

Machine Learning (CS60050) - Assignment 1 Report

Group :

Gangaram Arvind Sudewad- 20CS30017

Chenna Keshava Reddy- 20CS10014

Tasks :

1. Split Dataset C into 80%-20% to form training and testing sets, respectively. Build a Decision Tree Classifier using ID3 algorithm. Train the classifier using Information Gain (IG) measure (no packages to be used for Decision Tree Classifier).
2. Repeat (1) for 10 random splits. Print the best test accuracy and the depth of that tree.
3. Perform reduced error pruning operation over the tree obtained in (2). Plot a graph showing the variation in test accuracy with varying depths. Print the pruned tree obtained in hierarchical fashion with the attributes clearly shown at each level.

Dataset:

The given dataset has the information about the vehicles and the customers so that the person is interested in insurancing the vehicle.

The dataset has the following attributes:

ID	Unique ID for the customer
GENDER	Gender of the customer
AGE	Age of the customer
DRIVING_LICENSE	0: Customer does not have DL, 1: Customer already has DL
REGION_CODE	Unique code for the region of the customer
PREVIOUSLY_INSURED	1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance
VEHICLE_AGE	Age of the Vehicle
VEHICLE_DAMAGE	1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle was damaged in the past.

ANNUAL_PREMIUM	The amount customer needs to pay as a premium in the year
POLICY_SALES_CHANNEL	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
VINTAGE	Number of Days, Customer has been associated with the company
RESPONSE	1 : Customer is interested, 0 : Customer is not interested

The target function is a boolean-valued classification of whether or not a person will take the vehicle insurance or not,
The dataset has 131689 training examples and is provided in the file “Dataset_C.csv” in ‘.csv’ format.

The Decision Tree Algorithm Used

- We have used the ID3 algorithm for constructing the decision tree. However, the standard ID3 algorithm is restricted to attributes that take on a discrete set of values. To incorporate the continuous valued attributes in our dataset, we try to partition the continuous range of values into a discrete set of intervals.
- For each attribute, candidate thresholds are found out by first sorting its unique values and then finding the mid-points of all consecutive pairs of values. All data samples for which attribute value < threshold are put in the left child, and samples for which attribute value \geq thresholds are put in the right child.
- Also note that as compared to the normal decision tree algorithm where we consider an attribute only once in one branch, here we can split on the basis of the same attribute more than once, because we want to be able to create multiple partitions of the range of values taken by an attribute.

Procedure and Results

1. Comparing Impurity Measures

Procedure :

- Split the dataset randomly as an 80/20 split (80% for training and 20% for testing)
- Build the decision tree using the algorithm described above using the Information Gain impurity measures During splitting, we always try to maximize information gain.

```

*****LOADING THE DATA SET*****

*****PRINT THE COMPARITY MEASURES BEFORE SPLITTING*****

Impurity Measure: Information Gain
Train Accuracy: 87.7701%, Test Accuracy: 86.8369%, f1 score: 0.0000

*****PRINT THE ACCURACY AFTER SPLITTING OVER 10 RANDOM SPLITS *****

Split 1 - Test Accuracy: 85.8546%
Split 2 - Test Accuracy: 86.8369%
Split 3 - Test Accuracy: 86.6405%
Split 4 - Test Accuracy: 84.4794%
Split 5 - Test Accuracy: 84.2829%
Split 6 - Test Accuracy: 84.8723%
Split 7 - Test Accuracy: 86.2475%
Split 8 - Test Accuracy: 84.8723%
Split 9 - Test Accuracy: 85.0688%
Split 10 - Test Accuracy: 84.8723%
Average Test Accuracy: 85.4028%
Best Test Accuracy: 86.8369%

*****PRINT THE TEST ACCURACY BY VARYING DEPTHS *****

```

2. Determining Average Accuracy Over 10 Random Splits

Procedure :

- Divide the entire dataset into two parts - 80% training data, and 20% test data.
- Take 1/4th of the training data and keep it aside as the validation set, as it will be used later during pruning. Now our data looks like - 60% training set, 20% validation set and 20% test set.
- Repeat the above steps 10 times, and record the average test accuracy obtained
- Store the decision tree with the best test accuracy, as it will be used later for pruning.

```

*****LOADING THE DATA SET*****

*****PRINT THE COMPARITY MEASURES BEFORE SPLITTING*****

Impurity Measure: Information Gain
Train Accuracy: 87.7701%, Test Accuracy: 86.8369%, f1 score: 0.0000

*****PRINT THE ACCURACY AFTER SPLITTING OVER 10 RANDOM SPLITS *****

Split 1 - Test Accuracy: 85.8546%
Split 2 - Test Accuracy: 86.8369%
Split 3 - Test Accuracy: 86.6405%
Split 4 - Test Accuracy: 84.4794%
Split 5 - Test Accuracy: 84.2829%
Split 6 - Test Accuracy: 84.8723%
Split 7 - Test Accuracy: 86.2475%
Split 8 - Test Accuracy: 84.8723%
Split 9 - Test Accuracy: 85.0688%
Split 10 - Test Accuracy: 84.8723%
Average Test Accuracy: 85.4028%
Best Test Accuracy: 86.8369%

*****PRINT THE TEST ACCURACY BY VARYING DEPTHS *****

```

3. Variation of Depth

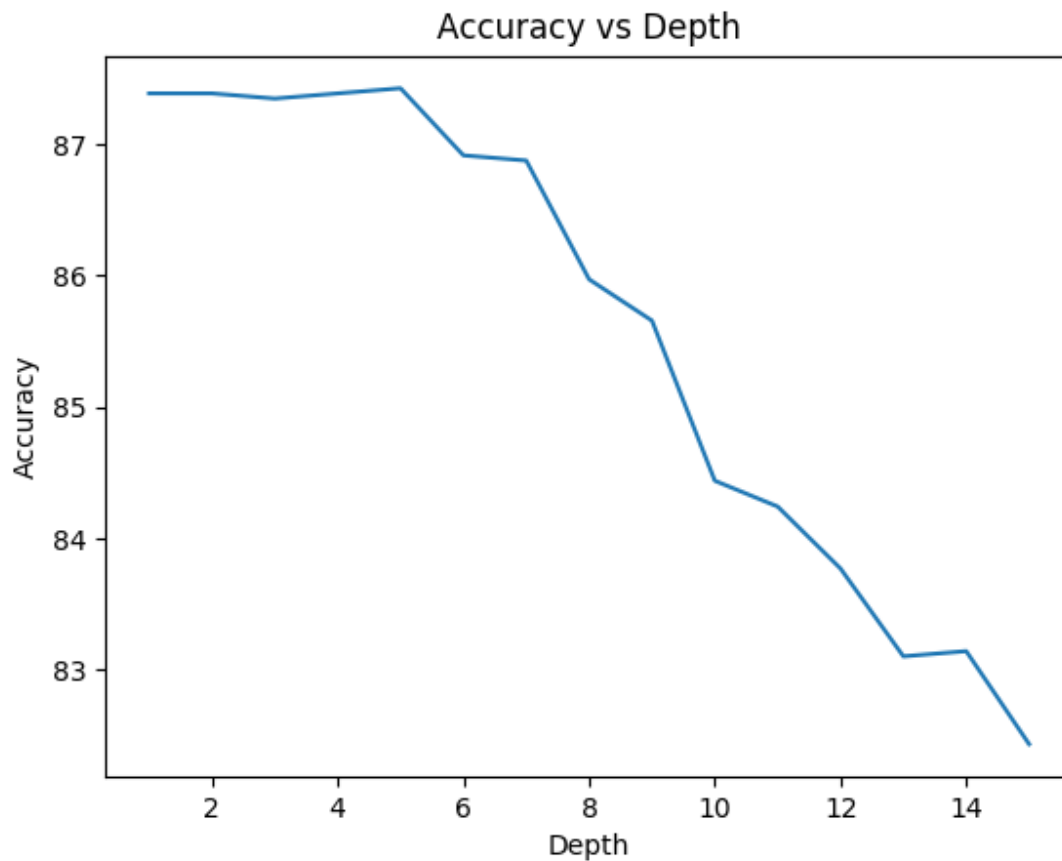
Procedure :

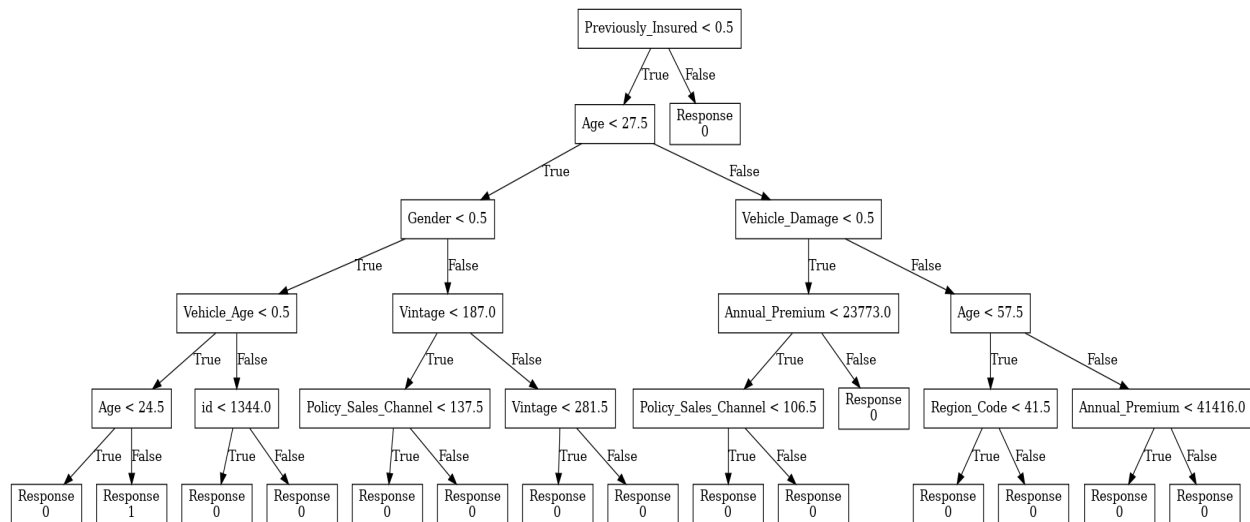
- Vary the maximum depth from 1 to 15 to observe the variation of test accuracy with the maximum depth of the decision tree
- Construct 10 decision trees for each depth using 10 random 80/20 splits, and take the average of the accuracies of each of them to compute the accuracy corresponding to that depth. Record this test accuracy for each depth. The bigger challenge is to observe the variation of test accuracy with the number of nodes in the tree.

If we try to vary the number of nodes over a wide range, then constructing a decision tree for each of them would be computationally very expensive. So, we apply a smart trick here. While

varying the depth itself, for each depth from 1 to 15, we are constructing 10 decision trees for each depth. So, for each of these trees, we count the number of nodes, and store them with the corresponding accuracy. This saves us a lot of time and effort, and we are able to combine the process of variation of depth and variation of nodes in a single function. Also, in this manner, we get a sufficient number of points for the number of nodes in order to plot a graph.

```
*****PRINT THE TEST ACCURACY BY VARYING DEPTHS *****
Depth: 1, Average Accuracy: 87.3870%
Depth: 2, Average Accuracy: 87.3870%
Depth: 3, Average Accuracy: 87.3477%
Depth: 4, Average Accuracy: 87.3870%
Depth: 5, Average Accuracy: 87.4263%
Depth: 6, Average Accuracy: 86.9155%
Depth: 7, Average Accuracy: 86.8762%
Depth: 8, Average Accuracy: 85.9725%
Depth: 9, Average Accuracy: 85.6582%
Depth: 10, Average Accuracy: 84.4401%
Depth: 11, Average Accuracy: 84.2436%
Depth: 12, Average Accuracy: 83.7721%
Depth: 13, Average Accuracy: 83.1041%
Depth: 14, Average Accuracy: 83.1434%
Depth: 15, Average Accuracy: 82.4361%
Best Depth: 5
Accuracy for depth 5: 87.4263%
*****REDUCE ORDER PRUNING OVER TREE FROM (2)*****
Test accuracy after pruning: 87.2299%
```





4. Pruning

Procedure:

- Take the decision tree with best test accuracy from part 2
- Start with the parents of the leaf nodes as leaf nodes cannot be pruned. For every such node, temporarily prune the subtree below, and check the accuracy on the validation set.
- If the accuracy increases, then permanently prune the tree below and make the current node a leaf node by assigning it the label of the majority vote.
- Move up the tree and continue the same process.
- Continue till the accuracy on the validation set increases

We have the decision tree with best test accuracy from part 2, now we apply the pruning operation in this tree. We had partitioned the 80% training data into the 60% grow set which we used to train and build the decision tree, and the 20% validation set. Now we use the validation set. The approach we take is known as reduced-error pruning. The benefit of pruning is that it helps in reducing overfitting significantly.

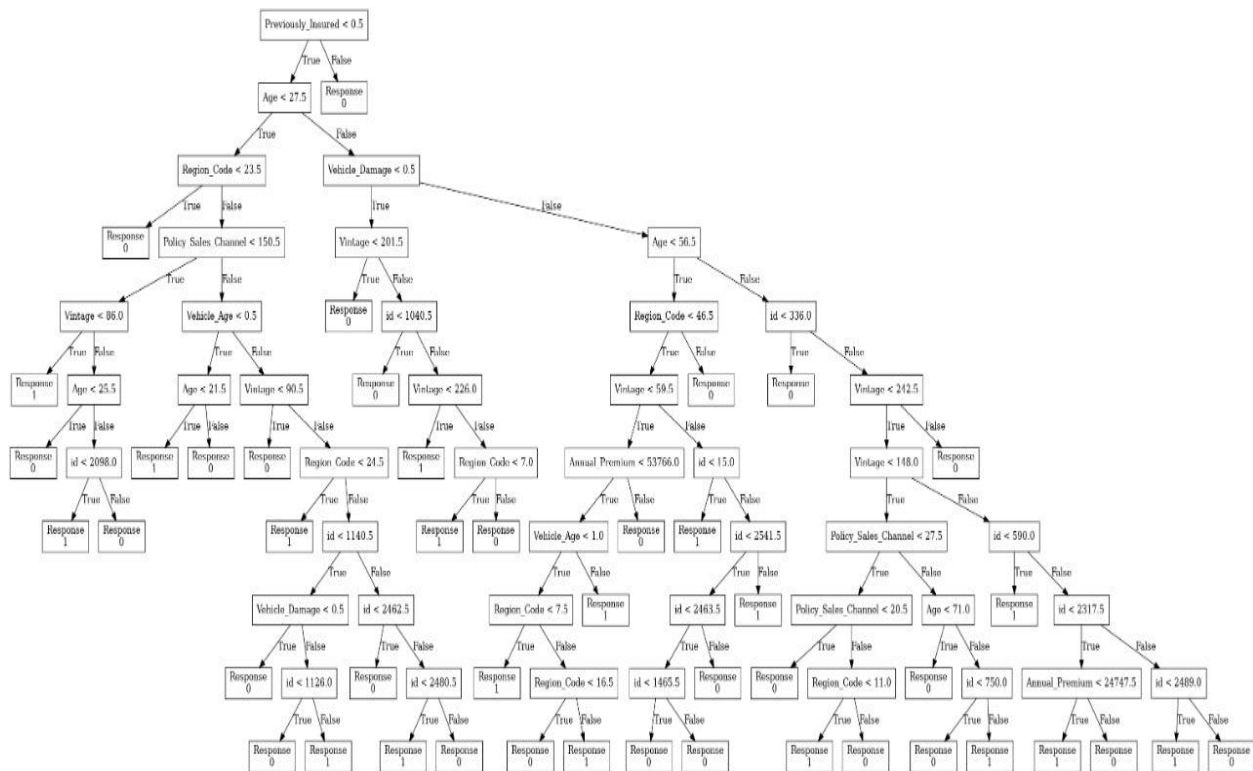
```

Best Test Accuracy: 87.4263%
*****PRINT THE TEST ACCURACY BY VARYING DEPTHS *****
Depth: 1, Average Accuracy: 87.3870%
Depth: 2, Average Accuracy: 87.3870%
Depth: 3, Average Accuracy: 87.3477%
Depth: 4, Average Accuracy: 87.3870%
Depth: 5, Average Accuracy: 87.4263%
Depth: 6, Average Accuracy: 86.9155%
Depth: 7, Average Accuracy: 86.8762%
Depth: 8, Average Accuracy: 85.9725%
Depth: 9, Average Accuracy: 85.6582%
Depth: 10, Average Accuracy: 84.4401%
Depth: 11, Average Accuracy: 84.2436%
Depth: 12, Average Accuracy: 83.7721%
Depth: 13, Average Accuracy: 83.1041%
Depth: 14, Average Accuracy: 83.1434%
Depth: 15, Average Accuracy: 82.4361%
Best Depth: 5
Accuracy for depth 5: 87.4263%
*****REDUCE ORDER PRUNING OVER TREE FROM (2)*****
Test accuracy after pruning: 87.2299%

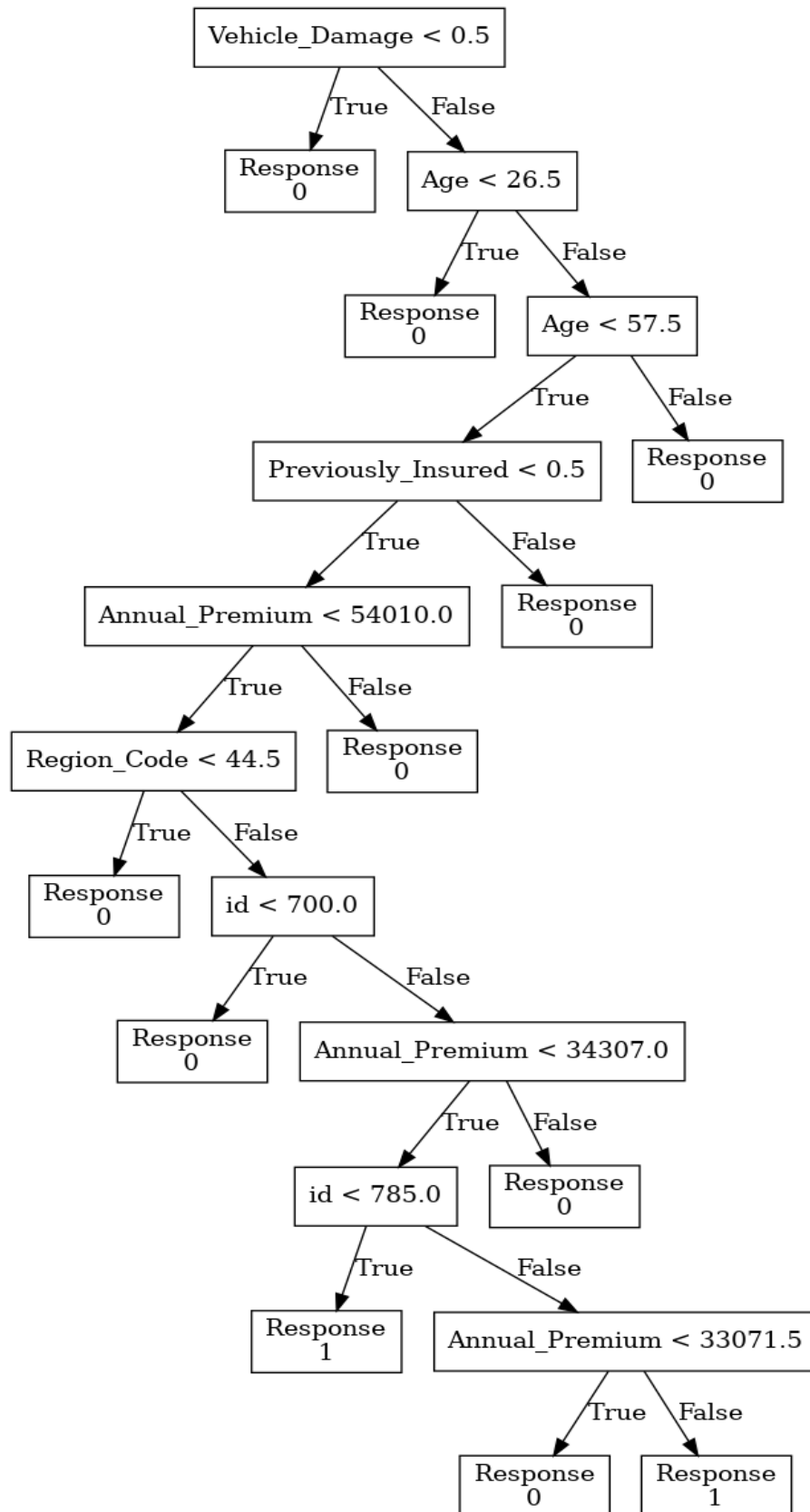
```

	Test Accuracy (in %)
Before Pruning	86.0511
After Pruning	88.0157

Before Pruning



After Pruning



Question -2

TASK:

1. Randomly divide Dataset C into 80% for training and 20% for testing. Encode categorical variables using appropriate encoding method (in-built function allowed).
2. A feature value is considered as an outlier if its value is greater than mean + 3 x standard deviation ($\mu + 3 \times \sigma$). A sample having maximum such outlier features must be dropped. Print the final set of features formed. Normalize the features as required.
3. Train the Naïve Bayes Classifier using 10-fold cross validation (no packages to be used for Naïve Bayes Classifier). Print the final accuracy.
4. Train the Naïve Bayes Classifier using Laplace correction on the same train and test split. Print the final accuracy.

Dataset:

The given dataset has the information about the vehicles and the customers so that the person is interested in insuring the vehicle.

The dataset has the following attributes:

ID	Unique ID for the customer
GENDER	Gender of the customer
AGE	Age of the customer
DRIVING_LICENSE	0: Customer does not have DL, 1: Customer already has DL
REGION_CODE	Unique code for the region of the customer
PREVIOUSLY_INSURED	1 : Customer already has Vehicle Insurance, 0 : Customer doesn't have Vehicle Insurance
VEHICLE_AGE	Age of the Vehicle
VEHICLE_DAMAGE	1 : Customer got his/her vehicle damaged in the past. 0 : Customer didn't get his/her vehicle was damaged in the past.
ANNUAL_PREMIUM	The amount customer needs to pay as a premium in the year
POLICY_SALES_CHANNEL	Anonymized Code for the channel of outreaching to the customer ie. Different Agents, Over Mail, Over Phone, In Person, etc.
VINTAGE	Number of Days, Customer has been associated with the company
RESPONSE	1 : Customer is interested, 0 : Customer is not interested

The target function is a boolean-valued classification of whether or not a person will take the vehicle insurance or not,

The dataset has 131689 training examples and is provided in the file "Dataset_C.csv" in '.csv' format.

The Bayesian (Naïve Bayes) Classifier Used

This article discusses the theory behind the Naive Bayes classifiers and their implementation.

Naive Bayes classifiers are a collection of classification algorithms based on Bayes' Theorem. It is not a single algorithm but a family of algorithms where all of them share a common principle, i.e. every pair of features being classified is independent of each other.

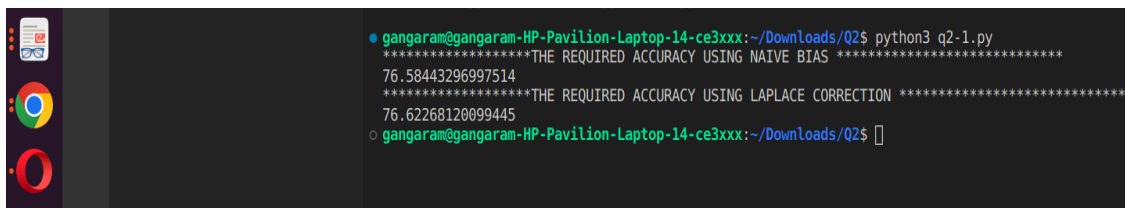
Procedure

We have used the standard Naive Bayes algorithm for designing the classifier in this assignment. The data used for training was provided in .csv format. We have shuffled the data. We considered the split of 80:20 for the Train and Test set. We reported the results with respect to ten-fold cross-validation.

Ten-fold cross-validation

There are a total of ten iterations. Each iteration considers one-tenth of the training set as validation set and rest as the train data. The model is trained on the training set and tested on the validation set. This approach helps to provide a generalization accuracy better, i.e., taking the average of the tenth accuracies above.

Result:

A terminal window screenshot showing the execution of a Python script. The prompt is 'gangaram@gangaram-HP-Pavilion-Laptop-14-ce3xxx:~/Downloads/Q2\$'. The command 'python3 q2-1.py' has been executed. The output displays two accuracy values: 76.58443296997514 for Naive Bias and 76.62268120099445 for Laplace Correction, each preceded by a line of asterisks indicating the required accuracy.

```
● gangaram@gangaram-HP-Pavilion-Laptop-14-ce3xxx:~/Downloads/Q2$ python3 q2-1.py
*****THE REQUIRED ACCURACY USING NAIVE BIAS *****
76.58443296997514
*****THE REQUIRED ACCURACY USING LAPLACE CORRECTION *****
76.62268120099445
○ gangaram@gangaram-HP-Pavilion-Laptop-14-ce3xxx:~/Downloads/Q2$
```