Group 11 (19CS10071, 19CS30041)
Anurat Bhattacharya
Sayantan Saha

Machine Learning Assign 1

18th September 2021

OVERVIEW

The task is to implement and use a decision tree classifier to classify and measure the accuracy of the predictions of the models on the **Pima Indians Diabetes Database**:
https://www.kaggle.com/uciml/pima-indians-diabetes-database

GOALS

1. Analyse and preprocess the data and do discretisation for continuous /many value attributes
2. Implement a decision tree classifier and classify on that dataset.
3. Analyse the results
4. Do pruning to further improve the accuracy

Description of the dataset

The dataset given to us shows the onset of diabetes based on eight different diagnostic measures.
The Diagnostic measures are listed below.

1. Pregnancies
2. Glucose
3. BloodPressure
4. SkinThickness
5. Insulin
6. BMI
7. DiabetesPedigreeFunction
8. Age

And the “Outcome” column depicts whether the patient with those diagnostic measures is diabetic or not. ‘0’ denotes that the patient is not diabetic and ‘1’ otherwise. Our aim is to use those eight diagnostic features as input attributes and predict whether a patient is diabetic.

Steps to Run:

1.Run pip install -r requirements.txt

2.Run .python assgn1_19CS10071_19CS30041.py

PROCEDURE

Data Analysis, Data Preprocessing and Data Discretizing :

First, we called dropna function(pandas.DataFrame) to drop missing data.The data length did not change implying no missing data.
Second,we checked the distribution of given data for each attribute.
We found that :

A. The dataset for “Pregnancies” is an almost right skewed distribution.
B. The dataset for “Glucose” is an almost normal distribution.
C. The dataset for “BloodPressure” is an almost normal distribution.
D. The dataset for “SkinThickness” is an almost normal distribution.
E. The dataset for “Insulin” is an almost right skewed distribution.
F. The dataset for “BMI” is an almost normal distribution.
G. The dataset for “DiabetesPedigreeFunction” is an almost right skewed distribution.
H. The dataset for “Age” is an almost right skewed distribution.

As the dataset for all the attributes are continuous(noy binary or ternary), we had to discretize the datasets of each attribute or perform the decision tree taking each value of an attribute as different labels. So, we listed some methods of data discretization to test the accuracy of the decision tree.

1. For each column attribute, the max and min were found and the data for that attribute were divided in ten uniform(of equal length) parts for the range (min,max). The thresholds would be like [(min + {(max-min)/10}), (min +2* {(max-min)/10}),.....,(min +9* {(max-min)/10})].
   Here we obtained,
   With entropy, 58.51%,72.72% before and after pruning(saved as uniform.png)
   With gini, 58.18%, 66.23% before and after pruning

2. As most of the attributes are normally distributed, we chose thresholds [μ-3*(sig), μ-2*(sig), μ-(sig), μ, μ+(sig), μ+2*(sig), μ+3*(sig)] for each column attribute .Here μ and sig are mean and standard deviation of a column attribute.
   We know, $P(\mu-3*(sig) < x < \mu+3*(sig)) = 99.7\%$ for a normally distributed variable x.
   Here we obtained,

With entropy, 61.16%,77.92% before and after pruning(saved as sigma3.png)
With gini, 63.24%, 81.16% before and after pruning

3. In this case, we decreased no.of thresholds a bit from the last one. Here, the thresholds are [μ-2*(sig), μ-(sig), μ, μ+(sig), μ+2*(sig)].
   Here we obtained,
   With entropy, 63.70%,81.18% before and after pruning(saved as sigma2.png)
   With gini, 63.05%, 84.41% before and after pruning

4. In this one, we increased the no. of thresholds from the 2nd case. The thresholds are [μ-0.8*(sig), μ-0.6*(sig), μ-0.4*(sig),μ-0.2*(sig), μ, μ+0.2*(sig), μ+0.4*(sig), μ+0.6*(sig),μ+0.8*(sig)].
   As we also know that P(μ-(sig) < x < μ+(sig)) = 95% for a normally distributed variable x, we divided that region in many parts.
   Here we obtained,
   With entropy, 51.16%,70.78% before and after pruning(saved as many_intv.png)
   With gini, 51.49%, 64.93% before and after pruning

5. In this one, we do not use any thresholds or no discretization is done. Each new value of an attribute column is considered as a new label and the decision tree is built.
   Here we obtained,
   With entropy, 42.40%,62.98% before and after pruning(saved as no_discrete.png)
   With gini, 41.23%, 62.99% before and after pruning

Inference:

On the basis of the above experiment we thus found that maximum accuracy is obtained when we use the criteria for discretisation as [μ-2*(sig), μ-(sig), μ, μ+(sig), μ+2*(sig)] (of course padded with the min and max of the data at both ends).This can be understood as follows,In this dataset the actual value of the data point does not matter.What matters is in which division is it when approximated under the normal curve,towards extreme high, towards extreme low or in between.This is also intuitively valid as often in medical data for diagnosing something, it is seen weather some parameter lie within the normal range or above it or below it.

Implementation of Decision Tree:

1. The decision tree was implemented using the ID3 algorithm.First after getting preprocessed data we split it into an 80:20 training and validation set.
2. Then the ID3 algorithm was run for 10 times for each random 80:20 split.The accuracy was averaged and the tree with the best validation accuracy was returned.

Description of some major functions:

1. read_prepare_data(): Reads the data in csv format to a pandas dataframe,discretises it and returns the tuple (data,bin) bin denoting threshold values of discretisation
2. train_test_split(data) : Makes a copy of the data splits it into training and validation and (80:20 ratio) and returns in the form of the tuple :((x_train,y_train),(x_val,y_val))
3. getEntropy(p,n): Returns entropy of a sample with p positive and n negative examples
4. getGini(p,n): Returns gini index of a sample with p positive and n negative examples
5. class Node : Class to store each node of the tree
    a. isLeaf() : Returns if the node is a leaf or not
    b. adj : Dict mapping attribute value to next node
    c. attrChosen :Attribute chosen to split current node into future nodes
    d. predictedVal : returns majority label of current node
6. class DTClassifier : The main class of decision tree classifier
    a. propagate(node) : This recursively implements the ID3 algorithm and gros the tree.
    b. predict(a,currNode) : Predicts the outcome based on input vector a and currentNode traversed till now
    c. getAccuracy(valX,valy) : To get accuracy on validation sets valX and valy
    d. getNodes():To get total number of nodes
    e. reduced_errorPruning(val_X,val_y) : Does reduced error pruning with respect to val_X and val_y (as validation sets)
    f. attrDict : This maps the attribute name(string) to the values it can take
    g. attrInd : This maps the attribute name(string) to the column number or index of the attribute.
    h. Func : This stores the impurity function to be used
7. class model : Class for managing and performing the processes as asked in the questions
    a. trainedModel() : Does a 80 20 split and trains and returns a model along with associated validation data
    b. finMaxAccTree(noIter) : Runs trainedModel noiter times and returns avg accuracy and best acc tree.
8. print_tree(rootNode,fileName,bins,attrInd) : Function to print the tree using graphviz

Analysis of test accuracy with respect to tree depth and size:

The accuracy on validation data was plotted for(pre pruning was done with current depth is reached for some node) depths in range 0...8 (0 meaning only one node).This exercise was done considering both the gini index and the information gain.Also in this step the plot for validation accuracy vs number of nodes was also generated and the results studied.

Pruning on the best tree obtained originally

Reduced error pruning was done on the best tree obtained earlier

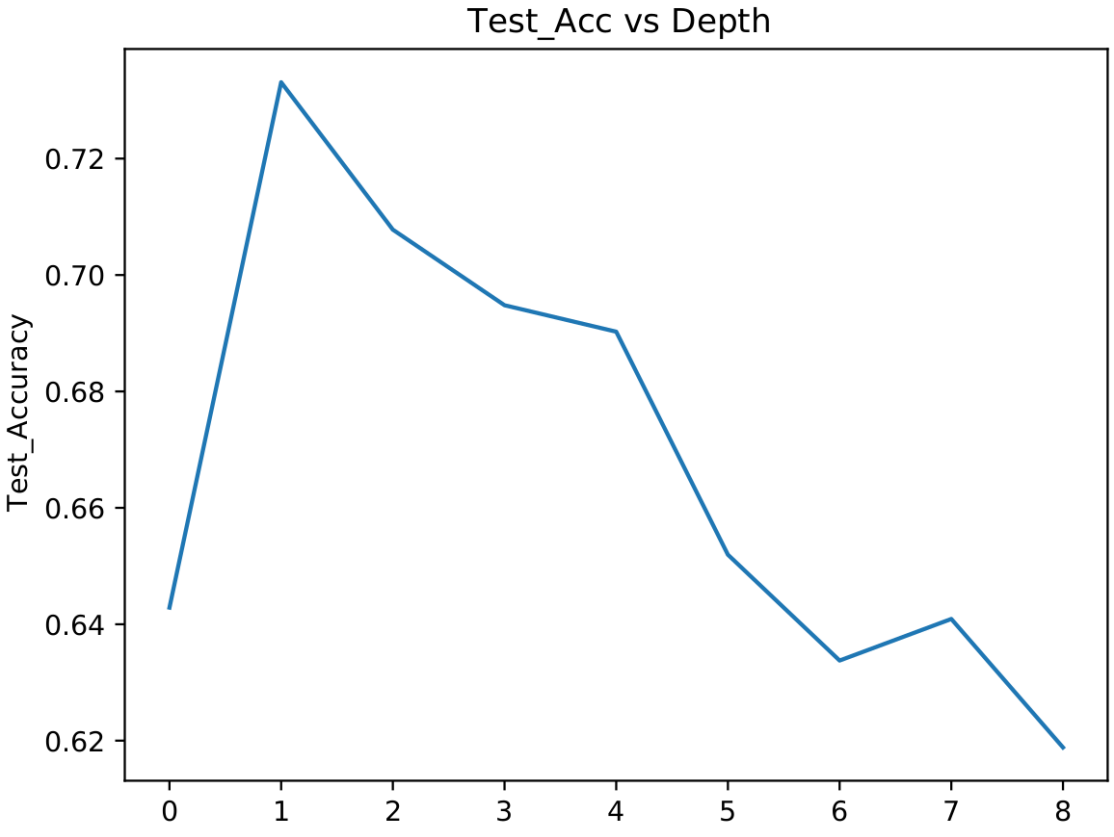
Finally both the original tree and the pruned tree for both impurity functions were printed using graphviz

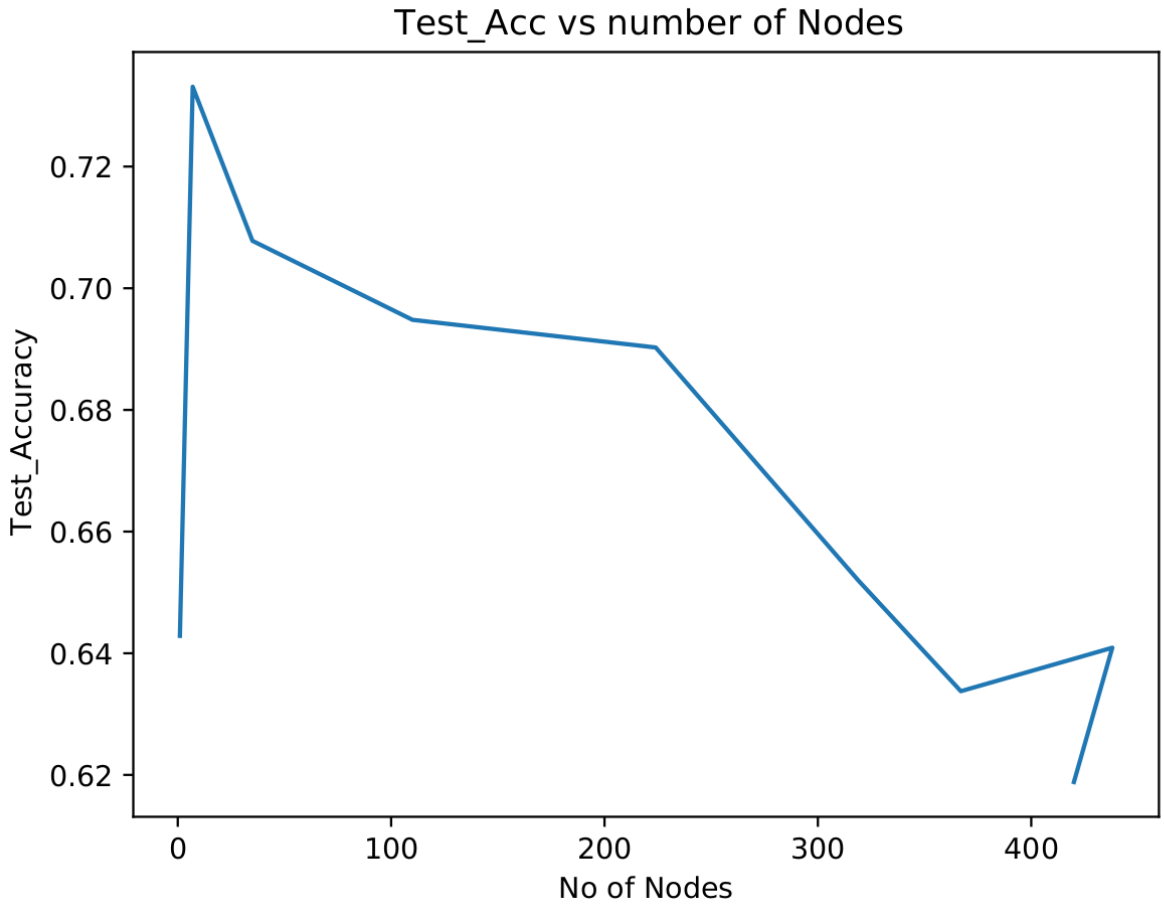
Results Obtained(Ans to Questions):

```
anurat@anurat-Inspiron-3576:~/mlassgn1$ /bin/python3 /home/anurat/mlassgn1/assgn1.py
********Taking entropy as Impurity function*******

Original Accuracy : 0.637012987012987

Accuracy After Pruning : 0.8181818181818182

********Taking Gini Index as Impurity function*******

Original Accuracy : 0.6305194805194805

Accuracy After Pruning : 0.8441558441558441
```

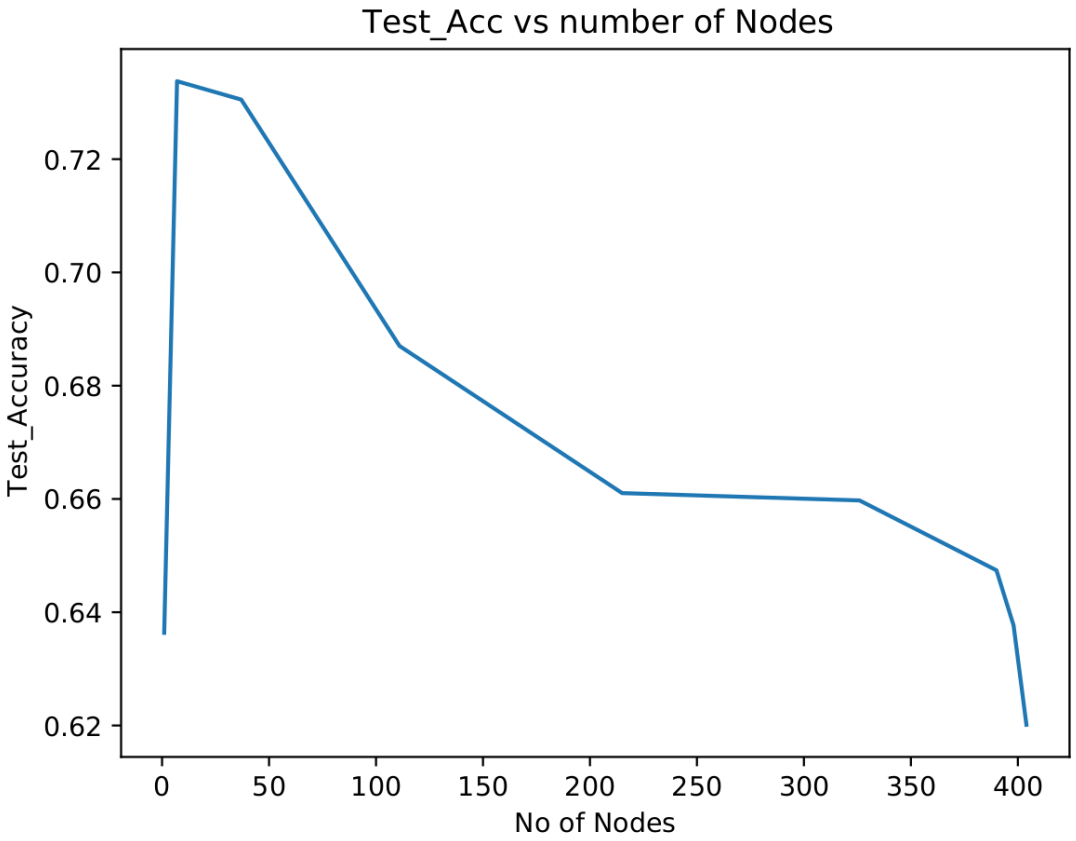
1. It is observed that the accuracy is more on using gini index as impurity function
2. 1. For gini index accuracy of 63.05 was obtained initially on averaging over 10 times
   2.For entropy accuracy of 63.70 was obtained initially on averaging over 10 times.
3. Using Gini Index:
   a. **Effect of Accuracy on depth:**

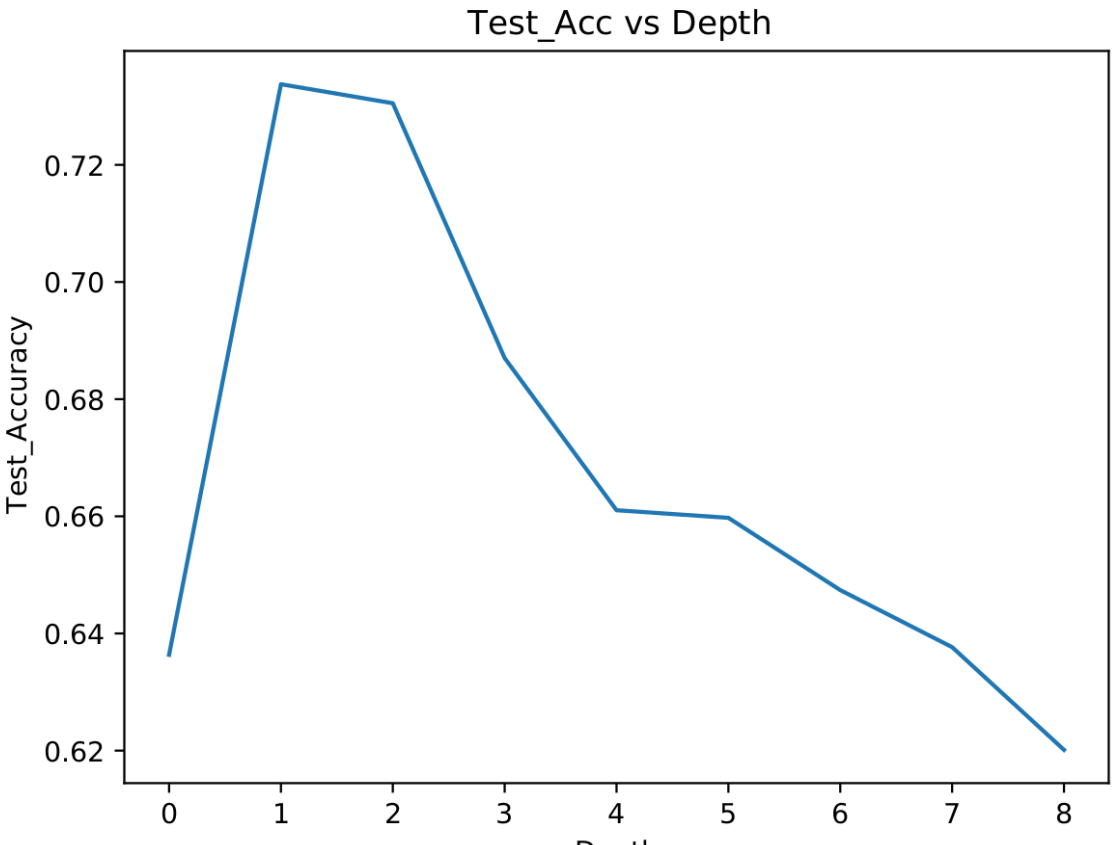
   Inference:From the graph we see that only 2 levels are required to get highest accuracy (Root is assumed level 0, we assumed 0 indexed depth) (Plots below)


Test_Acc vs Depth
Test_Accuracy
0.72
0.70
0.68
0.66
0.64
0.62
0
1
2
3
4
5
6
7
8


Test_Acc vs number of Nodes
Test_Accuracy
0.72
0.70
0.68
0.66
0.64
0.62
0
100
200
300
400
No of Nodes

Using Entropy:





Inference:So from the graph we see that only 2-3 levels are required to get highest accuracy (Root is assumed level 0, we assumed 0 indexed depth)

4. **Pruning :** As mentioned in procedure reduced error pruning was done for both cases using gini index and entropy
   For using gini index Accuracy after pruning : 84.16%

   For using entropy Accuracy after pruning : 81.82%

5. **For both cases** Gini index and Entropy the final and original decision trees were printed ,saved as
   Entropy_tree_original.gv.pdf,Entropy_tree_after_pruning.gv.pdf,
   Gini_tree_after_pruning.gv.pdf,Gini_tree_after_pruning.gv.pdf
   These were generated using graphviz