Title : Decision tree learning for car evaluation.

Introduction:

Decision Tree Learning is a powerful and widely-used machine learning technique that forms the foundation for various predictive modeling tasks, including classification and regression.A decision tree is a hierarchical structure that recursively partitions the dataset into subsets based on the values of input features, ultimately leading to a prediction or decision at each leaf node. These tree-like structures are constructed by selecting the best features to split the data at each internal node, optimizing for information gain by calculating entropy based on attributes and examples.

For our offline on decision tree , I have implemented a decision tree based on the car evaluation dataset. Based on a probabilistic approach , the entire dataset in splitted into two segments. In general 80% of the dataset are added to the training data and the rest 20% in the test data. After that, some new test data were used to check the accuracy of our trained model of decision tree.

Algorithm:

Decision tree learning follows a recursive and top-down approach to construct a tree structure. The algorithm for decision tree used in our offline can be broken down like below :

1. Select the Best Attribute : At each internal node of the tree, choose the attribute that provides the best split for the current dataset. The "best" attribute is determined based on information gain. We calculate the entropy before splitting an attribute and then calculate entropy after the split and find the best attribute .
2. Split the Examples : Divide the examples into subsets, each corresponding to a unique value or range of values for the chosen attribute. This splitting process transforms the initial dataset into smaller, more manageable subsets. Thus the problem splits into a smaller subset of the original problem.
3. Repeat Recursively : For each subset created in the previous step, repeat the decision tree construction process recursively. Continue splitting the data

until one of the stopping conditions is met. Common stopping conditions include achieving pure nodes (nodes containing data from a single class) , having no attributes left or having no examples left.

4. Assign a Class or Value : Once a leaf node is reached (either through splitting or stopping conditions), assign a class label for classification tasks or a predicted value for regression tasks. The assigned class or value represents the majority class or the mean of the target values within the leaf node's dataset.
5. Prediction :With the trained decision tree, the decision tree can make predictions on new, unseen data points. Starting from the root node, traverse the tree by following the attribute splits until you reach a leaf node. The class or value associated with that leaf node is the final prediction.

By following this algorithm, decision tree learning creates a model that can be used for classification or regression tasks. The pseudoCode followed for decision tree can be shown as :

function DECISION-TREE-LEARNING($examples, attributes, parent\_examples$) **returns** a tree

if $examples$ is empty **then return** PLURALITY-VALUE($parent\_examples$)
else if all $examples$ have the same classification **then return** the classification
else if $attributes$ is empty **then return** PLURALITY-VALUE($examples$)
else
  $A \leftarrow \text{argmax}_{a \in attributes}$ IMPORTANCE($a, examples$)
  $tree \leftarrow$ a new decision tree with root test A
  for each value v_k of A **do**
    $exs \leftarrow \{e : e \in examples$ **and** $e.A = v_k\}$
    $subtree \leftarrow$ DECISION-TREE-LEARNING($exs, attributes - A, examples$)
    add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
  return $tree$

Figure 18.5 The decision-tree learning algorithm. The function IMPORTANCE is described in Section 18.3.4. The function PLURALITY-VALUE selects the most common output value among a set of examples, breaking ties randomly.

In order to find the best argument , we followed the information gain property . The procedure can be viewed as :

$$Gain(A) = B(\frac{p}{p+n}) - Remainder(A) \ .$$

The entropy calculation is followed by the equation :

$$\text{Entropy:} \quad H(V) = \sum_k P(v_k) \log_2 \frac{1}{P(v_k)} = -\sum_k P(v_k) \log_2 P(v_k) \ .$$

The Remainder function can be calculated as :

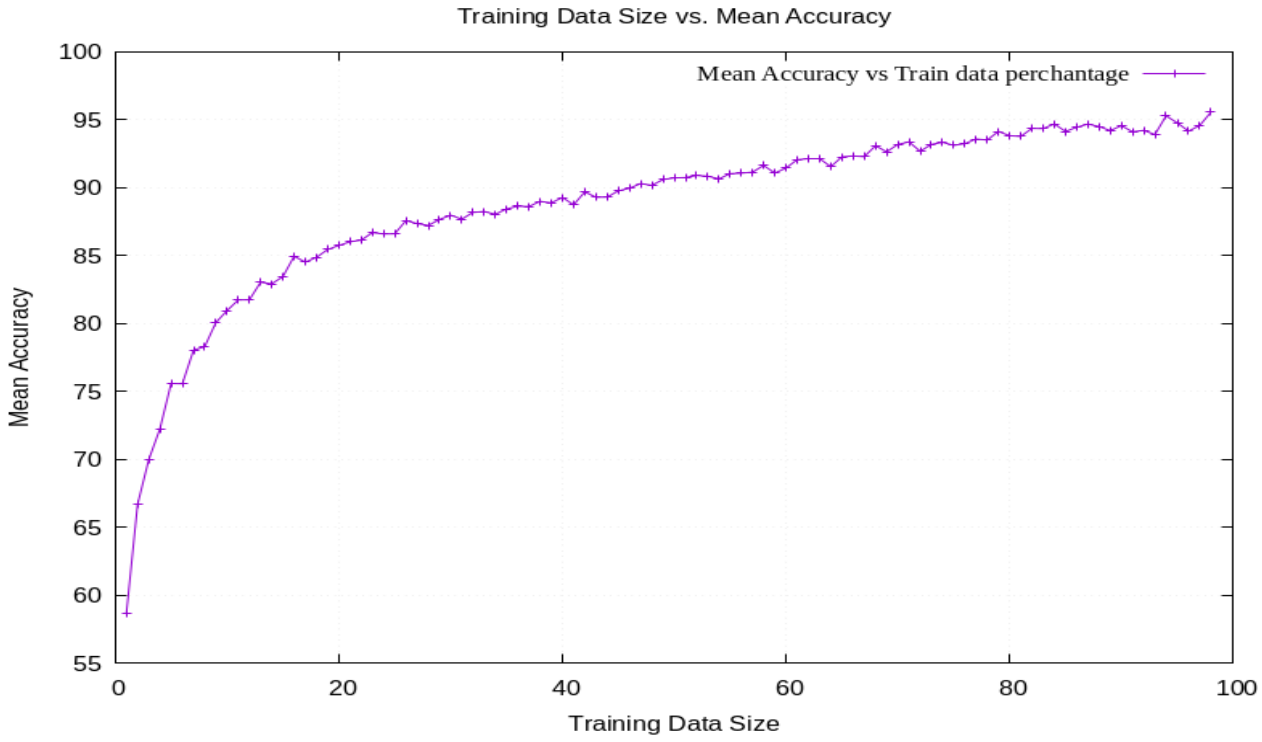
$$Remainder(A) = \sum_{k=1}^{d} \frac{p_k+n_k}{p+n} B(\frac{p_k}{p_k+n_k}) \ .$$

So , basically in every step , we are calculating the best choice for attribute based on Entropy(Before splitting the data) - Entropy (After splitting the data) and choosing the one that is the best choice for us.

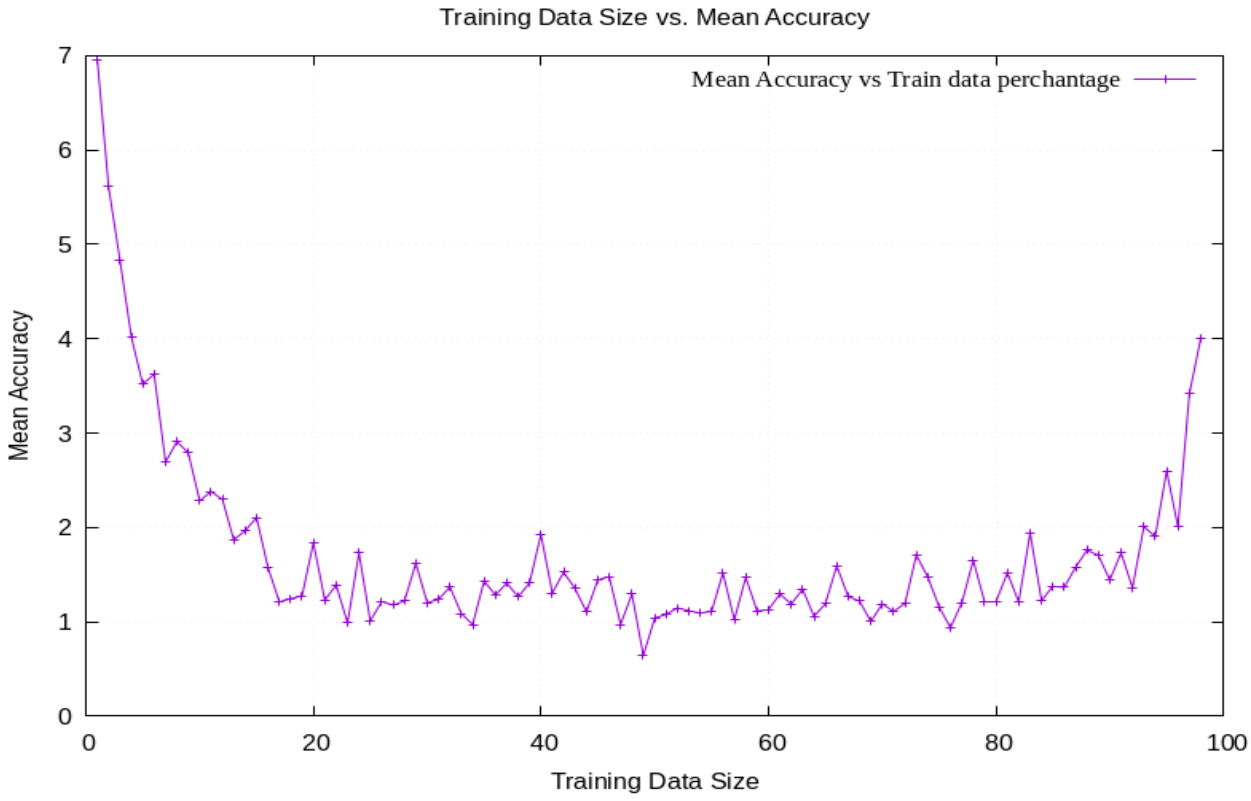
Output :

The output dataset can be done based on our percentage of splitting ratio of training and testing dataset. Then we can plot a graph showing how this impacts the accuracy of our graph. In general , increasing training data increases test accuracy . But since we are splitting at random , in some cases that may not come true.

In the graph here , we varied the split ratio between 1% to 98% and plotted the graph . As we can see that the accuracy in case of test data / unseen data with the amount of training data we provided.



Then for standard deviation , we can also see the graph as ,



The output can also be visualized at :
Decision Tree Output Data

Discussion:

Decision trees inherently perform feature selection by choosing the most informative attributes to split the data. Attributes that contribute significantly to the prediction are placed higher in the tree, while less relevant features are pushed lower or pruned altogether. This helps in identifying important factors in the dataset. Our generated decision tree performs in general of 93% accuracy in our test examples , which shows that our built decision tree did not overfit or underfit to our training data. Also a standard deviation of nearly 1% shows the model working correctly for any number of iterations.

In order to further improve our decision tree , we can prune our decision tree. After the decision tree is constructed, a pruning step can be applied to reduce its complexity and improve generalization. Pruning involves removing branches of the tree that do not contribute significantly to its predictive accuracy.

Conclusion:

Our decision tree is correctly trained based on some training data .We can verify it from the above 90% accuracy in our test cases. Further optimization can be done by pruning the decision tree.