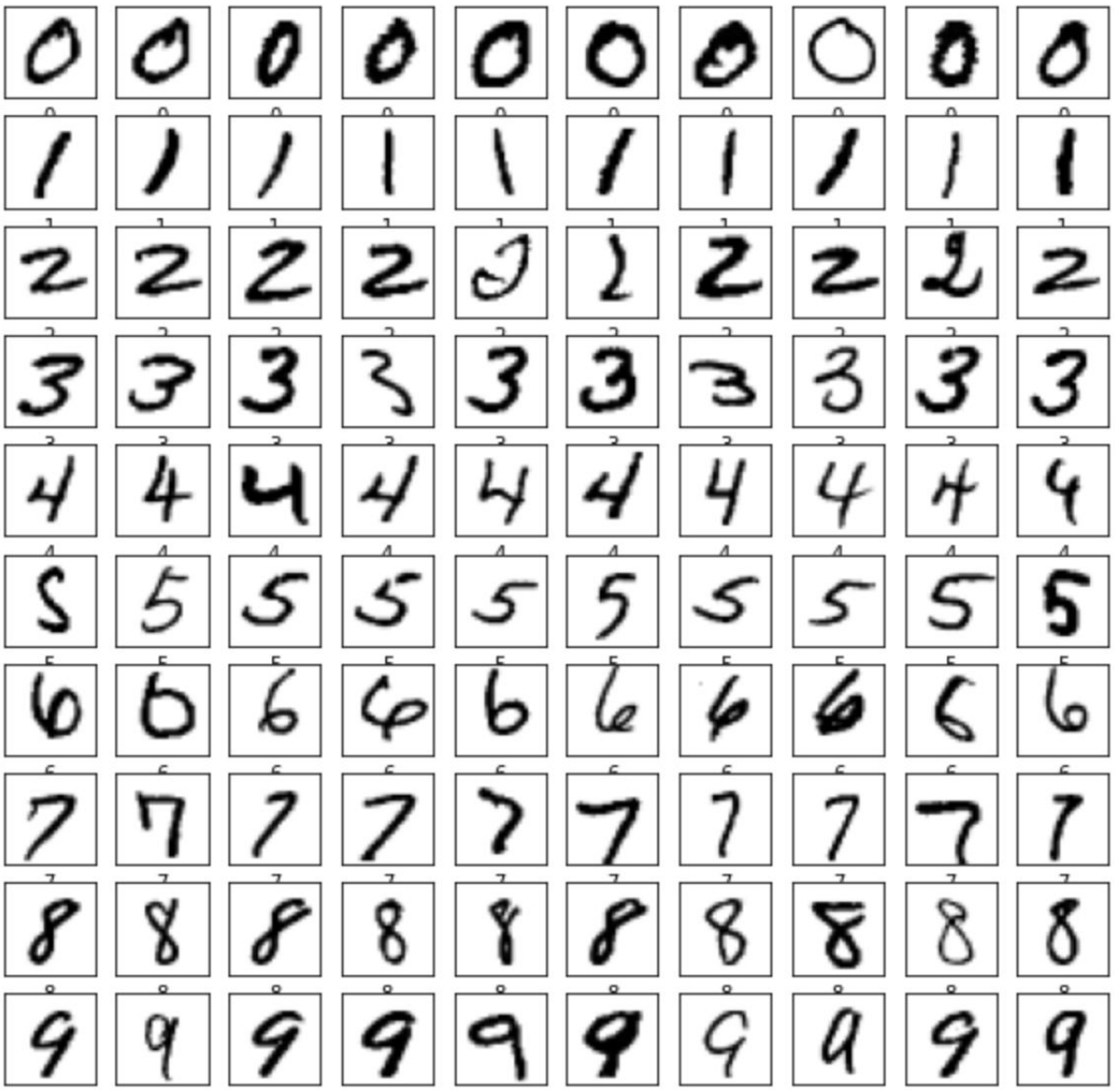
Q1

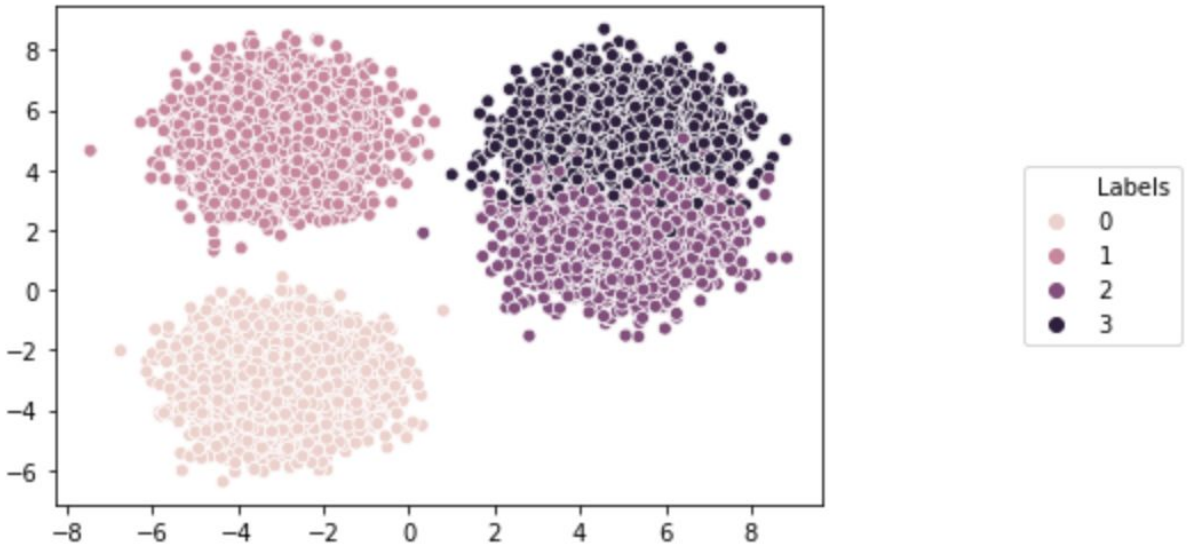
a)



10 samples of each class from ‘dataset 1’ in the form of images.

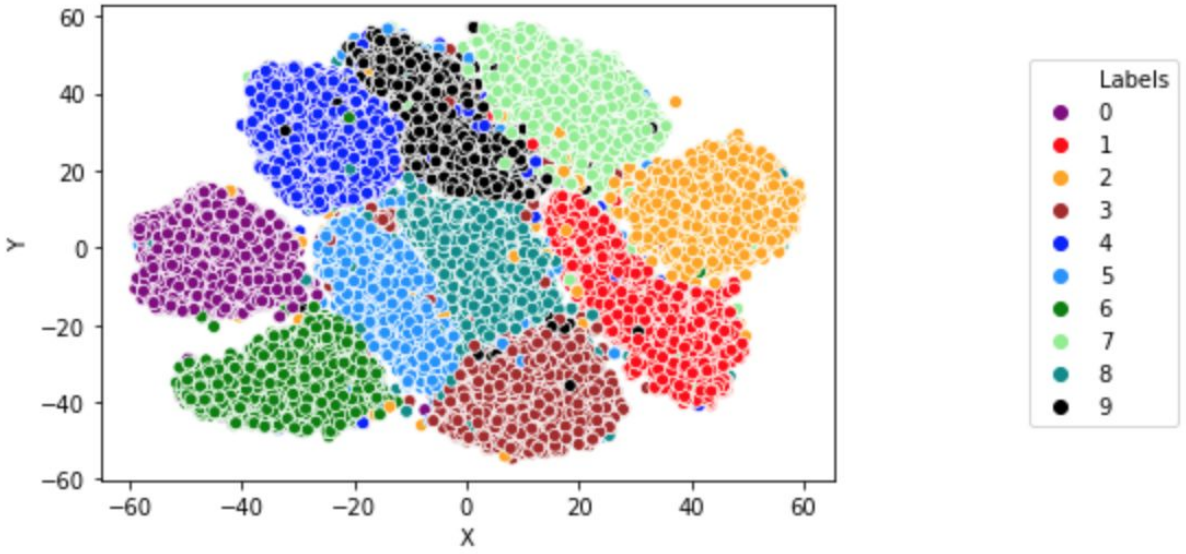
- Here the first thing was to separate the samples and labels and then reshape(50000,28,28,1) the samples and put labels in new dataframes.
- We can know the count by counting the labels i.e 50000.
- Then I have collected the index of samples of same class in the list in sequential manner by using labels using
- Then another loop is uses the index stored in previous step to plot the samples.
- Then I plotted the image using the index in the previous list in a plot of 10x10.

b)



- Data was given in mat format. So I have converted it to Data frame by some data wrangling and using matplot drew the plot.
- Label=0 has both axes less than 0.
- Label=1 have one data point greater than 1 other less than 0
- Label=2 and Label=3 are separated but are almost together in halves.
- Label=3 has almost all both the data point positive.

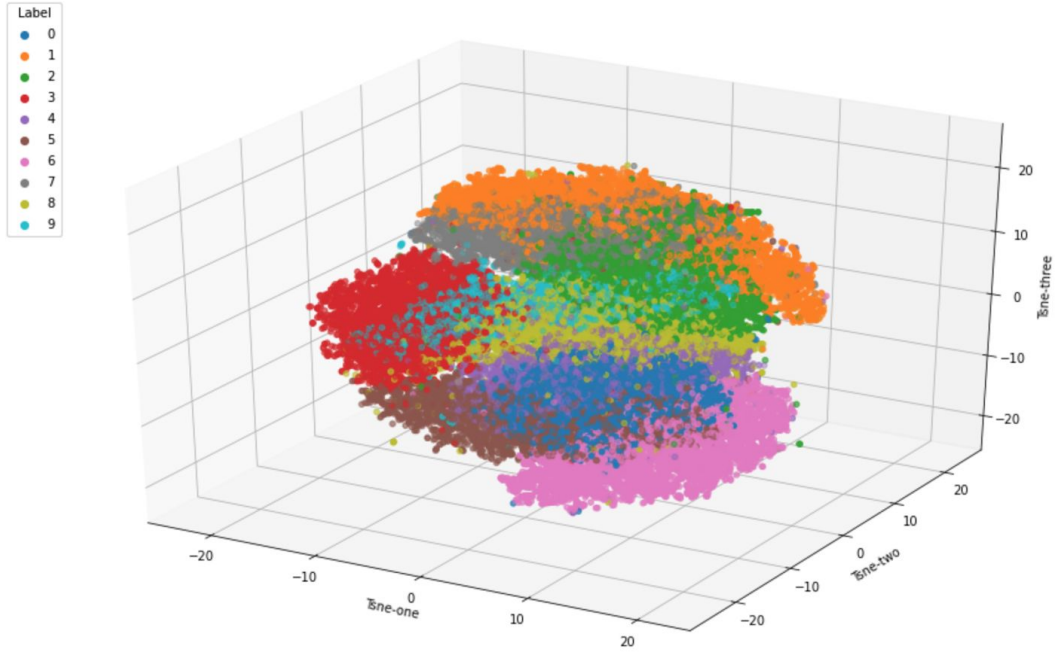
c)



- Here data was given in mat format. So I have separated them into Samples and labels into different data frames.
- Then resize the samples with (50000,28*28).

- Then applied tsne on samples with n_components=2 for 2d conversion
- Then converted the results and labels into dataframes i.e X,Y and labels
- Then using a seaborn scatter plot drew the graph.
- There appears to be 10 classes.
- Most of the points are very much separated according to classes and appear very neat.
- There are very few outliers for every label.
- Labels are separated and have very elegant class separation.
- Most of the labels are closer and not very far from each other.

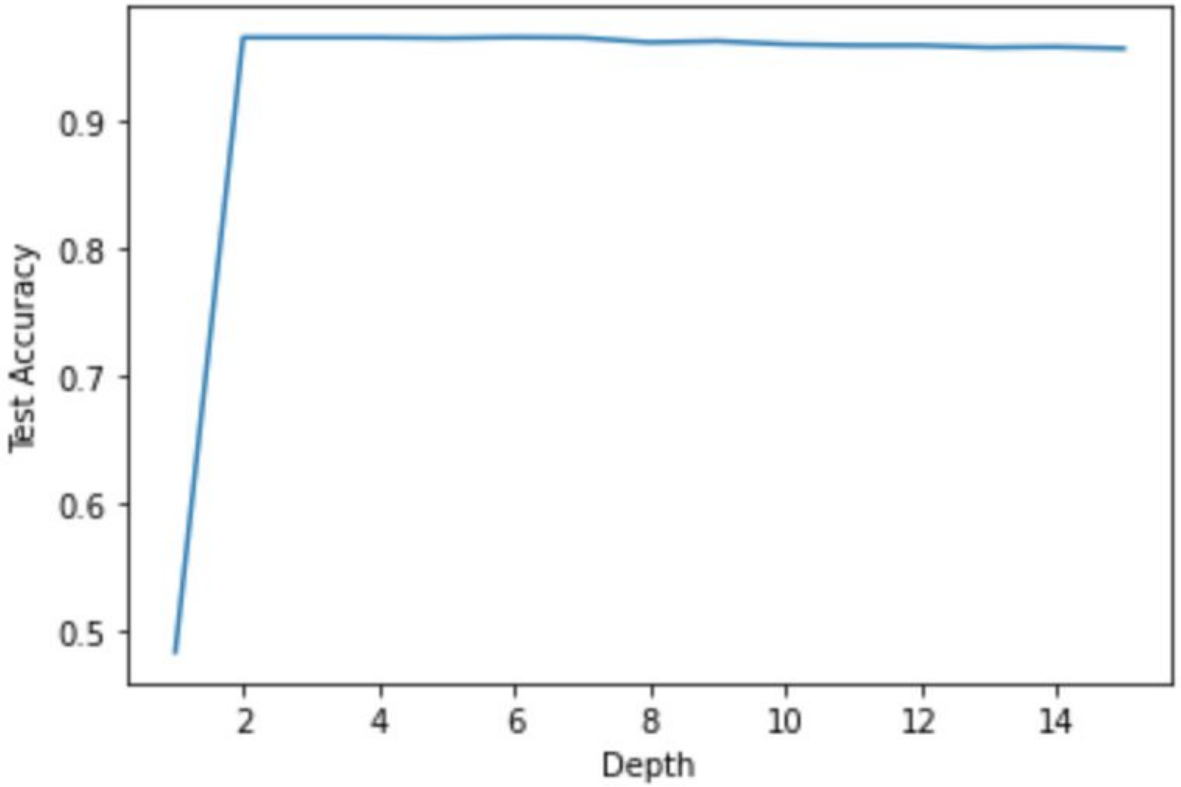
d)



- Here data was given in mat format. So I have separated them into Samples and labels into different data frames.
- Then resize the samples with (50000,28*28).
- Then applied tsne on samples with n_components=3 for 3d conversion
- Then converted the results and labels into dataframes i.e X,Y,Z and labels
- Then using scatter plot drew the graph in 3d.
- There appears to be 10 classes.
- The important distinction is that it has between plotted over 3 different axis and is in 3d plane. This grpg does not give lot of idea about the dataset.
- There are very few outliers for every label.
- They are not very neatly separated like it was in 1c.
- Some labels like 0,2,8 are pretty messed up.

Q2

a)



- Here dataset was in mat format. So I have loaded the dataset using loadmat and then with little data wrangling converted into a data frame.
- Split the dataset into train set and test set by 70:30 ratio.
- Then separated the features and target label into different dataframe i.e X_train,Y_train,X_test,Y_test.
- Calculated the test accuracy for depth=1 to depth=15 using loop. Used my own implementation where I account if Y_test and ypred are equal and evaluate equals_count/total.
- The test accuracy is decreasing slightly with the depth. The highest accuracy is found at depth =2,3,4.
- Yes it is very much consistent with 1b part as it divides the graph into sub divisions accurately starting from 2 to 4. Further it cannot make it better as there are only 4 separate labels which are almost together. So 4 separation i.e depth=4 should be max.

b)

	Depth	Test Accuracy	Train Accuracy
0	1	0.492667	0.503143
1	2	0.966167	0.966786
2	3	0.966167	0.966786
3	4	0.966167	0.966857
4	5	0.965000	0.967714
5	6	0.963000	0.968643
6	7	0.963833	0.969714
7	8	0.961667	0.971857
8	9	0.961000	0.973071
9	10	0.957500	0.974643
10	11	0.959000	0.976429
11	12	0.957500	0.979000
12	13	0.955333	0.981643
13	14	0.955500	0.983071
14	15	0.952500	0.986214

- Here dataset was in mat format. So I have loaded the dataset using loadmat and then with little data wrangling converted into a data frame.
- Split the dataset into train set and test set by 70:30 ratio.
- Then separated the features and target label into different dataframe i.e X_train,Y_train,X_test,Y_test.
- Calculated the test accuracy for depth=1 to depth=15 using loop. Used my own implementation where I account if Y_test and ytestpred are equal and evaluate equals_count/total.
- Calculated the test accuracy for depth=1 to depth=15 using loop. Used my own implementation where I account if Y_train and ytrainpred are equal and evaluate equals_count/total.
- depth=1 train accuracy <=50. Model appears to be underfit
- depth=2 model appears to be fit as train accuracy and test accuracy almost equal.
- depth=3 model appears to be fit as train accuracy and test accuracy almost equal.
- depth =4,5,6,7,8 model appears to be fit as train accuracy and test accuracy almost equal.
- Depth =9,10,12,13,14,15 model is good but training accuracy is slightly more than testing accuracy. So there is little overfitting.

c)

	Depth	Test Accuracy	Train Accuracy
0	1	0.497000	0.501286
1	2	0.964500	0.967571
2	3	0.964500	0.967571
3	4	0.964500	0.967571
4	5	0.963667	0.968071
5	6	0.962833	0.969643
6	7	0.961667	0.971429
7	8	0.959833	0.972929
8	9	0.955667	0.975071
9	10	0.954833	0.976857
10	11	0.954333	0.979429
11	12	0.952667	0.981571
12	13	0.951167	0.984071
13	14	0.952333	0.986429
14	15	0.952000	0.988357

- Similar to above implementation of 2b), Just accuracy is calculated using library function.
- There is not much difference between my implementation and inbuilt function of accuracy score. Only a slight difference that is almost negligible.

Q3

a)

```
clf = DecisionTreeClassifier(criterion="entropy")
clf = clf.fit(X_train,Y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
```

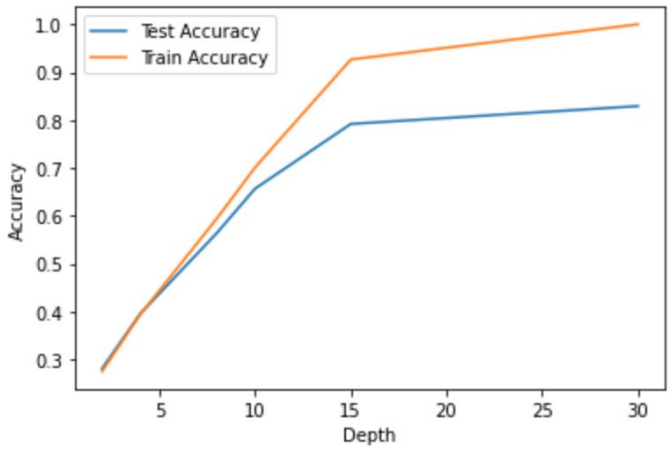
```
Accuracy: 0.8516828294352539
```

```
clf = DecisionTreeClassifier(criterion="gini")
clf = clf.fit(X_train,Y_train)
y_pred = clf.predict(X_test)
print("Accuracy:",metrics.accuracy_score(Y_test, y_pred))
```

```
Accuracy: 0.8470051340559042
```

- We read the data from the csv file then it goes into data frame.
- There are null value in column name ‘pw2.5’ . So I set it to zeros.
- There are string value in cbwd. So I have use encode to encode into four different classes i.e 'NW', 'cv', 'NE', 'SE' to 1, 3, 0, 2.
- Then split the data into train set and testset in 80:20 ratio.
- Separate the target and features into different data frames i.e X_train,Y_train,Y_test,Y_train.
- Then use decision tree to predict and select criterion = ‘entropy’
- Then use decision tree to predict and select criterion = ‘gini’

b)



- We read the data from the csv file then it goes into data frame.
- There are null value in column name ‘pw2.5’ . So I set it to zeros.
- There are string value in cbwd. So I have use encode to encode into four different classes i.e 'NW', 'cv', 'NE', 'SE' to 1, 3, 0, 2.
- Then split the data into train set and testset in 80:20 ratio.
- Separate the target and features into different data frames i.e X_train,Y_train,Y_test,Y_train.
- Made decision tree for depths = 2,4,8,10,15,20 and stored their train accuracy and test accuracy into list for every depth.
- Then convert the list into dataframe.
- Then drew the plot using matplot

c)

```
[53]: pred2 = pred2.loc[0, : ]

[54]: metrics.accuracy_score(Y_test, pred2)

[54]: 0.35322304620650313
```

- We read the data from the csv file then it goes into data frame.
- There are null value in column name ‘pw2.5’ . So I set it to zeros.
- There are string value in cbwd. So I have use encode to encode into four different classes i.e 'NW', 'cv', 'NE', 'SE' to 1, 3, 0, 2.
- Then split the data into train set and testset in 80:20 ratio.
- Now make 100 decision tree with max_depth=3 for different 50% of shuffled data from training data.
- Then separate them into features and target variable i.e X_train,X_test,Y_train,Y_test and X_train1,X_test1,Y_train1,Y_test1 (50%trainset).
- Then we make prediction accordingly on test data and store the predicted result.
- Then mode of predicted result is taken and test accuracy is calculated based on that.
- In regards to 3a and 3b, for 3a no max_depth was set so it will expand as long as possible. Accuracy is 0.84 approx for entropy and gini
  For 3b in max_depth=4 it was around 0.39 so it is almost similar to 0.35 for depth=3.

d)

```
Tree:50 Depth:2   Train Accuracy:0.273881   Test Accuracy :0.281803
Tree:100 Depth:2   Train Accuracy:0.273881   Test Accuracy :0.281803
Tree:150 Depth:2   Train Accuracy:0.273881   Test Accuracy :0.281803
Tree:200 Depth:2   Train Accuracy:0.273881   Test Accuracy :0.281803
Tree:300 Depth:2   Train Accuracy:0.273881   Test Accuracy :0.281803


Tree:50 Depth:4   Train Accuracy:0.400896   Test Accuracy :0.394980
Tree:100 Depth:4   Train Accuracy:0.399926   Test Accuracy :0.394067
Tree:150 Depth:4   Train Accuracy:0.398471   Test Accuracy :0.392470
Tree:200 Depth:4   Train Accuracy:0.399099   Test Accuracy :0.393269
Tree:300 Depth:4   Train Accuracy:0.401865   Test Accuracy :0.397946


Tree:50 Depth:8   Train Accuracy:0.650931   Test Accuracy :0.622818
Tree:100 Depth:8   Train Accuracy:0.653242   Test Accuracy :0.625442
Tree:150 Depth:8   Train Accuracy:0.655894   Test Accuracy :0.625898
Tree:200 Depth:8   Train Accuracy:0.657862   Test Accuracy :0.630918
Tree:300 Depth:8   Train Accuracy:0.657948   Test Accuracy :0.629321

Tree:50 Depth:10   Train Accuracy:0.797570   Test Accuracy :0.748659
Tree:100 Depth:10   Train Accuracy:0.803189   Test Accuracy :0.751055
Tree:150 Depth:10   Train Accuracy:0.801449   Test Accuracy :0.754022
Tree:200 Depth:10   Train Accuracy:0.804986   Test Accuracy :0.753908
Tree:300 Depth:10   Train Accuracy:0.803959   Test Accuracy :0.755847


Tree:50 Depth:15   Train Accuracy:0.984170   Test Accuracy :0.902339
Tree:100 Depth:15   Train Accuracy:0.987364   Test Accuracy :0.908842
Tree:150 Depth:15   Train Accuracy:0.987935   Test Accuracy :0.908956
Tree:200 Depth:15   Train Accuracy:0.987735   Test Accuracy :0.909412
Tree:300 Depth:15   Train Accuracy:0.988847   Test Accuracy :0.911124


Tree:50 Depth:20   Train Accuracy:0.998973   Test Accuracy :0.920707
Tree:100 Depth:20   Train Accuracy:0.999544   Test Accuracy :0.926412
Tree:150 Depth:20   Train Accuracy:0.999715   Test Accuracy :0.928237
Tree:200 Depth:20   Train Accuracy:0.999743   Test Accuracy :0.930291
Tree:300 Depth:20   Train Accuracy:0.999829   Test Accuracy :0.928580
```

- Preprocessing steps are Similar to 3c
- Then I have implemented 3 nested loops where for each depth[2, 4, 8, 10, 15, 20] different trees[50,100,150,200,300] are made.
- Then separate them into features and target variable i.e X_train,X_test,Y_train,Y_test and X_train1,X_test1,Y_train1,Y_test1 (50%trainset).
- Then we make predictions on test data accordingly and store the predicted result.
- Then we make predictions on train data accordingly and store the predicted result.
- Then mode of prediction on test data is taken and test accuracy is calculated based on that.
- Mode of predicted on train data is taken and train accuracy is calculated based on that.
- This occurs for each of the number of trees and depth combination.
- As depth of the tree increases the train accuracy and test accuracy increases.
- Train Accuracy is almost 99.9% for depth= 20 and Test Accuracy is 92%.
- As the depth increases and the number of trees increases the accuracy improves.
- The effect of depth on the model is significant.It affects the accuracy by very significant margin.