CS60050
MACHINE LEARNING

Assignment 1

Problem 1

1. Major functions used in main.py

There is only one file main.py. It consists following methods

- select_best_tree: Finds the best tree over 10 random splits
- abal_heigh: Finds the best depth limit for the tree by iterating over depths from 1 to 15
- Class Decission tree
- build_tree: This function recursively builds the decision tree, it checks if the current node should be a leaf or not, if not it finds the best split for the node and then recursively computes its left and right children and returns the node.
- fit: Builds a decision tree which fits the training data X. Calls build_tree on the complete dataset X and assigns the returned node to the root.
- calc_accuracy: Given test data, uses the predict function to predict the labels and compares them with the given labels to get the accuracy
- count_nodes: Counts the number of nodes in the tree
- create_children: For a given dataset, creates children based on a given attribute (column id) and a value for the same
- split: Splits the data into train and test, can pass the size the split, whether to shuffle the data, and the random seed for splitting
- val_ split: Similar to the above function but instead of two it splits into three parts
- Igain_Entropy_exp: runs all experiments with measure set to entropy

2. Dataset_C.csv

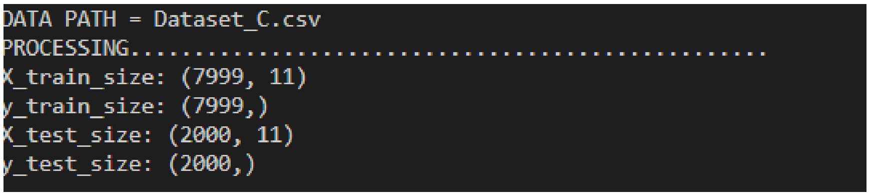
The given dataset. It has the following attributed, all of which are continuous variables:

1) id
2) Gender
3) Age
4) Driving_License
5) Region_Code
6) Vechicle_Age
7) Vehicle_Damage
8) Annual_Profit
9) Policy_Sales
10) Response

By following the standard ID3 algorithm, we only take binary splits, that is for discrete variables we check == and != and for continuous variables given a value we split the data on the basis of <= . If not a leaf we calculate all possible splits given the remaining data, then we choose the best split given the specified metric (entropy) and then recursively calculate the nodes left and right subtrees. For a node we first check if it should be leaf node, (check if depth is more than max depth, all variables belong to the same class or number of examples to split are less than the minimum specified leaf size). We take 80:20 train test splits, and for validation we take 60:20:20 train test validation splits.

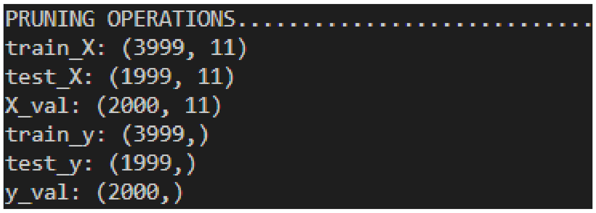
For 80:20 splits

HERE WE HAVE TAKEN ONLY 10000 inputs because its taking lot of time to compute 1.3 lakh inputs



```
DATA PATH = Dataset_C.csv
PROCESSING..................................................
X_train_size: (7999, 11)
y_train_size: (7999,)
X_test_size: (2000, 11)
y_test_size: (2000,)
```

For 60:20:20 splits



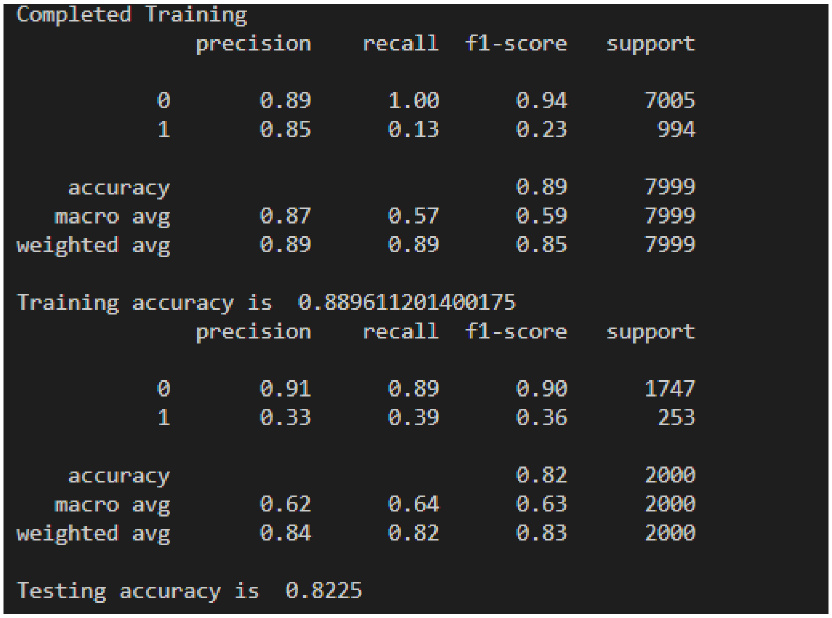
```
PRUNING OPERATIONS...........................
train_X: (3999, 11)
test_X: (1999, 11)
X_val: (2000, 11)
train_y: (3999,)
test_y: (1999,)
y_val: (2000,)
```

We do the following experiments

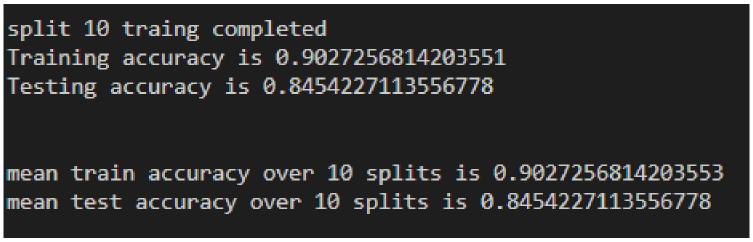
1. We construct a tree with depth=10 on a 80:20 split and print its classification report
2. We now create 10 random 60:20:20 splits, construct tree with depth=10 and choose the split with the best test accuracy
3. On the best split obtained in experiment 2, we do depth and number of nodes analysis
4. Now using the validation split obtained in experiment 2 we perform post pruning on the tree obtained in 2.

Using Entropy

1) Tree constructed using 80:20 split, using entropy, max depth of the tree = 10, Accuracy over 10 random 60:20:20 splits, max depth = 10

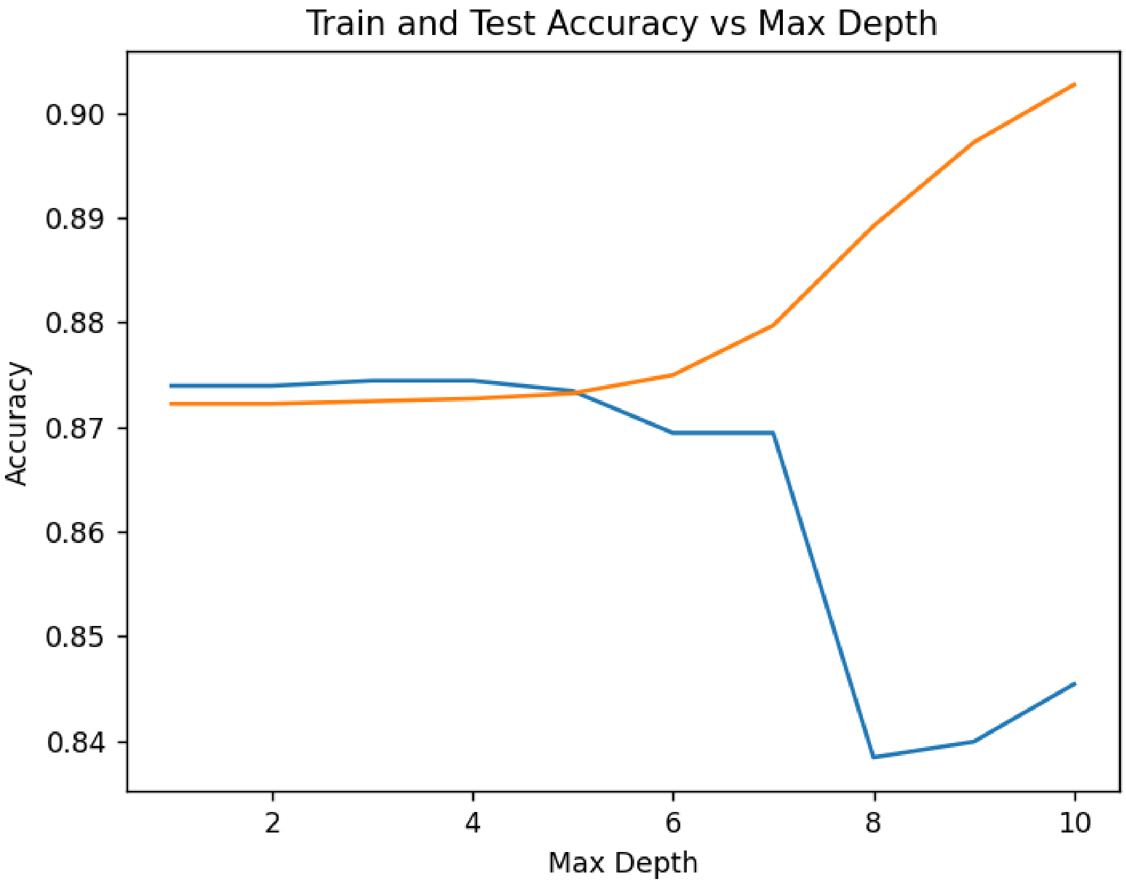


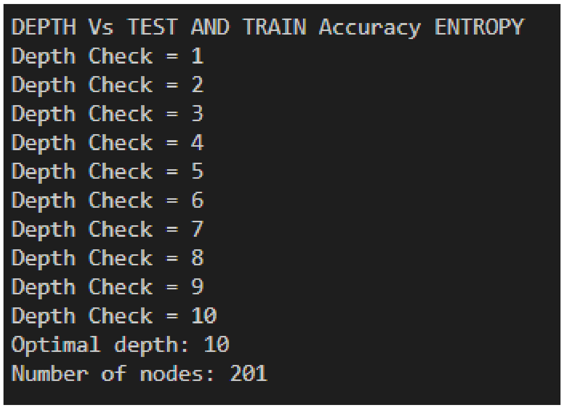
```
Completed Training
              precision    recall  f1-score   support

           0       0.89      1.00      0.94      7005
           1       0.85      0.13      0.23       994

    accuracy                           0.89      7999
   macro avg       0.87      0.57      0.59      7999
weighted avg       0.89      0.89      0.85      7999

Training accuracy is  0.889611201400175
              precision    recall  f1-score   support

           0       0.91      0.89      0.90      1747
           1       0.33      0.39      0.36       253

    accuracy                           0.82      2000
   macro avg       0.62      0.64      0.63      2000
weighted avg       0.84      0.82      0.83      2000

Testing accuracy is  0.8225
```



```
split 10 traing completed
Training accuracy is 0.9027256814203551
Testing accuracy is 0.8454227113556778


mean train accuracy over 10 splits is 0.9027256814203553
mean test accuracy over 10 splits is 0.8454227113556778
```

2. Accuracy vs depth analysis

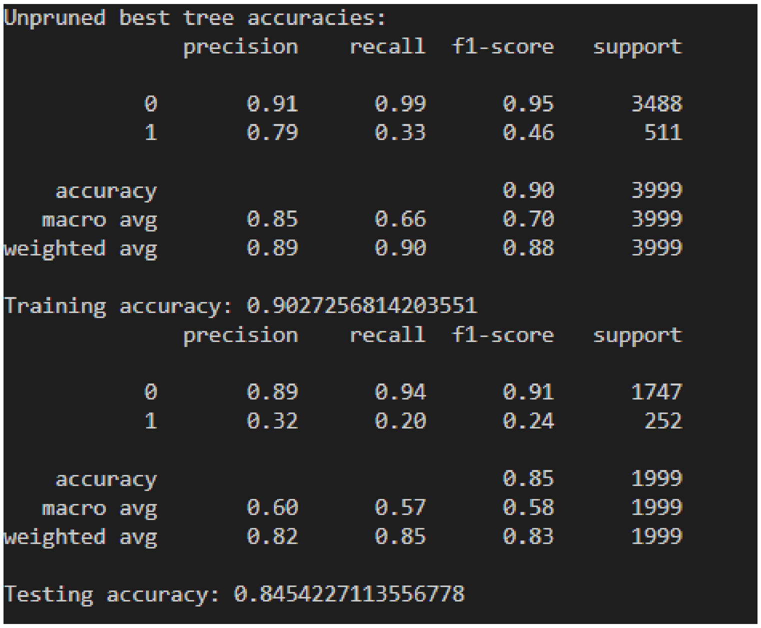




```
DEPTH Vs TEST AND TRAIN Accuracy ENTROPY
Depth Check = 1
Depth Check = 2
Depth Check = 3
Depth Check = 4
Depth Check = 5
Depth Check = 6
Depth Check = 7
Depth Check = 8
Depth Check = 9
Depth Check = 10
Optimal depth: 10
Number of nodes: 201
```

We now apply post pruning to the best tree obtained in experiment 2, at this point note that the max depth of the tree obtained in experiment 2 is 10 and the optimal depth is less than 10.

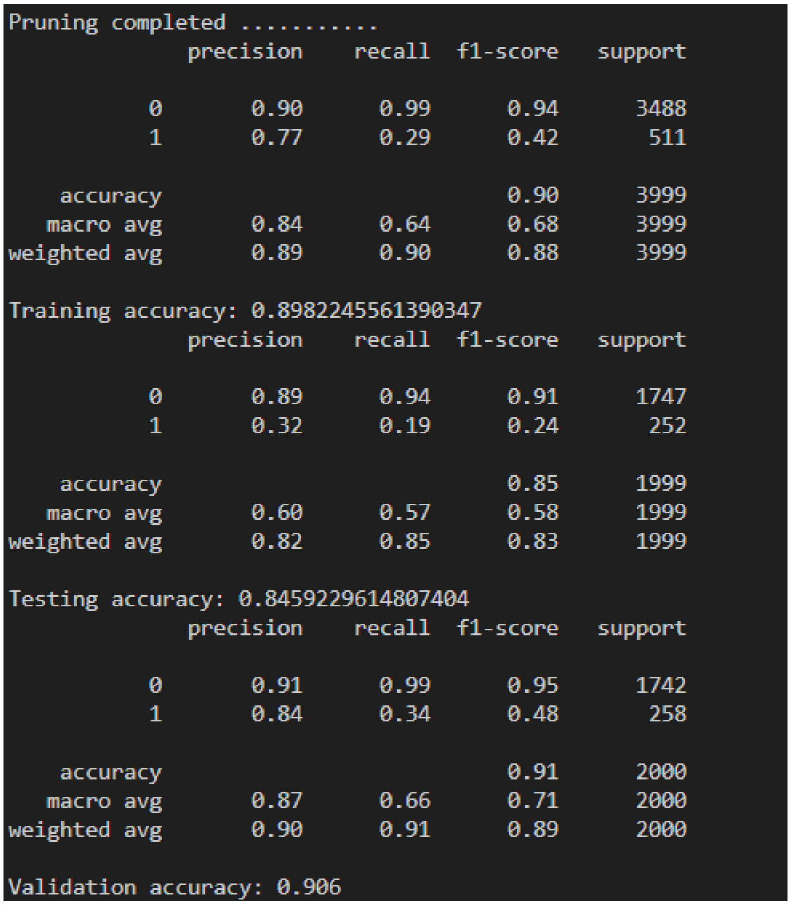
Before pruning the accuracies are as follow:



```
Unpruned best tree accuracies:
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      3488
           1       0.79      0.33      0.46       511

    accuracy                           0.90      3999
   macro avg       0.85      0.66      0.70      3999
weighted avg       0.89      0.90      0.88      3999

Training accuracy: 0.9027256814203551
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      1747
           1       0.32      0.20      0.24       252

    accuracy                           0.85      1999
   macro avg       0.60      0.57      0.58      1999
weighted avg       0.82      0.85      0.83      1999

Testing accuracy: 0.8454227113556778
```

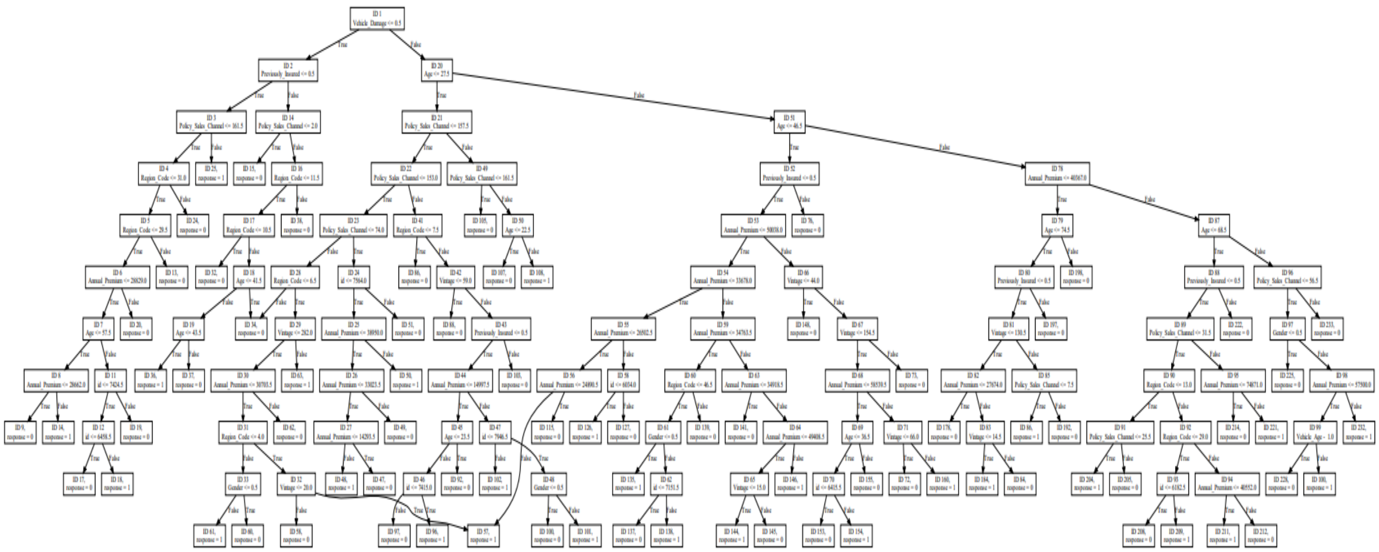
Also submitting PDF of TREE for better view

After pruning we observe that the validation and test accuracy increase, this is because pruning reduces the overfit of the depth=10 tree and reduces the model complexity. After pruning the accuracies are as follows:



```
Pruning completed ...........
              precision    recall  f1-score   support

           0       0.90      0.99      0.94      3488
           1       0.77      0.29      0.42       511

    accuracy                           0.90      3999
   macro avg       0.84      0.64      0.68      3999
weighted avg       0.89      0.90      0.88      3999

Training accuracy: 0.8982245561390347
              precision    recall  f1-score   support

           0       0.89      0.94      0.91      1747
           1       0.32      0.19      0.24       252

    accuracy                           0.85      1999
   macro avg       0.60      0.57      0.58      1999
weighted avg       0.82      0.85      0.83      1999

Testing accuracy: 0.8459229614807404
              precision    recall  f1-score   support

           0       0.91      0.99      0.95      1742
           1       0.84      0.34      0.48       258

    accuracy                           0.91      2000
   macro avg       0.87      0.66      0.71      2000
weighted avg       0.90      0.91      0.89      2000

Validation accuracy: 0.906
```

Post pruning the tree looks like this:


ID 1
Vehicle_Damage <= 0.5
True
False
ID 2
Previously_Insured <= 0.5
ID 20
Age <= 27.5
ID 14
Policy_Sales_Channel <= 2.0
ID 21
Policy_Sales_Channel <= 157.5
ID 51
Age <= 46.5
ID 15,
response = 0
ID 16
Region_Code <= 11.5
ID 22
Policy_Sales_Channel <= 153.0
ID 49
Policy_Sales_Channel <= 161.5
ID 52
Previously_Insured <= 0.5
ID 17
Region_Code <= 10.5
ID 38,
response = 0
ID 23
Policy_Sales_Channel <= 74.0
ID 41
Region_Code <= 7.5
ID 105,
response = 0
ID 50
Age <= 22.5
ID 53
Annual_Premium <= 50038.0
ID 76,
response = 0