Decision Tree Learning for Car Evaluation

1905066 - Abir Muhtasim

Overview

In this assignment, we implemented a decision-tree learning algorithm. Then using the algorithm and the car evaluation dataset (partitoned into 80% - 20% train set and test set) we constructed a decision tree. We predicted ouptuts for the train set from the decision tree and measured its accuracy.

Implementation

The Decision-Tree-Learning algorithm was implemented following the algorithm taught in our curriculum.

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**
      $exs \leftarrow \{e : e \in examples \text{ and } e.A = v_k\}$
      $subtree \leftarrow$ DECISION-TREE-LEARNING(exs, $attributes - A$, $examples$)
      add a branch to $tree$ with label $(A = v_k)$ and subtree $subtree$
    return $tree$

Here in each step, we identify the most *important* attribute. We generate a subtree with this identified attribute as its root, incorporating all possible values of that attribute as its branching nodes. This procedure is executed recursively.

Importance is calculated as the information gain of each attribute. The one with the most information gain is chosen. Here information gain is the amount of entropy reduction before and after the split on attribute A

$$Gain = Entropy\ before\ split - Entropy\ after\ split$$

And entropy $H(V)$ of a random variable V with values v_k, each with probability $P(v_k)$ is defined as :

$$H(V) = -\sum_k P(v_k) \log_2 P(v_k)$$

So, $Entropy\ before\ split$ is $H(output)$.

Now, if an attribute A with d distinct values divides the training set E into subsets $E_1, E_2, \ldots, E_d$. Then $Entropy\ after\ split$ is the expected entropy (weighted average) of the d subsets after splitting on attribute A.

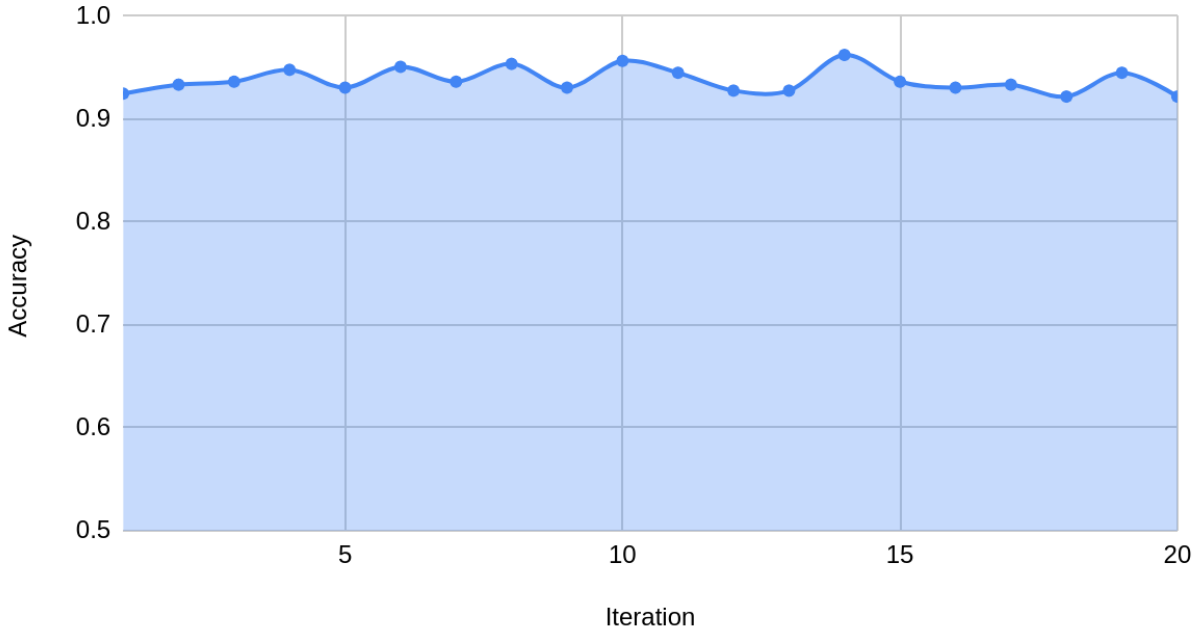
$$Remainder(A) = \sum_{k=1}^{d} \frac{|E_k|}{size\ of\ training\ set} H(E_k)$$

So,

$$Gain(A) = H(output) - Remainder(A)$$

Result

After running the experiment 20 times, we got the following accuracies

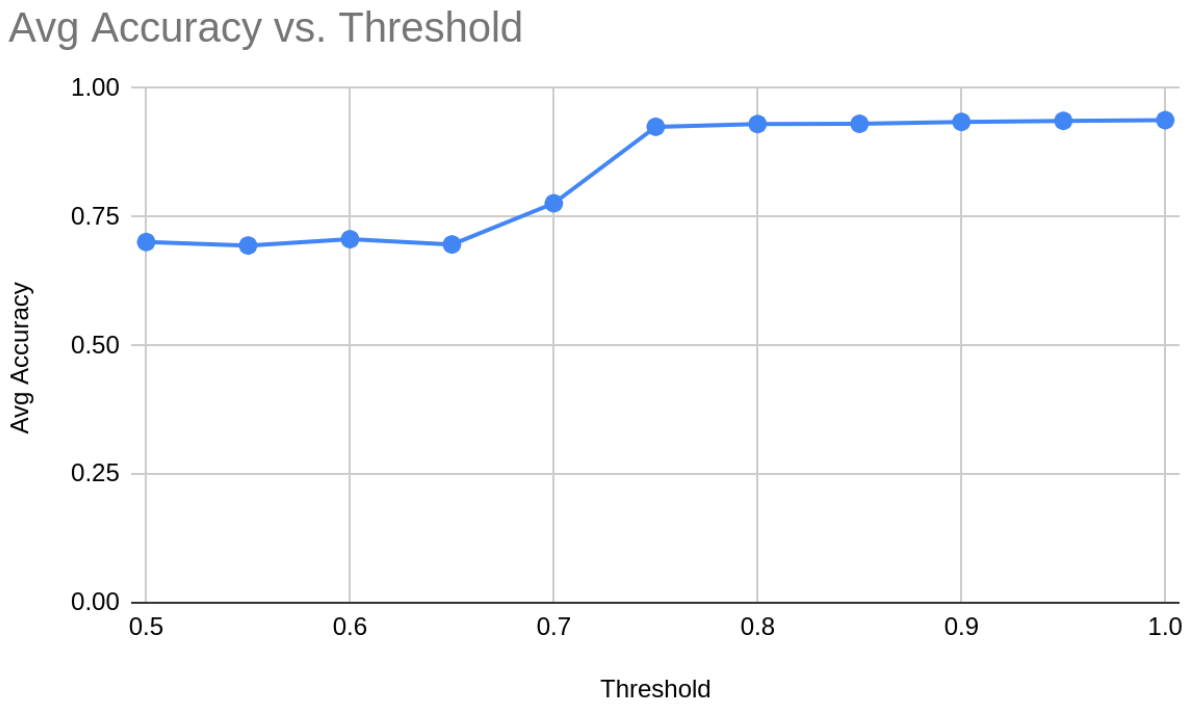


Max Accuracy	Avarage Accuracy	Standard Deviation
0.962	0.938	0.01184

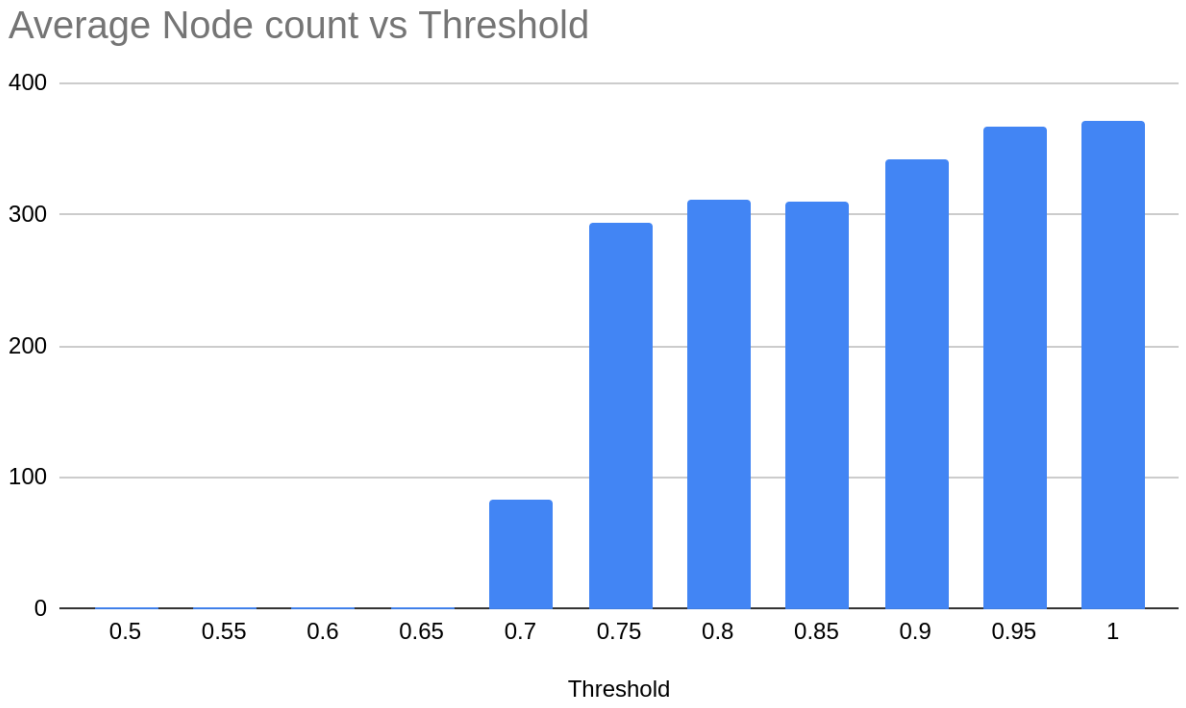
The standard deviation demonstrates a notably low value, while the average accuracy consistently hovers around 93%, with a maximum recorded accuracy of 96%. So we can say that our Decision Tree's performance is quite good.

Pruning

Subsequently, we implemented a straightforward pruning technique. At some node, if the percentage of a specific output value surpasses a predefined threshold, we halt further branching at that node. Instead, we designate the most frequently occurring output value as the leaf node for that branch.



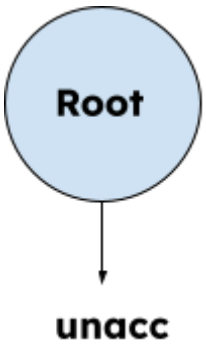
Here we see that there is a sudden from 0.65 to 0.75. In order to explain that first see two more data



We see that up to 0.65 threshold node count is 1. If we look at the output count of the whole dataset we see -

value	occurance	occurance/total
acc	384	0.22
good	69	0.0399
unacc	1210	0.7002
vgood	65	0.037

We can see that unacc occurs more than 70% in the whole dataset. So, when we take 80% train set from the data, the percentage of unacc will still be high. So the algorithm returns a decision tree like this for threshold below 0.7

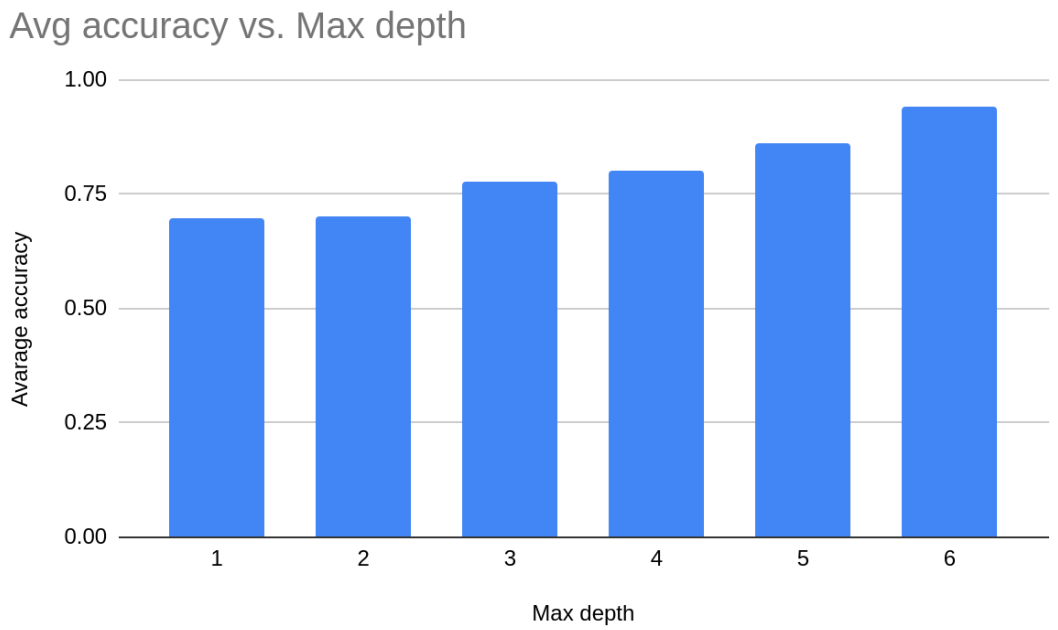


So, the decision tree returns ‘unacc’ for every input. This is why performance is poor for threshold below 0.7

We tried pruning in the hope of reducing overfitting and gaining better result. But we do not find any significant improvement in the average accuracy.

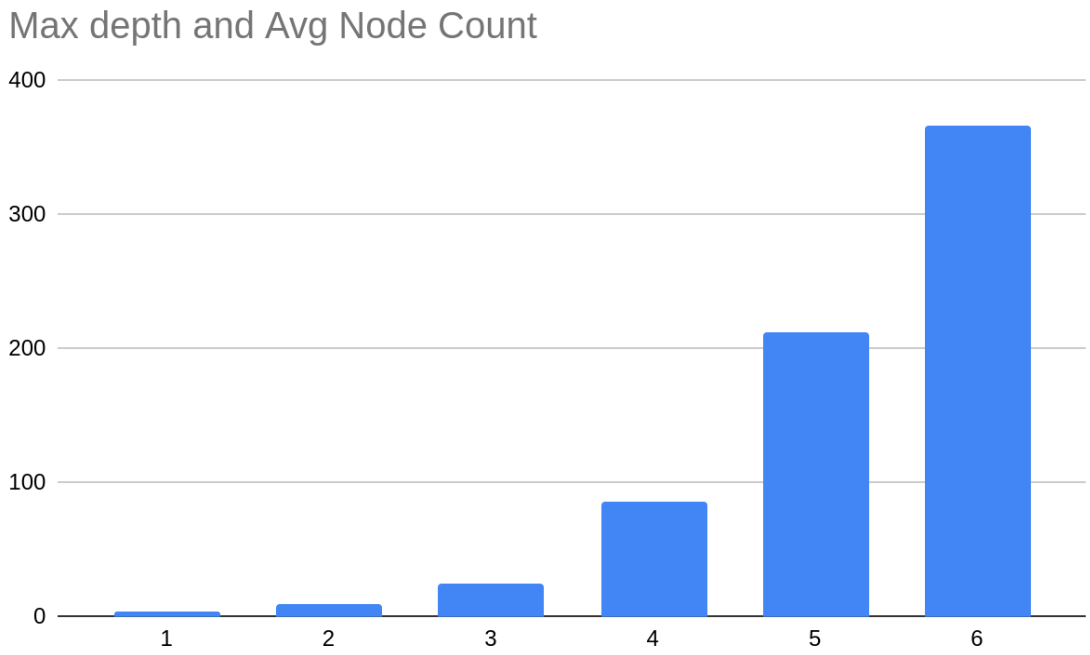
Max Depth

We tried to limit depth during tree construction in case it gets very large.



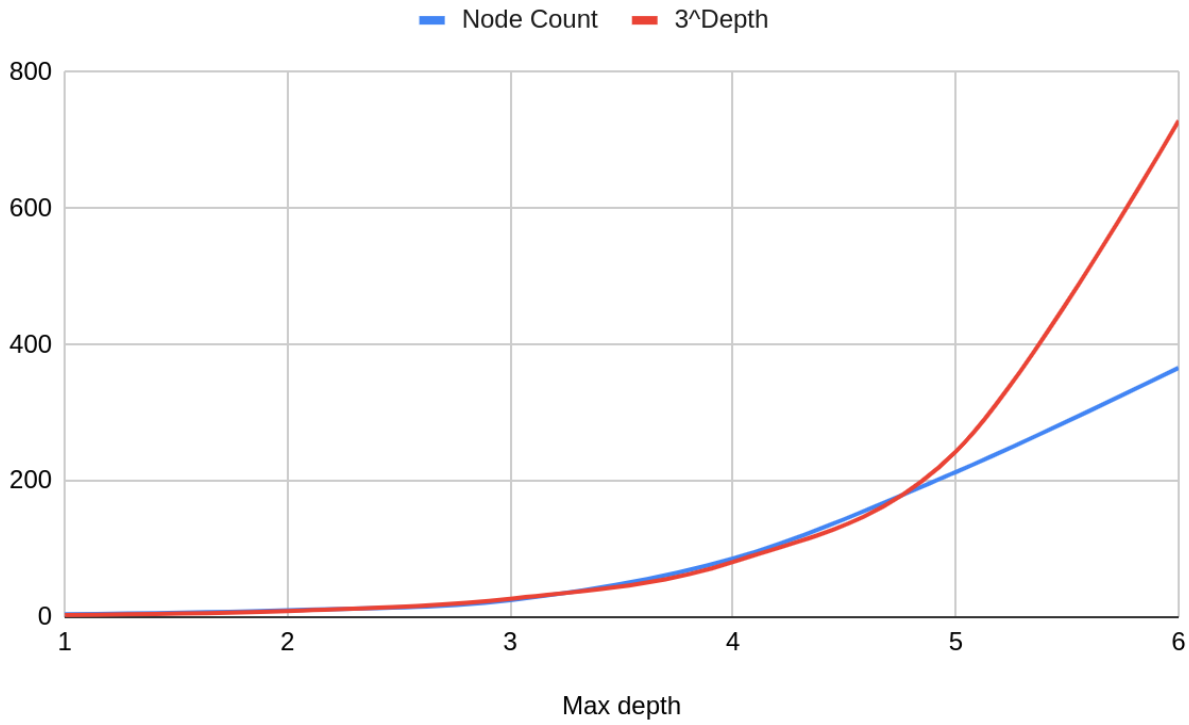
We can see that average accuracy increases with depth.

Our reason for limiting max depth is to stop the tree before it gets too large and we run out of memory.



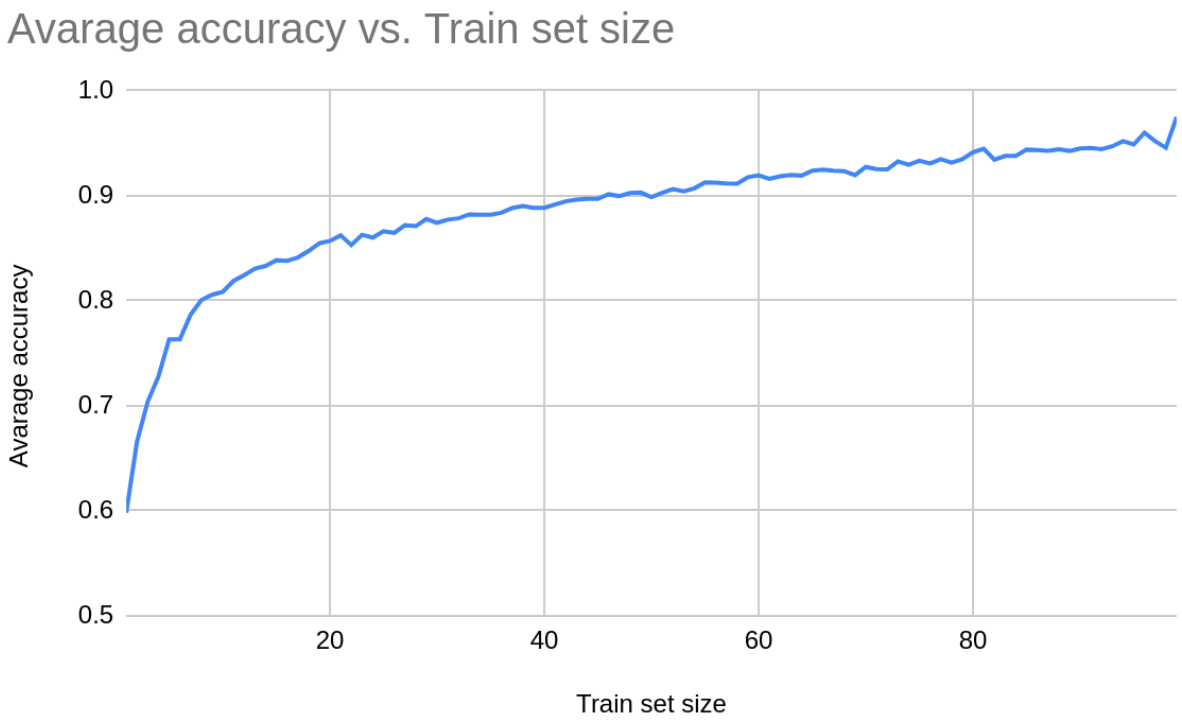
It looks exponential, But it is not. In out dataset, three of the attributes have 4 distinct values and three of them has 3 distinct values. In the tree was exponential then it would need at least 3^{depth} nodes. But if we compare 3^{depth} and average node count we can see that our tree does not grow exponentially.

```
--class values

unacc, acc, good, vgood

--attributes

buying:   vhigh, high, med, low.
maint:    vhigh, high, med, low.
doors:    2, 3, 4, 5more.
persons:  2, 4, more.
lug_boot: small, med, big.
safety:   low, med, high.

```



Learning Curve

We also varied our train set size and measured the average accuracy.



Attachments:

1. Experiment Results : Car Evaluation Decision Tree
2. Source Code: CSE 318 Offline 4
3. Best performing tree: Tree.txt