SMAI (CSE 471)

Assignment-1 Report

Part 1-1: Train the decision tree only on categorical data.

Basic Step:

1.Import Numpy and Pandas.

2.Select only those attributes/columns with categorical data.

3.Split the input dataset into 80 percent training data and 20 percent testing data.

Algorithm used:

1.Check if the input data is pure i.e. all values in column left are either zero or one and if it is pure return the value (zero or one which is present).

2.Calculate entropy for overall input data using E=-(q log(q) +(1-q) log(1-q)).

3.Calculate weighted sum (I) of entropy of a particular attribute.We consider all the unique values in that column and calculate entropy for each one of them and then using the individual entropy we compute the weighted sum and return it.

4.Calculate gain=E-I and select the attribute with max gain as the criteria to split.

5.Split the dataset into two halves and recursively follow the above steps and return the tree.

Results:

```
----------------------RESULTS-----------------------------------------
------------------------------------------------------------------------
True +ve= 0 True -ve= 1736 False +ve= 0 False -ve= 512
('Accuracy: ', 77.22419928825623)
('Precision: ', 0)
('Recall: ', 0)
('F1 score: ', 0)
```

Part 1-2: Train the decision tree with categorical and numerical features.

Basic Step:

1.Import Numpy and Pandas.

2.Select all attributes categorical as well as numerical.

3.Convert categorical attribute into numerical i.e.low,med,high get converted to 0,1,2.

4. Split the input dataset into 80 percent training data and 20 percent testing data.

Algorithm used:

1.Check if the input data is pure i.e. all values in column left are either zero or one and if it is pure return the value (zero or one which is present).

2.Calculate entropy for overall input data using E=-(q log(q) +(1-q) log(1-q)).

3.Calculate weighted sum (I) of entropy of a particular attribute. Since there can be a large number of different values in a column and thus, we can't split data on each one of them so to handle this problem we sort our data according to the attribute under observation and consider the average (breakpoint) of each pair consecutive values and the value with minimum weighted sum is selected. The weighted sum and the value(breakpoint) which gives minimum sum are returned.

4.Calculate gain=E-I and select the attribute with max gain as the criteria to split and breakpoint as the point to split.

5.If gain is less than a particular value or is zero means we can't split it further and in this case, we return the value in majority.

5.Split the dataset into two halves and recursively follow the above steps and return the tree.

Results:

```
True +ve= 488 True -ve= 1708 False +ve= 28 False -ve= 24
('Accuracy: ', 97.68683274021353)
('Precision: ', 94.57364341085271)
('Recall: ', 95.3125)
('F1 score: ', 0.9494163424124513)
```

Part 1-3: Contrast the effectiveness of Misclassification rate, Gini, Entropy as impurity measures.

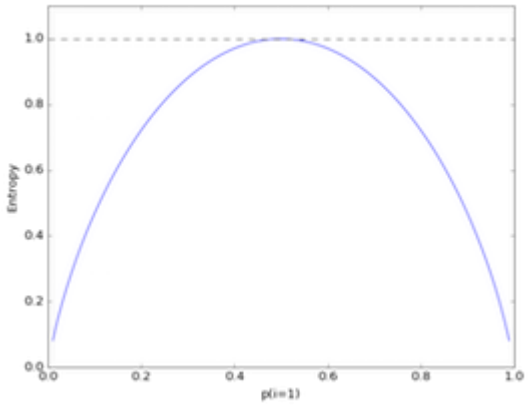
Formula used:

1.Entropy:

We write the Entropy equation as

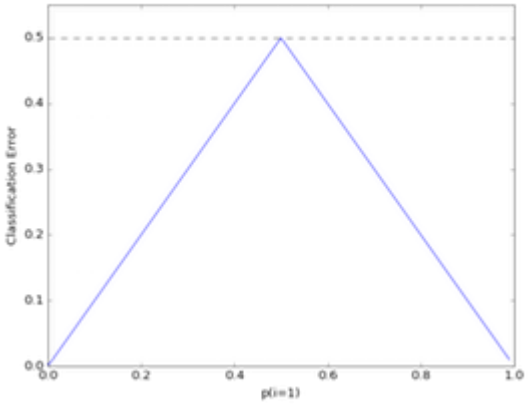
$$I_H(t) = -\sum_{i=1}^{C} p(i \mid t)\ log_2\ p(i \mid t)$$

or all non-empty classed $p(i \mid t) \neq 0$, where $p(i \mid t)$ is the proportion (or frequency or probability) of the samples that belong to class i for a particular node t; C is the number of unique class labels.



2. Miss Classification:

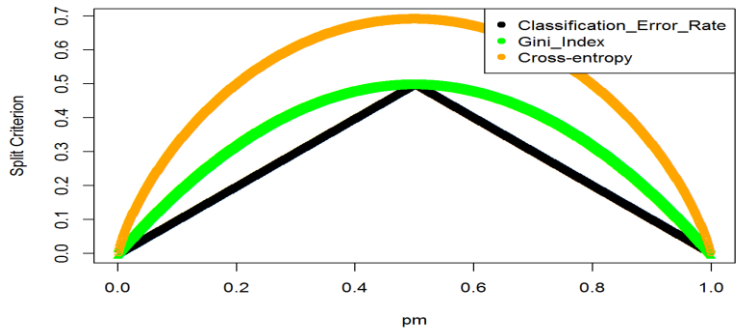
$$I_E(t) = 1 - max\{p_i\}$$



where *p(i)* is the proportion (or frequency or probability) of the samples that belong to class *i* for a particular node.

3.Gini Index:

$$1 - \sum_{t=0}^{k} P_t^2$$



4.Information Gain:

$$IG(D_p, x_i) = I(D_p) - \frac{N_{left}}{N_p} I(D_{left}) - \frac{N_{right}}{N_p} I(D_{right})$$

- IG: Information Gain
- x_i : feature to perform the split
- N_p : number of samples in the parent node
- N_{left} : number of samples in the left child node
- N_{right} : number of samples in the right child node
- I: impurity
- D_p : training subset of the parent node
- D_{left} : training subset of the left child node
- D_right: training subset of the right child node

Results:

```
----------------------RESULT ENTROPY-----------------------
True +ve= 499 True -ve= 1688 False +ve= 36 False -ve= 25
('Accuracy: ', 97.2864768683274)
('Precision: ', 93.27102803738318)
('Recall: ', 95.22900763358778)
('F1 score: ', 0.9423984891406987)
-----------------------------------------------------------
----------------------RESULT GINI INDEX--------------------
True +ve= 501 True -ve= 1683 False +ve= 41 False -ve= 23
('Accuracy: ', 97.15302491103202)
('Precision: ', 92.43542435424355)
('Recall: ', 95.61068702290076)
('F1 score: ', 0.9399624765478425)
-----------------------------------------------------------
----------------------RESULT MISS CLASSIFICATION-----------
True +ve= 383 True -ve= 1723 False +ve= 1 False -ve= 141
('Accuracy: ', 93.68327402135232)
('Precision: ', 99.73958333333334)
('Recall: ', 73.09160305343512)
('F1 score: ', 0.8436123348017621)
-----------------------------------------------------------
```

Part 1-4: Visualize training data on a 2-dimensional plot taking one feature (attribute) on one axis and other feature on another axis.

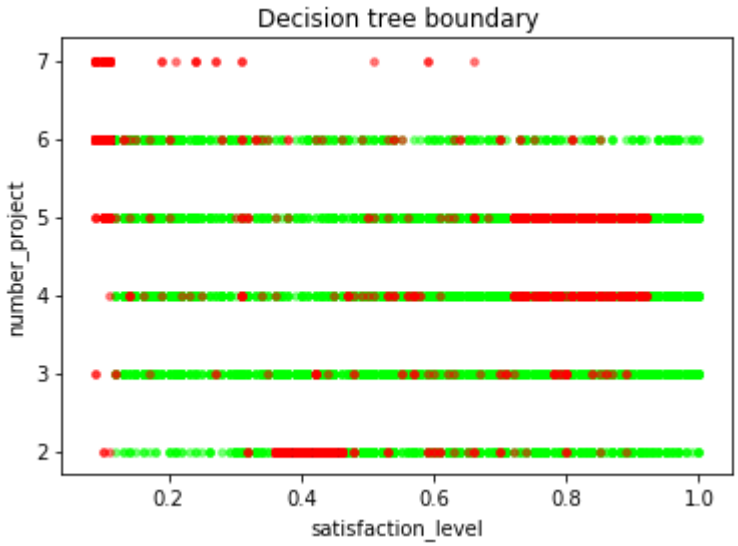
Steps to plot:

Step 1: Calculate the Information Gain of all the features and select the two attribute (a,b) with maximum information gain.

Step2: Using these two attributes plot the graph between these attribute and the attribute left.

Step3: To plot the graph use matplot library.

Results:



Part 1-5: Plot the training and validation error with respect to number of nodes in the decision tree.

Nodes vs Error:

For each node count build a new tree until that node count and using the tree predict result. Using the predicted result build the graph.