Assignment 1: Decision Tree

Group: 41
Parth Tusham 19CS30034
Shashwat Shukla 19CS10056

Procedure

- We have used the standard ID3 algorithm for designing the Decision Tree. The data used for training was provided in .csv format. The data is the Avila data set that has been extracted from 800 images of the 'Avila Bible', an XII century giant Latin copy of the Bible.). We have shuffled the data. We considered the split of 80/20 as Train, Validation, set as the Test set was provided separately. The data had the following attributes:
    - Intercolumnar distance
    - upper margin
    - lower margin
    - Exploitation
    - row number
    - modular ratio
    - interlinear spacing
    - Weight
    - peak number
    - modular ratio/ interlinear spacing
    - Class: A, B, C, D, E, F, G, H, I, W, X, Y (Target Variable)

Major Functions

- **grow_tree:** Build a decision tree by recursively finding the best split on the basis of either gini index or entropy/information gain, using the formula:
    - $Gain(S,A) = Entropy(S) - \sum (|S_V| / |S|) * Entropy(S_v) \; (\forall \; v \in Values(A))$
    - $Gini = 1 - \sum (p_i)^2$ for all $i \in \{class_1, ... class_n\}$

- **best_split:** Finds the best split for a node.
    - Where "Best" means that the average impurity of the two children, weighted by their population, is the smallest possible.
    - Additionally, it must be less than the impurity of the current node. To find the best split, we loop through all the features and consider all the midpoints between adjacent training samples as possible thresholds.
    - We compute the Gini impurity or entropy of the split generated by that particular feature/threshold pair and return the pair with the smallest impurity.
    - Returns:
        - best_idx: Index of the feature for best split, or None if no split is found.
        - best_thr: Threshold to use for the split, or None if no split is found.

- **prune:** Prunes the Decision Tree built with the height provided to construct_tree. We have used the validation set to prune the tree.

- **predict:** Provided a decision tree root and data, this function predicts the output on the basis of training. It outputs the prediction values along with accuracy.

- **print_decision_tree:** Uses Digraph (from Graphviz package) to print the tree. Stores the output in ".gv" and ".pdf" format

Helper Functions

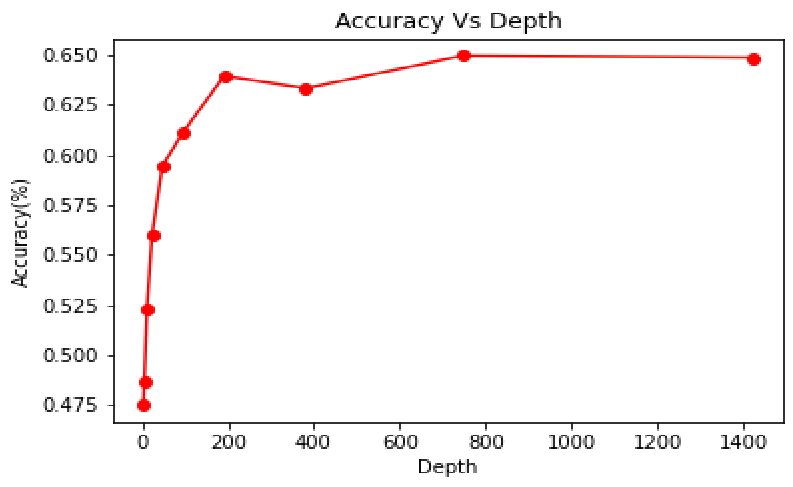
- **train_test_split:** Splits the data into three parts i.e, Train and Validation in the ratio of 80:20.
- **gini:** Computes Gini impurity of a non-empty node.
- **Info_gain:** Given a Pandas Series, it calculates the entropy.
- **remove_children:** Removes the children of the current node. Used in the pruning method.
- **Count_node:** Provided the root of a Decision Tree, it counts and returns the number of nodes.
- **is_leaf:** Returns true if the given node is a leaf node. Used to terminate the recursion in the predict_one and get_node method.

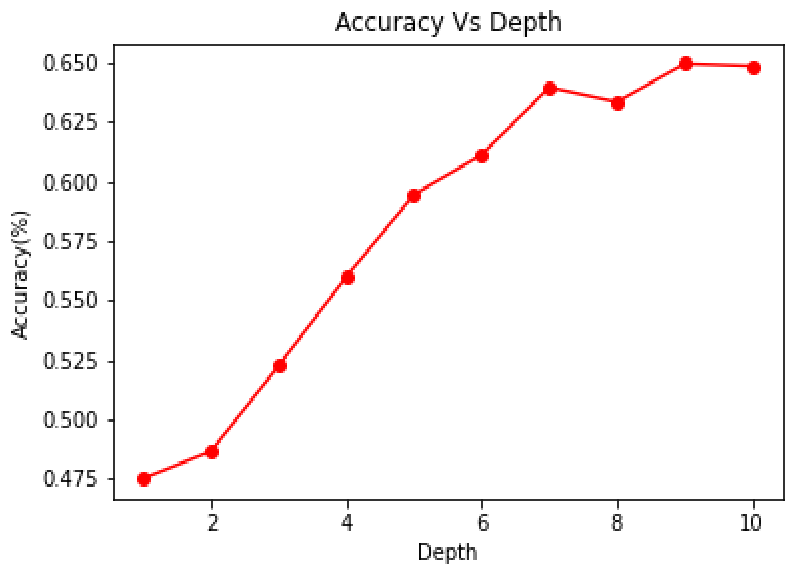
Pruning

- We will first recursively prune the children of the current node then will decide whether to prune the correct node or not
- Pruning begins from the parent of leaf nodes. We consider a node. We temporarily remove the children of the node and calculate the new error
- If the error decreases on the validation set, then we remove the children of that node and update the attributes of the current node, thus updating the tree.
- We repeat these steps until no more pruning is possible (overfitting has reduced significantly).
- The above-mentioned pruning method is also called reduce-error- pruning. In short, we use the train set to build the tree, use the verification set for pruning, and use the test set to estimate the final accuracy

Results

- Before Pruning the maximum accuracy was 68% after pruning it increased to 70%




Accuracy Vs Depth
Accuracy(%)
Depth
0.650
0.625
0.600
0.575
0.550
0.525
0.500
0.475
2
4
6
8
10