Vatsal Soni 2018201005

SMAI Assignment-1 Report

PART-1: Train decision tree only on categorical data. Report precision,recall, f1 score and accuracy.

- Import all the required modules like numpy,pandas,matplotlib.
- Load dataset in dataframe.
- Separate input and output data from dataframe.
- Split dataset into two parts 1) Training data 2) Validation data.
- Define numeric attributes that we should not consider for this part and drop it from input dataframe.
- Define node structure which contains value as feature name and number of positive and negative counts.(ID-3 Algorithm)
- Choose best node by calculating information gain and build the tree(train model) by repeating same process for each node (Entropy as impurity criteria).
- Test the model by feeding testing input and calculate predicted output.
- Calculate precision,recall,f1 score and accuracy from predicted output and test output.
- **Reference:** q-1-1.py

Trained Model	**System Model**
Confusion Matrix [[1709 1] [538 0]] precision recall f1-score support 0 0.76 1.00 0.86 1710 1 0.00 0.00 0.00 538 **Accuracy**:- 0.7602313167259787	Confusion Matrix [[1709 1] [538 0]] precision recall f1-score support 0 0.76 1.00 0.86 1710 1 0.00 0.00 0.00 538 **Accuracy**:- 0.7602313167259787

PART-2: Train the decision tree with categorical and numerical features.Report precision, recall, f1 score and accuracy.

- Import all the required modules like numpy,pandas,matplotlib.
- Load dataset in dataframe.
- Separate input and output data from dataframe.
- Split dataset into two parts 1) Training data 2) Validation data.
- Define numeric attributes.
- Define node structure which contains value as feature name,number of positive, negative counts,split point and n- number of children and n-values for each branch .
- Numeric attribute node has two children and one split point.

- In order to calculate split point
    - Take unique values for that feature
    - Calculation information gain for each unique value
    - One unique value for which information gain is maximum will become splitting point for that attribute.
- Categorical attribute node has n-children and n-values(one label corresponds to one value).
- Choose best node by calculating information gain for both categorical attributes and numeric attributes and build the tree(n-ary) (train model) by repeating same process for each node.
- Test the model by feeding testing input and calculate predicted output.
- Calculate precision,recall,f1 score and accuracy from predicted output and test output.
- **Reference:** q-1-2.py

Trained Model	**System Model**
Confusion Matrix [[1682 18] [27 521]] precision recall f1-score support 0 0.98 0.99 0.99 1700 1 0.97 0.95 0.96 548 **Accuracy**:- 0.979982206405694	Confusion Matrix [[1690 34] [14 510]] precision recall f1-score support 0 0.99 0.98 0.99 1724 1 0.94 0.97 0.96 524 **Accuracy**:- 0.9786476868327402

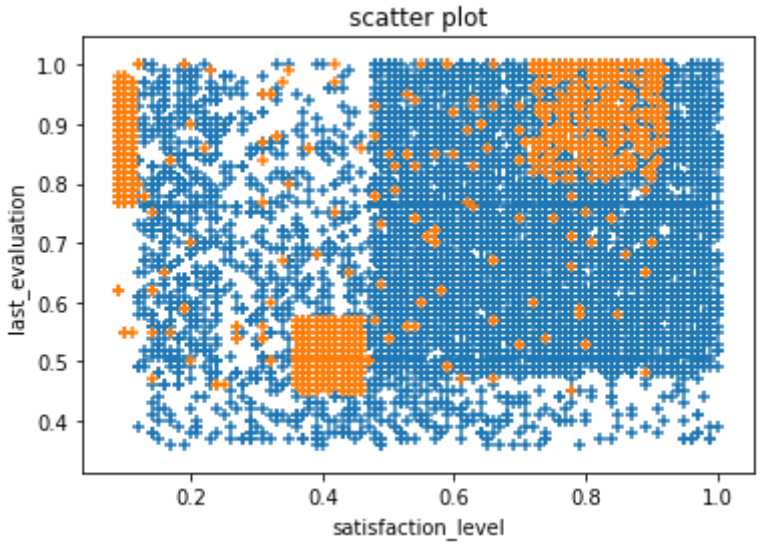
PART-3: Contrast the effectiveness of Misclassification rate, Gini, Entropy as impurity measures in terms of precision, recall and accuracy.

- Import all the required modules like numpy,pandas,matplotlib.
- Load dataset in dataframe.
- Separate input and output data from dataframe.
- Split dataset into two parts 1) Training data 2) Validation data
- Define numeric attributes.
- Train the model by taking entropy as impurity criteria (Refer part-2).
- Train the model by taking gini as impurity criteria.
- Train the model by taking misclassification rate as impurity criteria.
- Test the model by feeding testing input and calculate predicted output.
- Calculate precision,recall,f1 score and accuracy from predicted output and test output for each impurity criteria and compare it.
- **Reference:** q-1-3.py

Entropy impurity criteria	**Gini impurity criteria**	**Misclassification impurity criteria**
Confusion Matrix [[1702 25] [27 494]] precision recall f1-score support 0 0.98 0.99 0.98 1727 1 0.95 0.95 0.95 521 **Accuracy**:- 0.9768683274021353	Confusion Matrix [[1709 18] [32 489]] precision recall f1-score support 0 0.98 0.99 0.99 1727 1 0.96 0.94 0.95 521 **Accuracy**:- 0.9777580071174378	Confusion Matrix [[1693 34] [85 436]] precision recall f1-score support 0 0.95 0.98 0.97 1727 1 0.93 0.84 0.88 521 **Accuracy**:- 0.9470640569395018

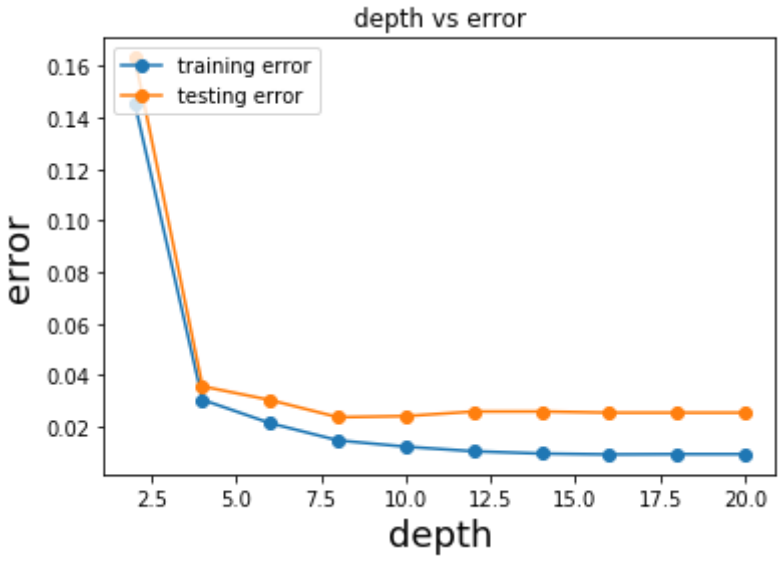
PART-4: Visualise training data on a 2-dimensional plot taking one feature on one axis and other feature on another axis. Take two suitable features to visualise decision tree boundary.

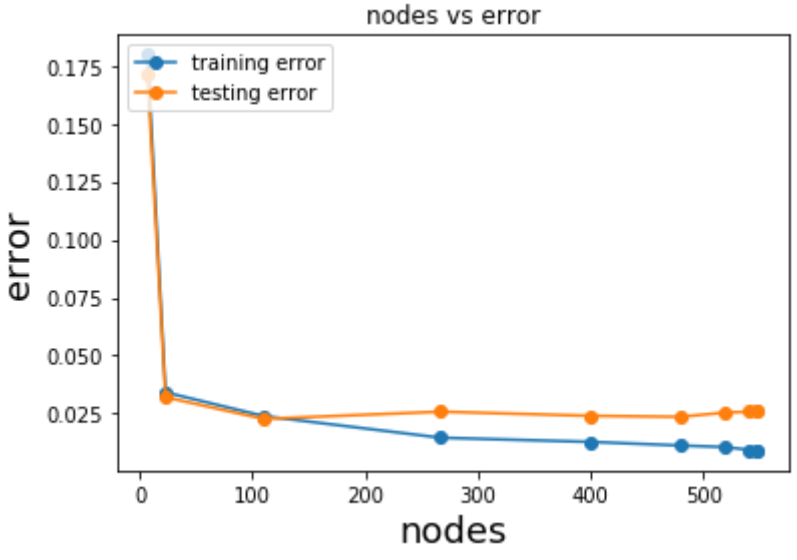
- Import all the required modules like numpy,pandas,matplotlib.
- Load dataset in dataframe.
- Separate input and output data from dataframe.
- Split dataset into two parts 1) Training data 2) Validation data.
- As we have already trained the model(in Part-2), we know that what are the two best attributes.
- 1) Satisfaction level.
- 2) last_evaluation.
- Take satisfaction_level on x-axis and last_evaluation on y-axis and plot the scatter graph.
- **Reference:** q-1-4.py



PART-5: Plot a graph of training and validation error with respect to depth of your decision tree. Also plot the training and validation error with respect to number of nodes in the decision tree.

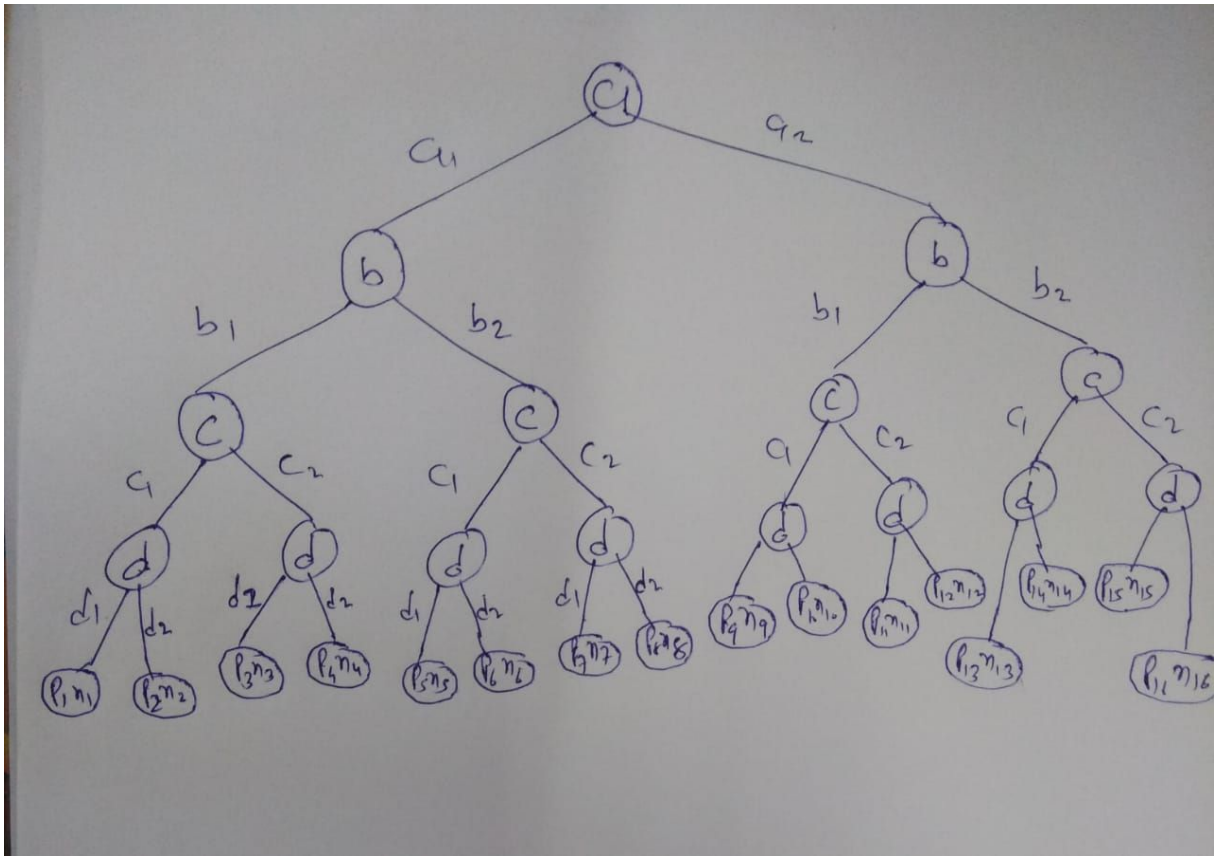
- Import all the required modules like numpy,pandas,matplotlib.
- Load dataset in dataframe.
- Separate input and output data from dataframe.
- Split dataset into two parts 1) Training data 2) Validation data.
- For depth = 2 to 20 (20 is the max depth of model).
    - Trained the model by restricting depth(Refer part-2).
    - Test the model by feeding testing input and calculate predicted output.
    - Calculate error rate from predicted output and test output.
    - Store error rate for training and testing in list for each depth.
- Plot the graph by taking depth on x-axis and error rate on y-axis.
- Keep record of number of nodes at each depth and plot the graph by taking nodes on x-axis and training error on y-axis.
- **Reference:** q-1-5.py





PART-6: Explain how decision tree is suitable handle missing values in data.

- In decision tree we can handle missing values in following way.
- The idea is whenever we encounter missing value for any particular attribute then we will visit all the branches of that attribute, whenever we reach to the leaf node we will sumup all the positive counts and negative counts. Whichever is dominating would be the predicted answer.
- Assume that we have 4 attributes (a,b,c,d)



    - Case-1: (a1,b1,c1,_) :- d attribute value is missing.
        - If p1+p2 > n1+n2 then answer will be positive.
        - else answer will be negative.
    - Case-2: (a1,_,c1,d2) :- b attribute is missing
        - If p2+p6 > n2+n6 then answer will be positive.
        - Else answer will be negative
    - Case-3: (a2,_,_,d2) :- b and c attribute is missing.
        - If p10+p12+p14+p16>n10+n12+n14+n16 then answer will be positive.
        - Else answer will be negative.
    - Case-4: (a1,_,c2,_) :- b and d attribute is missing.
        - If p3+p4+p7+p8 > n3+n4+n7+n8 then answer will be positive.
        - Else answer will be negative.
- In this way missing values can be easily handled by decision tree hence decision tree is suitable for handling missing values.