MACHINE LEARNING

ASSIGNMENT 2: DECISION TREE

SARTHAK CHAKRABORTY (16CS30044)

Part 1 (Toy Dataset)

The dataset that has been provided to us is as follows:

a) Training Data

price	maintenance	capacity	airbag	profitable
low	low	2	no	yes
low	med	4	yes	no
low	high	4	no	no
med	med	4	no	no
med	med	4	yes	yes
med	high	2	yes	no
high	med	4	yes	yes
high	high	2	yes	no
high	high	5	yes	yes

b) **Test Data**

price	maintenance	capacity	airbag	profitable
med	high	5	no	yes
low	low	4	no	yes

A) We had to build a Decision Tree that would fit the training data as shown above and required to classify the test data

a) Using **GINI INDEX** of a node as impurity measure

The tree was built using GINI SPLIT at each node.

$$GINI_{split} = \sum_{i=1}^{k} \frac{n_i}{n} GINI(i)$$

$$GINI(t) = 1 - \sum_{j} [p(j \mid t)]^2$$

The built tree is as follows:

```
################################
DECISION TREE trained using Gini Split
################################
Structure:

|maintenance = low : yes
|maintenance = med
        |price = low : no
        |price = med
                |airbag = no : no
                |airbag = yes : yes
        |price = high : yes
|maintenance = high
        |capacity = 2 : no
        |capacity = 4 : no
        |capacity = 5 : yes
```

The value of GINI Index of the root node and the accuracy that the model achieved on the test data is:

```
Gini Index of Root Node: 0.4938

Labels generated on test data:
1. yes
2. yes
Actual Labels:
1. yes
2. yes

Accuracy on Test Data: 1.0
```

b) Using **INFORMATION GAIN** of a node as a measure of impurity

$$Entropy(t) = -\sum_j p(j \mid t) \log_2 p(j \mid t)$$

$$GAIN_{split} = Entropy(p) - \left(\sum_{i=1}^{k} \frac{n_i}{n} Entropy(i) \right)$$

The built tree is as follows:

```
################################
DECISION TREE trained using Information Gain
################################
Structure:

|maintenance = low : yes
|maintenance = med
        |price = low : no
        |price = med
                |airbag = no : no
                |airbag = yes : yes
        |price = high : yes
|maintenance = high
        |capacity = 2 : no
        |capacity = 4 : no
        |capacity = 5 : yes
```

The value of INFORMATION GAIN of the root node and the accuracy that the model achieved on the test data is:

```
Information Gain of Root Node: 0.1861

Labels generated on test data:
1. yes
2. yes
Actual Labels:
1. yes
2. yes

Accuracy on Test Data: 1.0
```

Thus, we see that the data has been correctly classified by the Decision Tree.

B) Decision Tree Implementation of **scikit-learn**

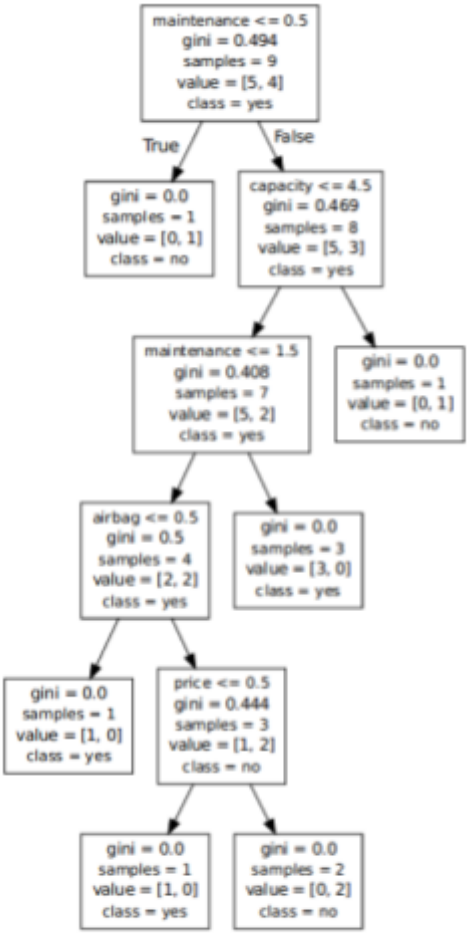
a)Test accuracy and GINI index of the root node (Split using GINI split):

```
DECISION TREE trained using Gini Split
-----------------------------------
Gini Index of Root Node: 0.4938

Labels generated on test data:
1. yes
2. yes
Actual Labels:
1. yes
2. yes

Accuracy on Test Data: 1.0
```

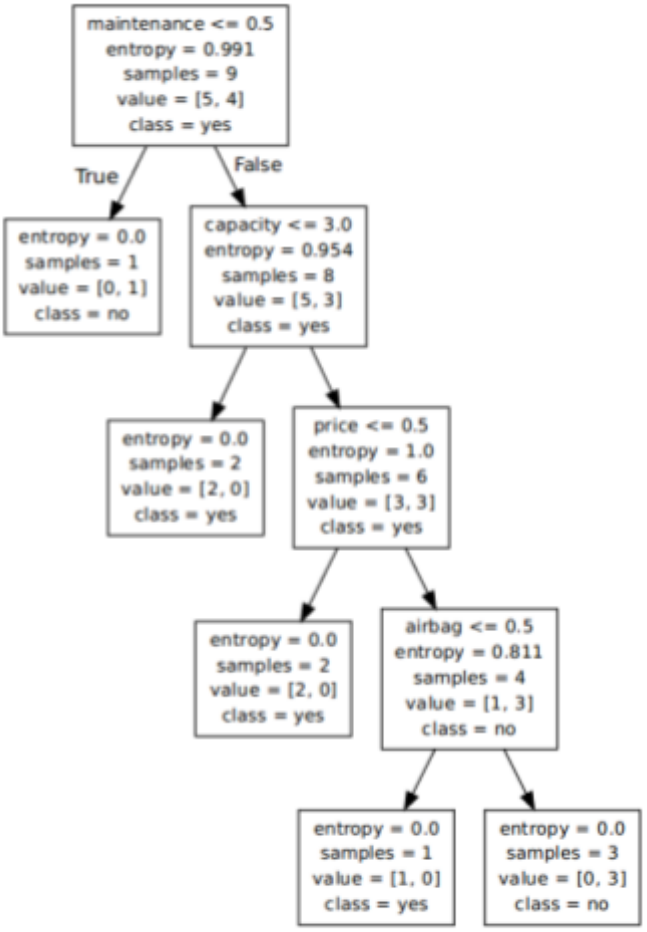
b) Test accuracy and INFORMATION GAIN of the root node (split using Information gain of every node):

```
DECISION TREE trained using Information Gain
------------------------------------
Information Gain of Root Node: 0.1427

Labels generated on test data:
1. no
2. yes
Actual Labels:
1. yes
2. yes

Accuracy on Test Data: 0.5
```

The trees that are formed using scikit-learn are plotted using `graphviz` module of python. The structure of the tree is submitted along with the assignment.



GINI Split



INFORMATION Gain

NOTE: I have used INTEGER ENCODING in my implementation as well as for scikit-learn. Since my implementation is a multi-way split with discrete attributes, even ONE-HOT-ENCODING will perform equally well. However, scikit-learn assumes ordinality among data when Integer Encoding of categorical data is used. In our case, this may work since the attribute values are in fact ordinal (high, medium, low), but in general case, this doesn't work well. In general, we need to convert categorical data to ONE-HOT-ENCODING where the feature space gets increased (e.g., suppose attribute 'price' has three values: high, medium, and low. A one-hot-encoding of this attribute will make three newer attributes as 'price_high', 'price_medium' and 'price_low', where each sample will have only one among these three attributes as value 1 while the other two as value 0. However, I have not used this in my implementation firstly because this is not needed and, secondly this would not be easily interpretable as to which among the original attribute was used for splitting.
