Introduction

In this project, the objective is to write python code for creating "Binary Decision Tree" classifier from scratch.

About Dataset

There are three datasets used in this analysis. All datasets consist of 7 columns. The first column is the class label (0 or 1) and the following columns indicate attributes which get only integer values. Train and test sets contain around 120 and 430 records, respectively.

Each dataset reflects classification of a subcategories of an animal species considering its attributes like length, height, weight, color, paw size and tale length.

After coding Binary ID3 classifier, the train data are used to train the model and then test data are used to evaluate the capability of the classifier.

Algorithm Implementation

Binary Decision Tree classifier is a type of ID3 classifier in which every node contains only two branches. In this classifier, the splitting criterion is chosen from among all the possible attribute-value pairs. Let's clarify it with a simple example.

Suppose a dataset has two attributes attr1 and attr2. Also, the attribute x1 take values from {a,b,c} and x2 from {c,d}.

All the splits are chosen from the initial attribute-value pair list which is a list of all pairs of attributes with their corresponding values as follows:

[(attr1, a),
(attr1, b),
(attr1, c),
(attr2, d),
(attr2, e)]

In each split step, the information gain for all attribute-value pairs are calculated and then is split is done using the pair with the highest gain. To better understand the tree in plotting, the node is labeled as, for instance, [(attr2 = d)?]. Finally this pair is removed from the attribute_value_pairs list. The process is continued until one of the following stop criteria is reached:

1. If all the attribute_value_pairs are considered in the split process.
2. If the depth of tree reached a predefined value.
3. If all the labels in the split data is pure and so the algorithm returns the value of y as label.

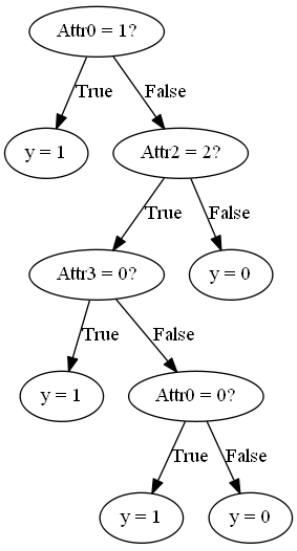
The final Decision tree is stored as nested dictionaries. Below you can see a sample of a binary decision tree in the form of a nested dictionary:

```
{(attr1, a, False):
     {(attr1, b, False):
          {(attr1, c, False): 1,
           (attr1, c, True): 0},
      (attr1, b, True):
          {(attr2, d, False):
               {(attr2, e, False): 0,
                (attr2, e, True): 1}
           (attr2, d, True): 1}},
(attr1, a, True): 1}
```

The decision tree is then plotted in the python console similar to the following sample. In this way the user can observe how decision tree is formed.

```
Tree
|
+=====[0,1,T]
|     1
+=====[0,1,F]
|     +=====[2,2,T]
|     |     +=====[3,0,T]
|     |     |     1
|     |     +=====[3,0,F]
|     |     |     +=====[0,0,T]
|     |     |     |     1
|     |     |     +=====[0,0,F]
|     |     |     |     0
|     +=====[2,2,F]
|     |     0
```

Finally, to better visualize the process of prediction in tree model, a figure which is plotting the nodes and leaves of the tree is drawn using graphviz package. To do so, firstly the dot_string source data is composed for graphviz and then using the dot format, graphviz creates the tree model and saves it as *Learned_Tree.png*. The figure below depicts a graph generated by graphviz for a sample binary decision tree.



Preprocessing

No preprocessing is needed in this project.

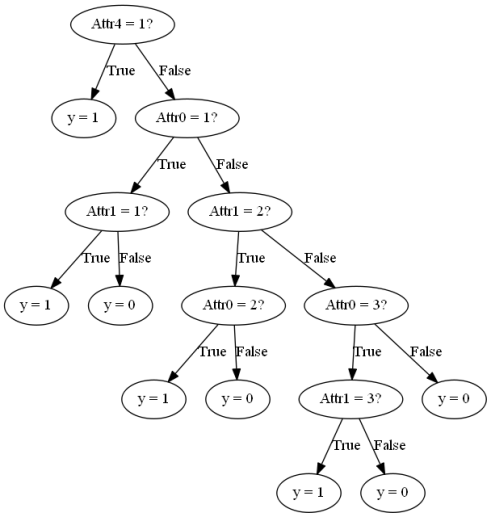
Project Outline

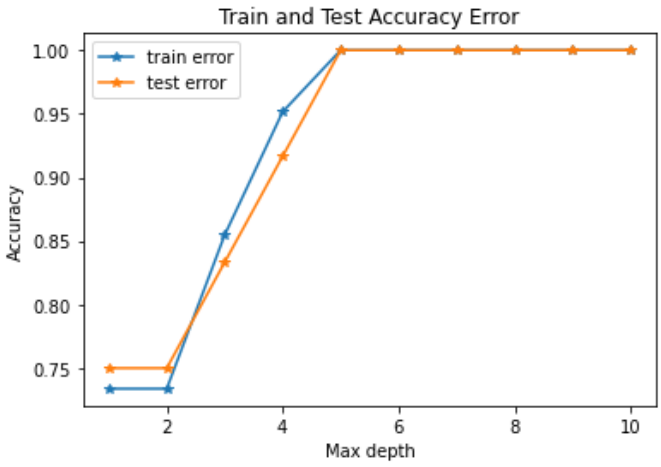
The report is outlined in 3 parts. In each part, the accuracy score of prediction of the decision tree model on the 3 datasets for different maximum tree depths are presented.

- Part 1: Implementation on dataset1
- Part 2: Implementation on dataset2
- Part 3: Implementation on dataset3
- Part 4: Conclusion
- Appendix A: Modules used in coding

Part 1: Implementation on dataset1

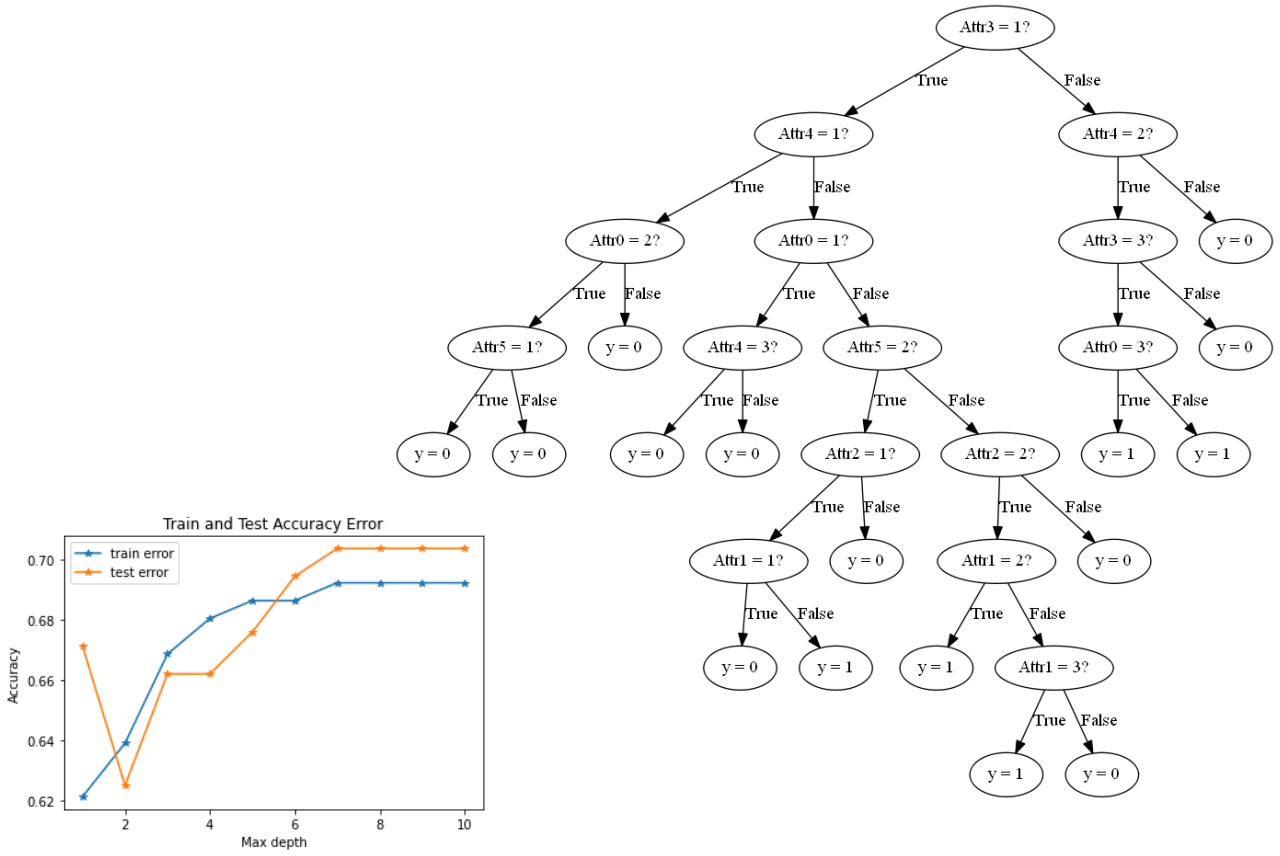
Accuracy score and graph plot of binary decision tree classifier on dataset 1 is shown below.





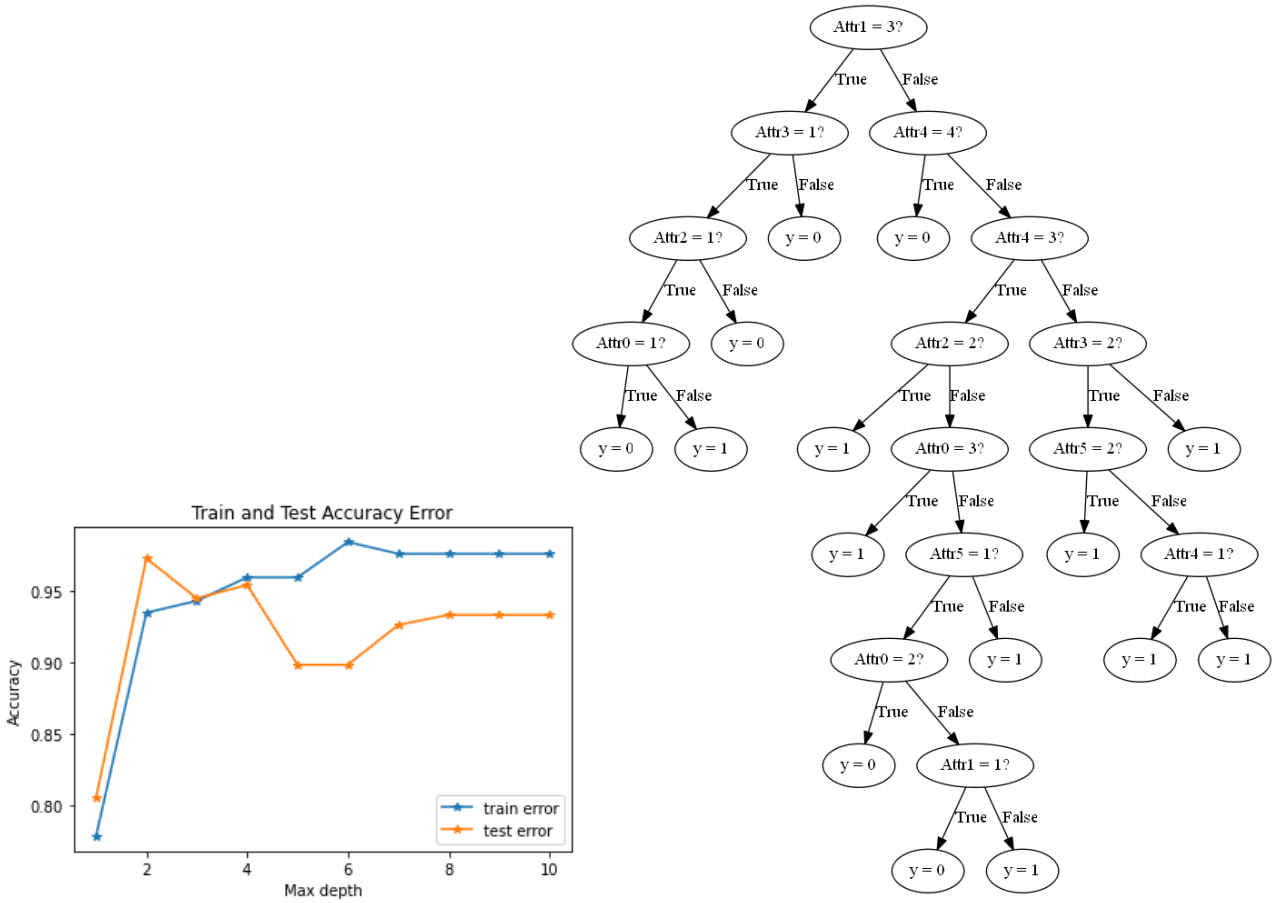
Part 2: Implementation on dataset2

Accuracy score and graph plot of binary decision tree classifier on dataset 2 is shown below.



Part 3: Implementation on dataset3

Accuracy score and graph plot of binary decision tree classifier on dataset 3 is shown below.



Part 4: Conclusion

It is seen that the binary decision tree algorithm is coded correctly and it considers the stopping thresholds in the right way. Also, increasing the maximum depth of the tree results in more accurate predictions.

Appendix A: Modules used in coding

The following Modules are used in python codes:

- **Class:** is created and used for better programming quality.
- **Logging:** an enhanced logger file is created to monitor the progression of codes and log any messages emitted from process. The results of run are logged in the file too in a pretty manner.
- **Argparse:** to read the data input by user.
- **Graphviz:** are used to create visual illustration of decision trees.