# ML (PG) ASSIGNMENT - 1
## REPORT

Student Name: Drishti De
Degree: M.Tech
Roll No.: MT20075

CONTENTS

## 1. Approach

The approach used in this assignment is to analyze the type of data that is required to be dealt with in each of the questions. Based on the type of data, data preprocessing is done and the data is stored in final data-frame format using Pandas. These data frames obtained are then analyzed based on the instructions given in the questions. The main tasks involved in this assignment involve but may not be limited to the following:
i. Data pre-processing
ii. Data analysis
iii. Model training
iv. Visualizing the data or the results obtained after model training and evaluation.

The different datasets are uploaded into the coding interface (google colab) using Google Drive mounted into the colab interface.

## 2. Data Preprocessing

### 2.1 Dataset for Question 1 :

The data that Question 1 deals with mostly is the data stored in the MATLAB file called “dataset_1.mat”. This is a 3D data where the sample dimensions are 3D and the labels have dimensions of 2D. The dimensions of the data are (50000 x 28 x 28).

Here the data is loaded with the help of scipy.io.loadmat. This data is then reshaped to 2D format with sample data (50000 x 28 x 28) converted into (50000 rows and 784 columns in pandas data frame) and the labels are also converted to 1D data and attached to the data frame. This final data frame is then analyzed as per the instructions in question 1.

### 2.2 Dataset for Question 2 :

The data that Question 2 deals with is stored in the MATLAB file called “dataset_2.mat”. It is a 2-Dimensional data and therefore need not be reshaped to convert them into a pandas data frame. The labels are in 2D dimension therefore it is reshaped before including in the dataframe for data analysis. After converting them into a data frame, the shape of the data frame is 20000 rows and 3 columns (including the labels column).

### 2.3 Dataset for Question 3 :

The data (Beijing PM2.5 Data Data Set) used for this question is taken from UCI Machine Learning repository. The number of instances in this data are 43824 and the number of attributes are 13. This data is loaded into google colab using the csv file obtained from the dataset repository.

The data preprocessing done in this data is mainly handling the null values (by replacing it with the man of the attribute concerned) and removing the attributes that have no role to play in its analysis and model building. These attributes are: No, year, day, and hour. The target variable here which is used for model training and evaluation is the month attribute.

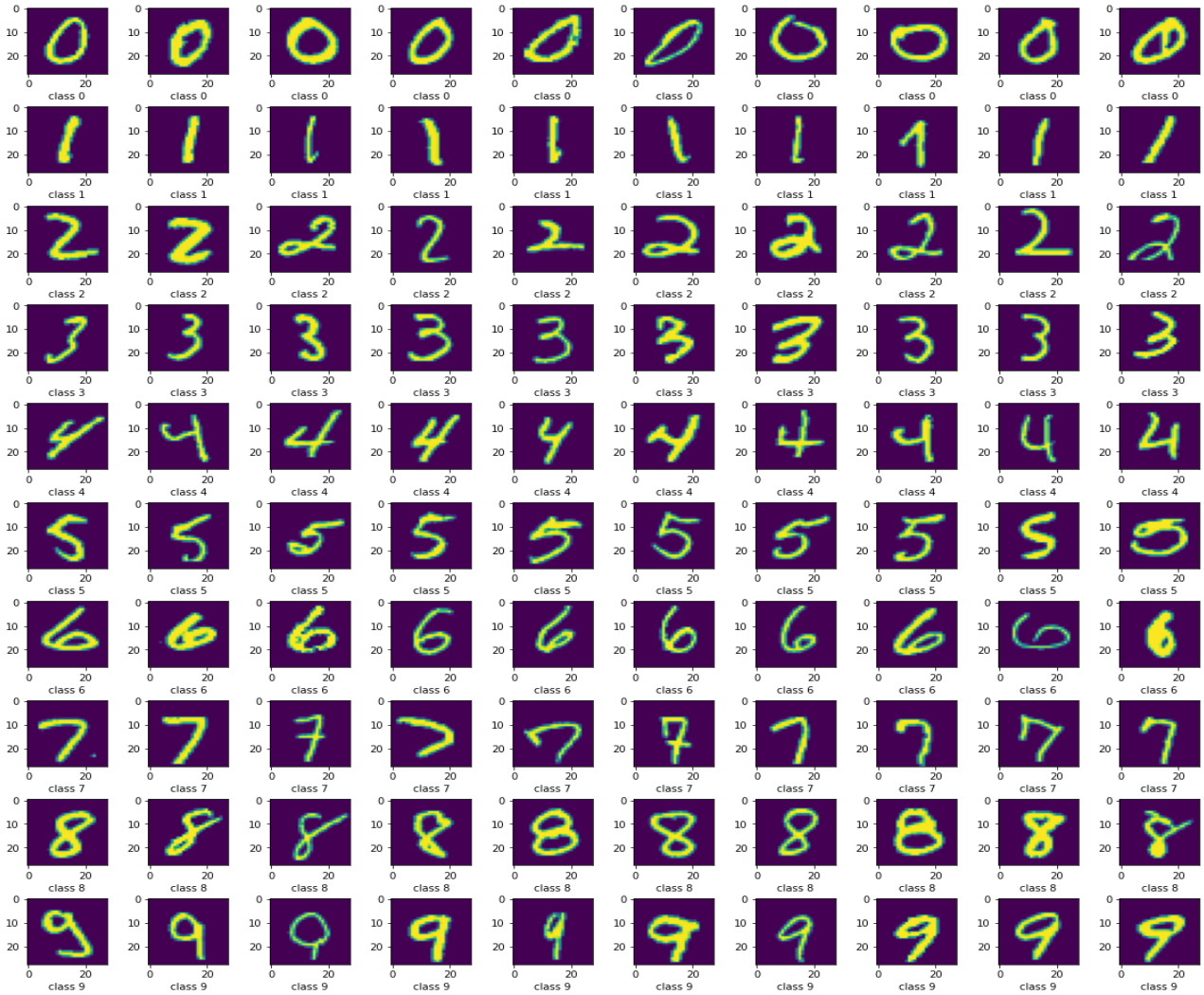
## 3. Analysis and Assumptions

This section of the report mainly focuses on the analysis results and visualizations obtained from the data or after model training and evaluation as per different the instructions given in each question.

### 3.1 For results in Question 1 :

**Part (a):**

Here, as per instruction, the visualization obtained after obtaining 10 samples from each class (class labels are given below each image) in the form of image (using the matplotlib pyplot imshow() plot for visualizing images) is as shown below:

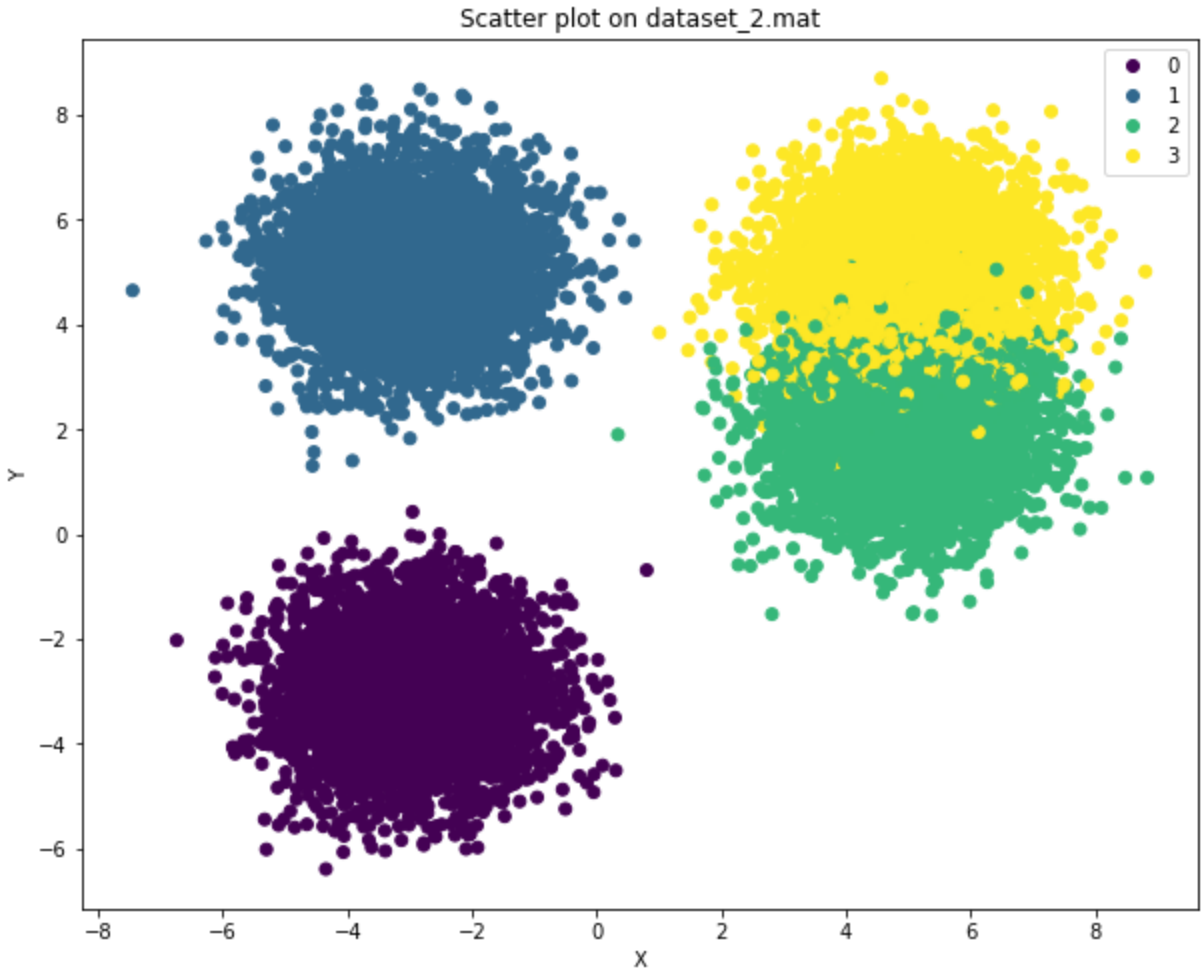


From the above plot we can have the following assumptions:

1. The information regarding the different images (based on labels) are present in the 28 x 28 grid (reshaped into 784 columns in the data frame).
2. These 10 samples from each of the 10 given class labels show image data for different digits (i.e., 0-9) in 10 different handwritings. This dataset can be used as a part of handwriting recognition problems in machine learning.

**Part (b):**

The visualization of the data (dataset_2) in the form of scatter plot can be shown below:

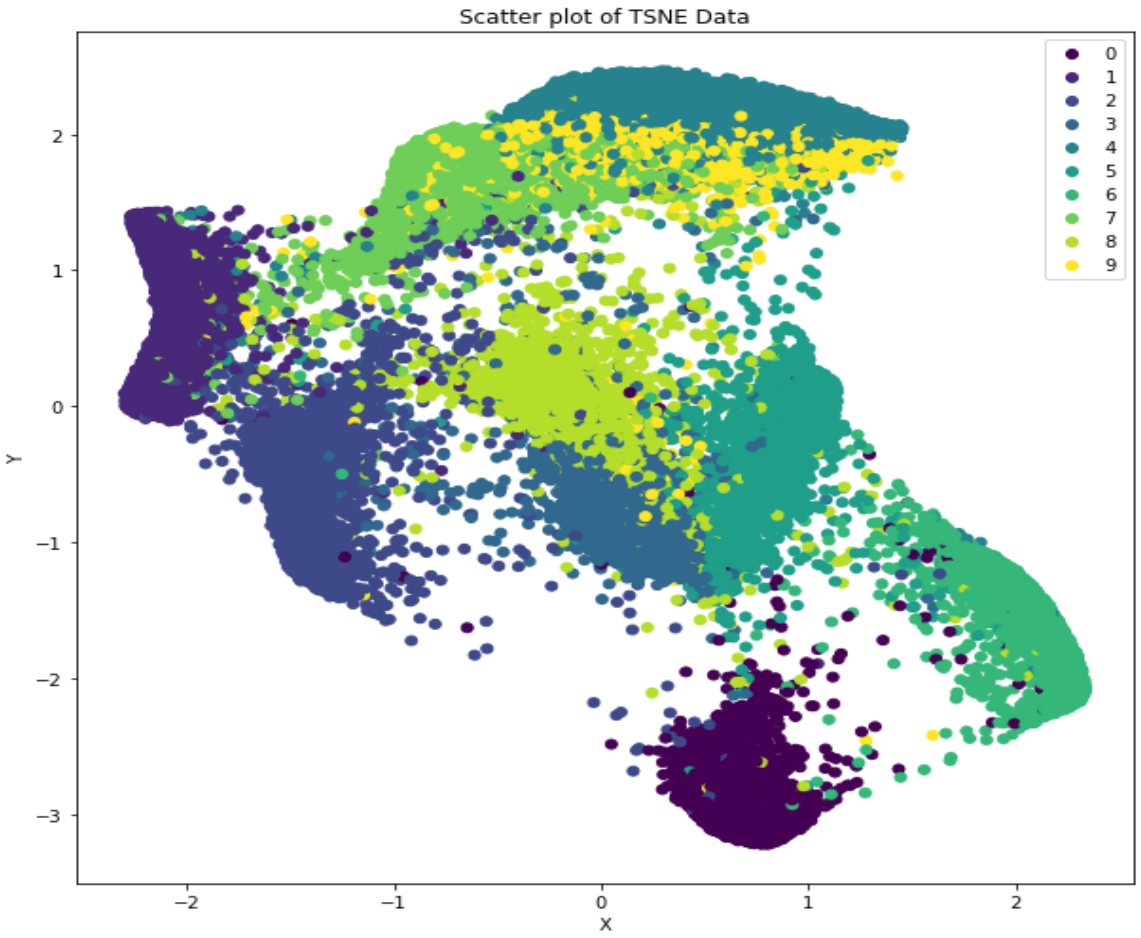


The assumptions that we can infer from the above visualization are:

1. The four different clusters obtained above denote the 4 different classes/labels in the dataset.
2. The clusters for classes 2 and 3 are close to each other and therefore may share some similarity based on the behaviour with respect to the different attributes.
3. Some good clustering models with techniques to find the optimum number of clusters for best results in terms of reduced costs can be helpful in order to differentiate clusters 2 and 3 as two separate clusters instead of one cluster.

**Part (c):**

Using t-SNE to reduce the 'dataset 1' to 2 dimensions (using n_components = 2) and visualizing the scatter plot as given below:

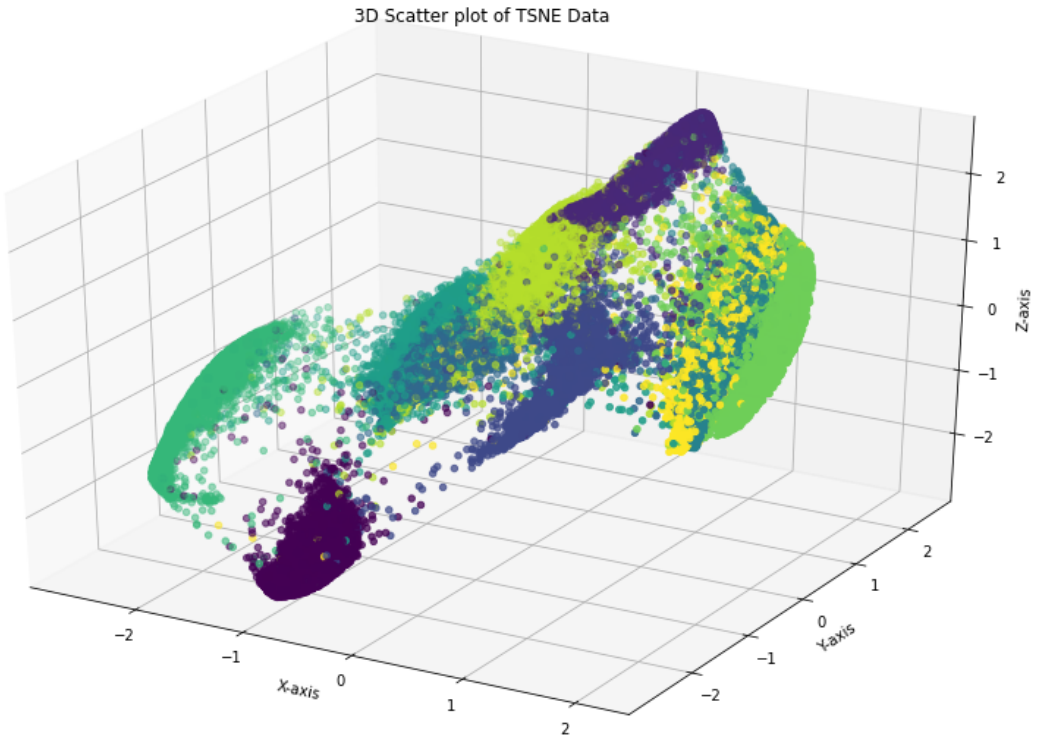



This plot is obtained using perplexity of 40. Increasing the perplexity is seen to bring the clusters closer together for better visualization of data points.

From the above plot we see that all the class labels are forming different clusters with some data points behaving as outliers as they are plotted far away from their clusters. This outlier behaviour can be kept in check if the t-SNE model is trained with a higher perplexity of around 60. But not all clusters can be seen for visualizations based on higher perplexities.

**Part (d):**

For this part of Question 1, the t-SNE is trained for n_components = 3 (to visualize 3D data) using scatter plot as given below:




This data visualization is obtained after training the t-SNE model on dataset_1 for a perplexity of 30. This is done to show different clusters formed by different data points based on their class labels.

This plot is useful in order to understand where the data points lie in the 3D space and therefore it tells a lot more about the behaviour of the data based on their different classifications than that of the 2D data seen previously.

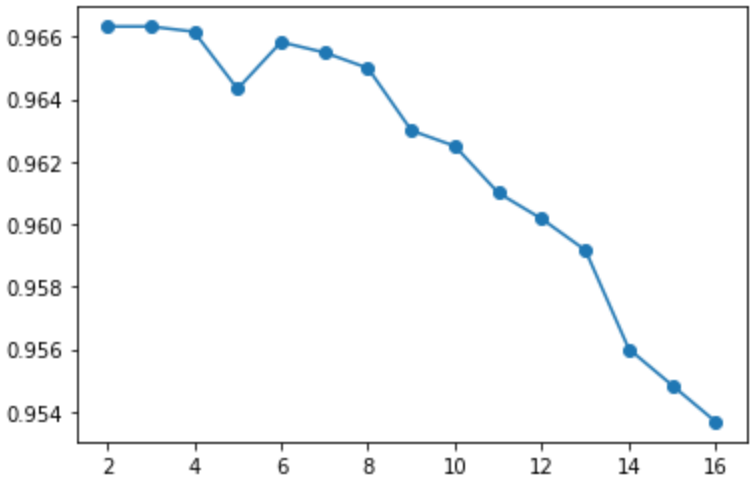
**3.2 For results in Question 2 :**

Taking 70% of the data for training and the rest for testing using a function to split the data in order to train and test the decision tree classifier.

**Part (a):**

In order to perform grid-search technique on the decision tree classifier, we use a for loop in order to take 15 depths (from depth 2 to depth 16) and then pass on those depth values as a hyperparameter into the classifier to see how the depth measure affects the model  model accuracy.

This can be shown with the help of the following plot for depth vs testing accuracy.




The above plot gives us the following information:

1. The optimum level of accuracy was seen for a depth of 2 where the accuracy was 96.633% which remained consistent till depth = 3, which is quite consistent with the class label based visualization done in Question 1 part (b).
2. Even though the decrease is only from 96.633% to 95.366%, the graph shows a decrease from depth 4 to depth 16.
3. A sudden drop is seen for a depth of value 5.
4. For depths 6 to 16, the decrease in accuracy was somewhat consistent.

Assumptions:

Since the classifier also uses balanced class weight as a hyperparameter, along with default gini index criteria therefore these values may have a negative impact with the increase in the depth values.

**Part (b):**

A table representing the different training and testing accuracies for the decision tree classifier on different tree depths (from 2 to 16) as a hyperparameter are shown below:

| | Depth | Validation_Accuracy | Training_accuracy |
|---|---|---|---|
| **0** | 2.0 | 0.966333 | 0.966929 |
| **1** | 3.0 | 0.966333 | 0.966929 |
| **2** | 4.0 | 0.966167 | 0.967000 |
| **3** | 5.0 | 0.964333 | 0.967929 |
| **4** | 6.0 | 0.965833 | 0.969357 |
| **5** | 7.0 | 0.965500 | 0.970357 |
| **6** | 8.0 | 0.965000 | 0.972714 |
| **7** | 9.0 | 0.963000 | 0.974714 |
| **8** | 10.0 | 0.962500 | 0.977429 |
| **9** | 11.0 | 0.961000 | 0.980000 |
| **10** | 12.0 | 0.960167 | 0.981786 |
| **11** | 13.0 | 0.959167 | 0.984143 |
| **12** | 14.0 | 0.956000 | 0.986286 |
| **13** | 15.0 | 0.954833 | 0.988071 |
| **14** | 16.0 | 0.953667 | 0.989714 |

From the definition of overfitting and underfitting and observing the training and testing (validation) accuracies for each depth, we see that the model correctly fits (neither overfits nor underfits) the data and hence is able to rightly predict both the train and the test data.

But for depth value 16, we see that the model is about to reach 100% accuracy for the training data which might result in the decrease in accuracy for the test data hence resulting in overfitting.

**Part (c):**

After replicating part (b) with the sklearn accuracy_function, we have the following table:

| | Depth | Validation_Accuracy | Validation_Accuracy(metrics) | Difference |
|---|---|---|---|---|
| **0** | 2.0 | 0.966333 | 0.966333 | 0.000000 |
| **1** | 3.0 | 0.966333 | 0.966333 | 0.000000 |
| **2** | 4.0 | 0.966167 | 0.966167 | 0.000000 |
| **3** | 5.0 | 0.964333 | 0.964500 | 0.000000 |
| **4** | 6.0 | 0.965833 | 0.966000 | 0.000000 |
| **5** | 7.0 | 0.965500 | 0.965500 | -0.000167 |
| **6** | 8.0 | 0.965000 | 0.965000 | 0.000000 |
| **7** | 9.0 | 0.963000 | 0.962500 | 0.000167 |
| **8** | 10.0 | 0.962500 | 0.962167 | 0.000000 |
| **9** | 11.0 | 0.961000 | 0.961000 | 0.000500 |
| **10** | 12.0 | 0.960167 | 0.960333 | -0.000333 |
| **11** | 13.0 | 0.959167 | 0.958667 | -0.000333 |
| **12** | 14.0 | 0.956000 | 0.956000 | 0.000000 |
| **13** | 15.0 | 0.954833 | 0.954833 | 0.001000 |
| **14** | 16.0 | 0.953667 | 0.954000 | 0.000500 |

According to the table given above, the following information can be inferred:

1. The deviation of the testing (validation) accuracy (calculated manually) from the one derived through instructed sklearn metrics is quite small.
2. The deviations in accuracy values are seen for depths 7, 9, 11, 12, 13, 15 and 16.
3. The total deviation is slightly towards the positive side i.e. 0.001334.
4. This may be due to the fact that the manual calculation takes into account only the rounded-off values while calculating the percentage of true values out of the entire set of values (true values + false values).

**3.3 For results in Question 3 :**

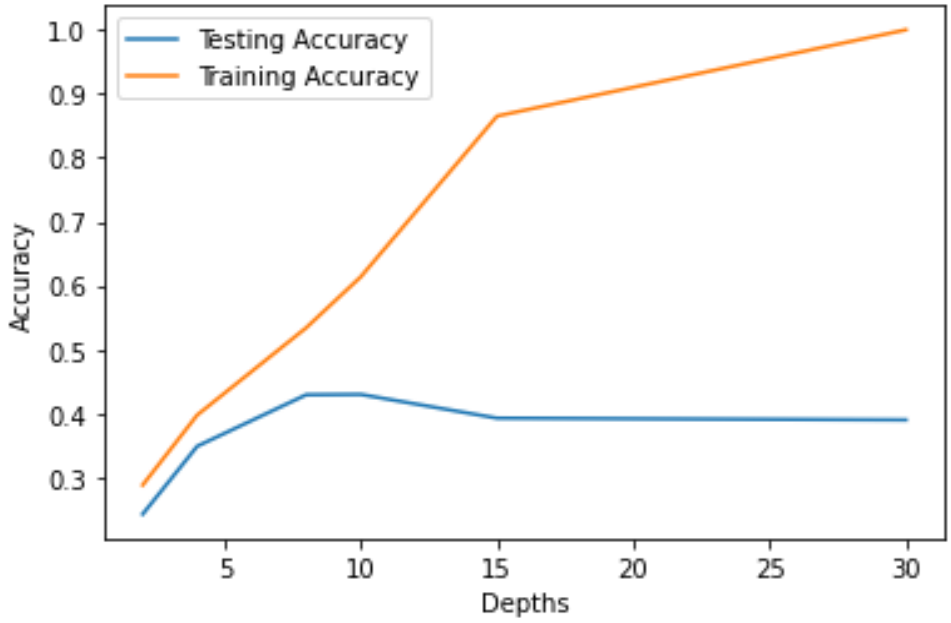
**Part (a):**
Here the decision tree classifier is trained on different criteria namely **gini index** and **entropy**.

**Observation:**

The accuracy achieved through entropy is found to be mostly a bit greater than the one obtained using the gini index criterion.

Hence, the further training using decision tree classifiers are made using the criterion **entropy**.

**Part (b):**
After training the decision trees with different maximum depths [2, 4, 8, 10, 15, 30], the following plot is obtained for the training and testing accuracies with respect to different depths.




Here, from the above plot we see that the training accuracy has an almost steep rise in accuracy than the testing data.

For depth 30, the model overfits the training data since the accuracy for the training data is almost 100% but for the test data, it decreases in the middle and remains to 38.98%

**Part (c):**

In this part of Question 3, Ensemble Method is used to create 100 decision stumps for a depth of 3 and then a majority vote for the 100 different predictions are taken and its accuracy is tested.

Since the classifier gives slightly different predictions each time for 100 decision stumps, therefore the majority vote also deviates a little after running it multiple times. Hence, it also affects the accuracy of the Ensemble method but it gives slightly better accuracy for the model most of the time with the highest being 40% which is greater than 38% from part (a) and part (b).

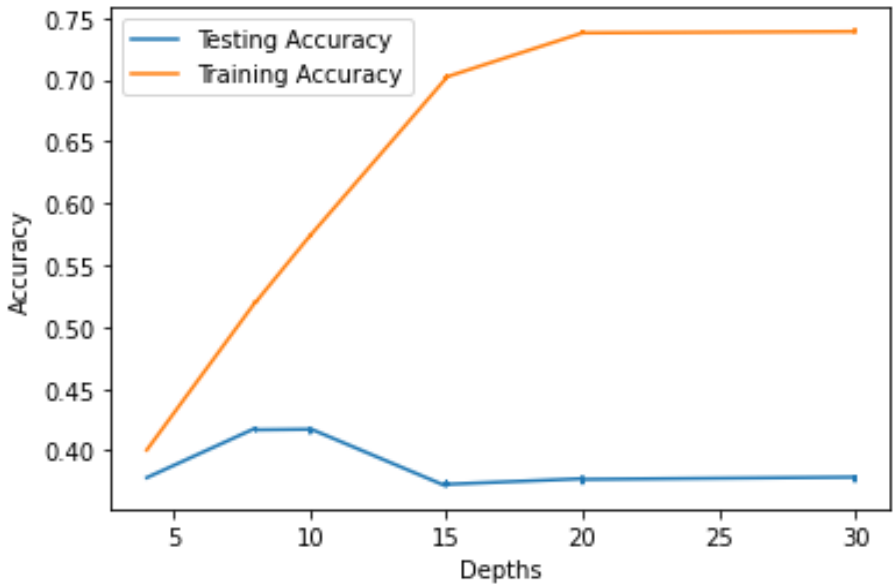
**Part (d):**

Tuning the decision stumps by changing the max-depth [4, 8, 10, 15, 20, best achieved from (b) i.e. 30] and number of trees [75, 100, 150, 300, 500, 1000] we obtain the testing and training accuracies as shown in the table below:

| | Depths | Trees | Testing_Accuracy | Training_Accuracy |
|---|---|---|---|---|
| 0 | 4 | 75 | 0.378095 | 0.400439 |
| 1 | 4 | 100 | 0.378095 | 0.400439 |
| 2 | 4 | 150 | 0.378095 | 0.400439 |
| 3 | 4 | 300 | 0.378095 | 0.400439 |
| 4 | 4 | 500 | 0.378095 | 0.400439 |
| 5 | 4 | 1000 | 0.378095 | 0.400439 |
| 6 | 8 | 75 | 0.418254 | 0.520494 |
| 7 | 8 | 100 | 0.416429 | 0.520152 |
| 8 | 8 | 150 | 0.416771 | 0.520380 |
| 9 | 8 | 300 | 0.416771 | 0.520665 |
| 10 | 8 | 500 | 0.416771 | 0.520494 |
| 11 | 8 | 1000 | 0.416771 | 0.519981 |
| 12 | 10 | 75 | 0.417228 | 0.573947 |
| 13 | 10 | 100 | 0.415859 | 0.573918 |
| 14 | 10 | 150 | 0.416429 | 0.573633 |
| 15 | 10 | 300 | 0.414489 | 0.573633 |
| 16 | 10 | 500 | 0.416201 | 0.573433 |
| 17 | 10 | 1000 | 0.417684 | 0.573776 |
| 18 | 15 | 75 | 0.371249 | 0.701361 |
| 19 | 15 | 100 | 0.371592 | 0.701874 |
| 20 | 15 | 150 | 0.374558 | 0.702416 |
| 21 | 15 | 300 | 0.374102 | 0.702530 |
| 22 | 15 | 500 | 0.372504 | 0.701589 |
| 23 | 15 | 1000 | 0.372732 | 0.702102 |
| 24 | 20 | 75 | 0.377296 | 0.737756 |
| 25 | 20 | 100 | 0.376954 | 0.737899 |
| 26 | 20 | 150 | 0.373759 | 0.737642 |
| 27 | 20 | 300 | 0.378437 | 0.738384 |
| 28 | 20 | 500 | 0.376497 | 0.738555 |
| 29 | 20 | 1000 | 0.376726 | 0.737756 |
| 30 | 30 | 75 | 0.378551 | 0.738869 |
| 31 | 30 | 100 | 0.376269 | 0.739667 |
| 32 | 30 | 150 | 0.378095 | 0.739610 |
| 33 | 30 | 300 | 0.375471 | 0.738897 |
| 34 | 30 | 500 | 0.376726 | 0.738783 |
| 35 | 30 | 1000 | 0.377867 | 0.739724 |

From the above table we see that the testing and training accuracies are not that good.

The graph plot obtained is shown as follows:




The following information can be obtained from the data:

1. The best testing accuracy is achieved for a depth of 8 at with number of trees = 75.
2. At depth 30 (best in part (b)) and number of trees = 100 (i.e. 100 decision stumps), the accuracy is lesser, that is, 37.62%
3. Training accuracy keeps on increasing in arbitrary fashion from 40.04% to 73.97%

Assumptions:

1. The model has a tendency of overfitting the data (seen due to significantly less testing accuracy than that of training accuracy).
2. The training accuracy values may be due to the size of the testing data (original training data) being 2 times larger than that of the training data (50% of the original training data).