Information About Dataset

The dataset I've used for this project is called "Car Evaluation Dataset". It is one of the famous (most viewed) datasets in UC Irvine Machine Learning Repository. I encountered with this dataset in the UCI ML Repository website but couldn't download the dataset in csv format on that website. Hence, later on I visited Kaggle -which is one of the famous websites for machine learning and data enthusiastics- and found the same dataset. Luckily I could have downloaded the dataset in csv format which is easy to read and manage using Python modules.

Our dataset simply evaluates cars based on different attributes. Before talking about attributes I want to briefly mention what "evaluation means. Evaluation for this case is basically assigning scores to cars based on their attributes. Yet, these "scores" are not numerical values. Instead this dataset uses categorical target attribute where target may be "unacceptable", "acceptable", "good" and "very good". We will be using abbreviations for all of the attributes throughout this report and implementation, hence in other words **we will categorize (evaluate) each car in the set of ("unacc", "acc", "good", "vgood")**.

Finally we can talk about attributes that will be used while evaluating each car. We can list attributes and their possible values as such :

- **Buying price** ("bp") : This attribute is simply the buying price of the car. It can take 4 different values which are "low", "med", "high" and "vhigh"

- **Price of the maintenance** ("maint") : This attribute represents the price needed to pay to cover the maintenance of the car. It can take 4 different values which are "low", "med", "high" and "vhigh".

- **Number of doors** ("doors") : This attributes corresponds to number of doors that car have. It can take 4 different values which are "2" , "3" , "4" and "5more", where "5more" means 5 or more number of doors.

- **Capacity in terms of persons to carry** ("persons") : This attributes simply the capacity of the car in the unit of people (persons per car). It can take 3 different values which are "2", "4" and "more" where more means 5 or more number of people.

- **The size of luggage boot** ("lug_boot") : This attributes represents the size of the luggage boot of the car and it can take 3 different values which are "small", "med" and "big".

- **Estimated safety of the car** ("safety") : This attributes gives idea about the safetiness that car provides to customers and it can take 3 different values which are "low", "med", "high".

Detailed information can be found in below websites:

- https://archive.ics.uci.edu/dataset/19/car+evaluation
- https://www.kaggle.com/datasets/elikplim/car-evaluation-data-set

Use of 5-Fold Cross-Validation

Idea of k-fold cross-validations is basically using every data sample in two different duties which are “training” and “validation”. By using k-fold cross-validation we can estimate the performance of our decision tree more accurately compared to estimating based on a single performance (accuracy) score. Since usage of k-fold cross-validation is explained in slides and course textbook I will be mentioning about my implementation of 5-fold cross-validation.

I’ve basically divided the dataset into 5 partition of equal sizes. Then created five decision trees based on given training data. There will be five different training data since for each application of algorithm one of the partitions will be set aside as validation data (assuming test data is already set aside) and the other four partition will be considered as training data (one can simply observe there will be 5 different partition for validation data hence same for training data). After creating the decision trees and apply the learning algorithm on validation data I’ve computed the accuracy of each tree which I will be using as the performance measure of my algorithm. As the course slide suggests we can take the average of the evaluated scores (accuracies) in order to determine the error generated as the result of the algorithm (“1 – accuracy” will give the error). Via evaluating every tree individually, we can take the one that has the smallest validation error (highest accuracy) as the best tree generated.

$$E_{gen} \approx \langle E_{val} \rangle = \frac{1}{K} \sum_{k=1}^{K} E_{val}(k)$$

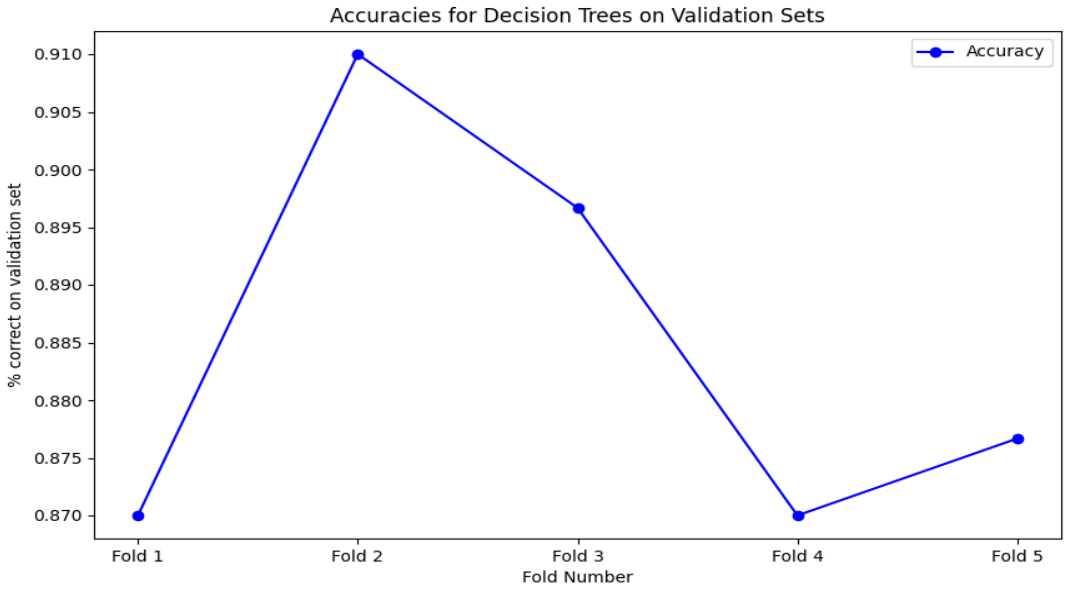
In order to enhance our model we may want to alter some of the parameters that our algorithm intrinsically use which are called hyperparameters. Some common examples can be given for these hyperparameters such as maximum depth of the decision tree, minimum number of sample to split a node or criterion that is used to measure the quality of split which is entropy in our case. In order to increase the performance of our decision tree algorithm we can use the partitioned validation set which is a subset of the dataset remaining after setting test set aside. Idea of using validation set is to increase the accuarcy of the decision tree using different hyperparameter while using the algorithm on this validation set. After setting up the most efficient (as possible) hyperparameters that are decided by using sample data on validation set, we can use the unseen data (test set) in order to determine the final performance of the decision tree.

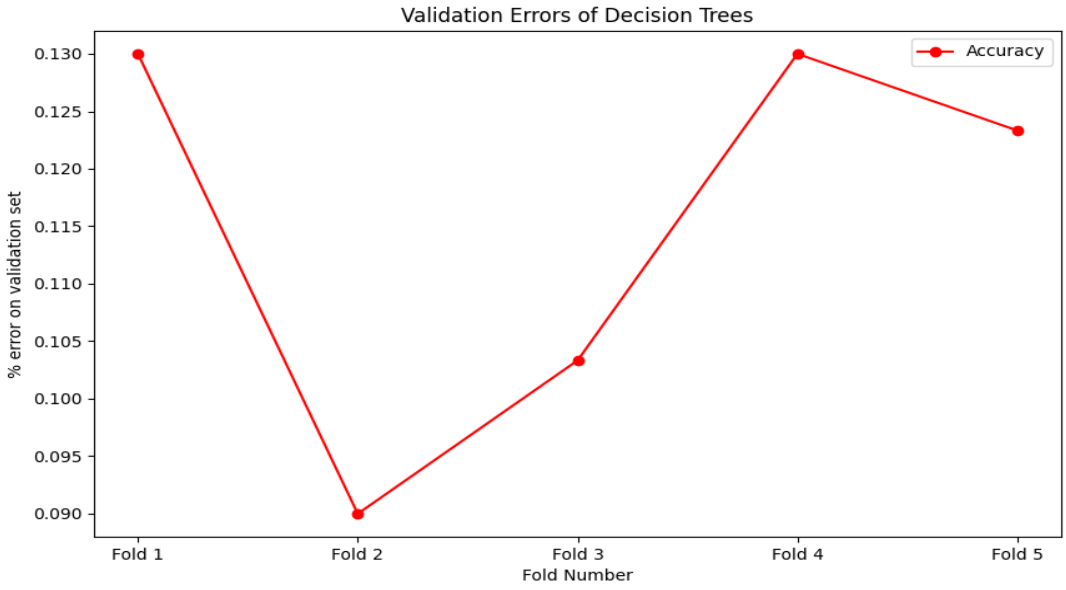
Nearly 15% of the data set is set aside as test set. Remaining dataset is partitioned into 5 subsets (folds). While applying 5-fold cross-validation each different fold is used as validation set while other 4 is used as training set. For each decision tree generated after training, validation error is calculated and the tree resulted in minimum validation error (max accuracy) is considered as the best/final tree.

Error Plots

Error Plot for Validation Datasets
For every different folds we will have different training set and possibly different decision tree. Below plot shows the accuracies of each decision tree algorithm on its corresponding validation set (fold).

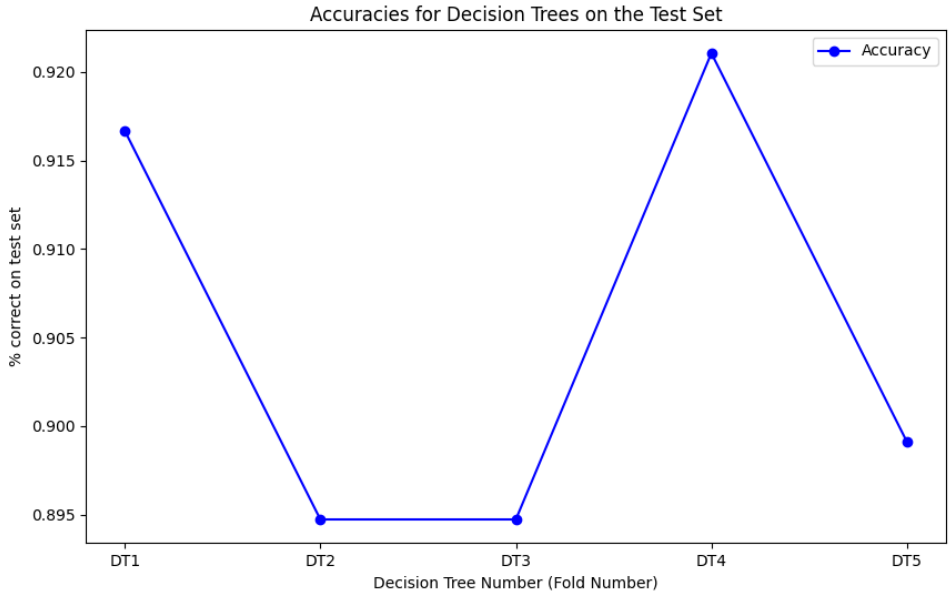
- Fold 1 : (Validation Set : Partition 1)
- Fold 2 : (Validation Set : Partition 2)
- Fold 3 : (Validation Set : Partition 3)
- Fold 4 : (Validation Set : Partition 4)
- Fold 5 : (Validation Set : Partition 5)

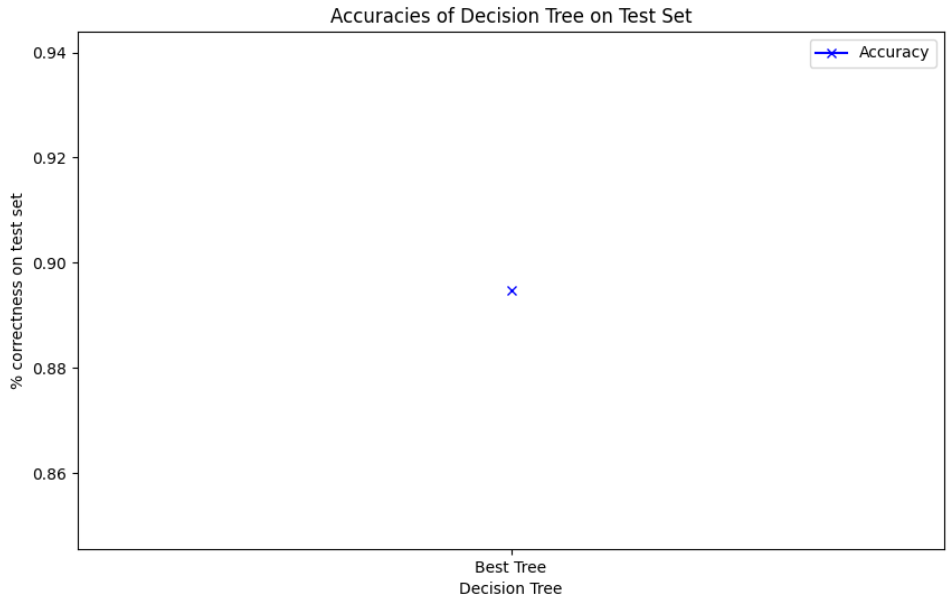




Error Plot for Test Set

After seperating test set aside from our dataset (since it will represent the unseen data for us) we can make the final performance measurement on our decision trees. We are using test set as our performance measurement since it represents the realistic data we can obtain from the outside world.





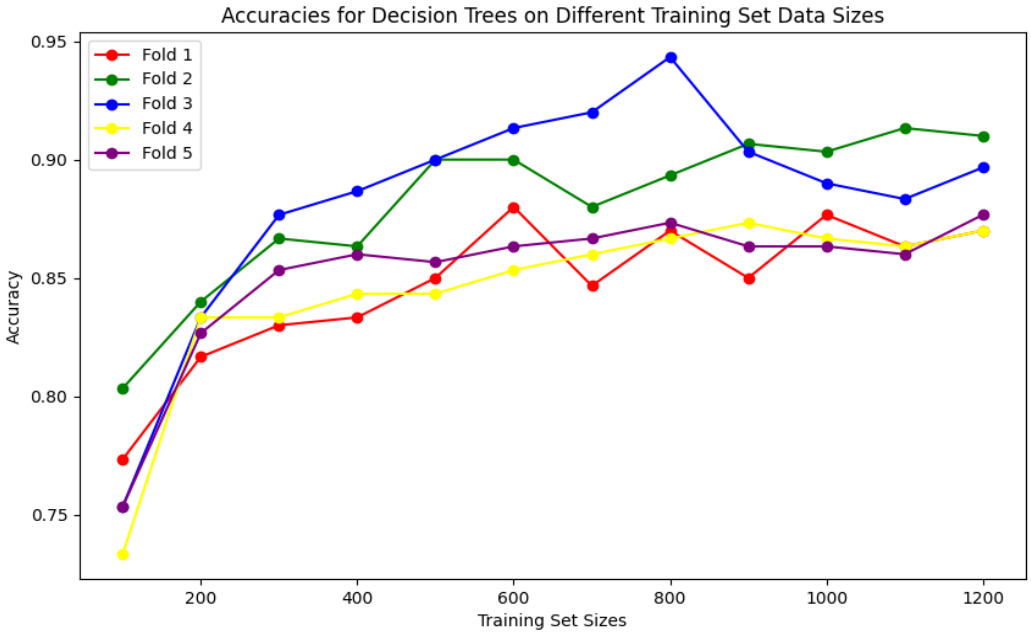
Error Plot for Training Sets

Since we are using the training set while generating our decision tree model, we are expecting to have 100% accuracy for each sample data we are providing from the training set. So as expected, below plot indicates the accuracy of the trained decision trees applied on training sets.

- X-axis is labeled as fold number, representing different folds assigned to validation sets. However, accuracy result is calculated on training set and as expected model works with 0% of error since the provided data is training data.

Additionally, we can plot the accuracy of each model with different training set sizes. Below plotting graph is generated for each decision tree of each fold. As expected, even though some exceptions are observed, as training set size increases accuracy of the decision tree for the same validation set also increases.

- Fold 1 (Validation Set: Partition 1, Training Set Sizes = [100,200,300,…,1200]
- Fold 2 (Validation Set: Partition 2, Training Set Sizes = [100,200,300,…,1200]
- Fold 3 (Validation Set: Partition 3, Training Set Sizes = [100,200,300,…,1200]
- Fold 4 (Validation Set: Partition 4, Training Set Sizes = [100,200,300,…,1200]
- Fold 5 (Validation Set: Partition 5, Training Set Sizes = [100,200,300,…,1200]



Final Decision Tree

After applying 5-fold cross validation on my dataset, I've obtained different accuracy results for my decision tree algorithm. As the final decision tree, I took the most accurate one which is the decision tree I've generated from my "Fold 2" where validation set is the second partition of my dataset. It has the accuracy of 0.91 which can be observed on plots above. Before sharing the final decision tree, I want to talk about the generated tree and nodes' specifications. Each node in the tree has a name which is constructed in a special way such that observer can deduce the splitting criterion and attributes by just looking at the node. Additionally, each node is assigned with an integer (incremental starts with 0) that enable us to differentiate the node with other nodes that has the same splitting attribute and attribute value. So each node has this format of name :

<splitting_attribute>_<attribute_value>_<node_id>

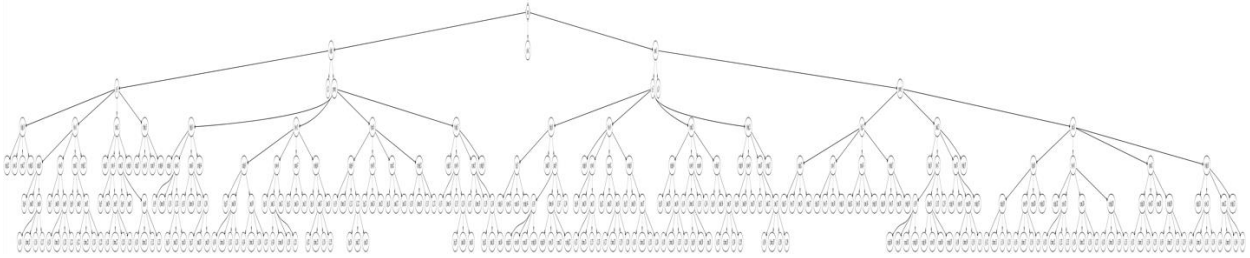
- Buying price ("b") : "low", "med", "high" and "vhigh"
- Price of the maintenance ("m") : "low", "med", "high" and "vhigh".
- Number of doors ("d") : "2" , "3" , "4" and "5more", where "5more"
- Capacity in terms of persons to carry ("p") : "2", "4" and "more"
- The size of luggage boot ("l") : "small", "med" and "big"
- Estimated safety of the car ("s") : "low", "med", "high"

Since tree is too wide to draw by hand even using drawing tools, I've used an open source graph drawing tool which is a replicated version of "Graphviz" which is also open source. Additionally,

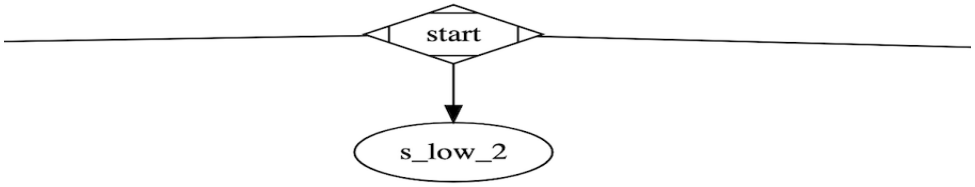
since the tree is too big to observe overall, you can download the png file I'm putting here and review it on your personal computer and observe the tree in more detail.

- Note : Leaf nodes on the tree are decision nodes.

Tree (Can download using Google Drive hyperlink : here)



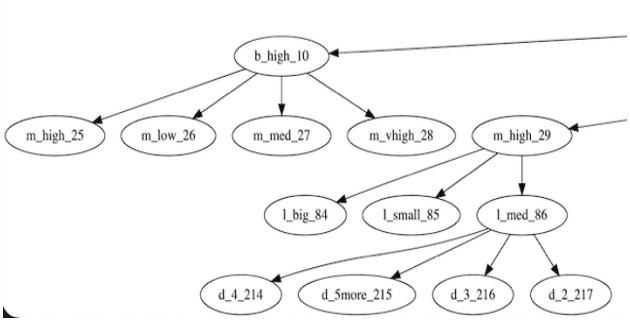
Parts of the tree to understand the workflow



- For given part of the tree:
  - s (safety) is selected as the splitting attribute because it has the smallest entropy value among all splitting attributes.
  - s_low_2 is a leaf node (it is assigned as leaf node since entropy is 0 no this node), hence contains a decision (which is "unacc" – by looking at the dataset for all cars if car is unsafe (safety : low), it is evaluated as unacceptable)

Other example part (just for visualization purposes) :

- For the subtree below, we can observe that, after splitting the data on buying price criterion ("b") it is decided to split further on price of maintenance ("m") criterion. For all maintenance criterions would be resulted as leaf nodes. (m_vhigh will include "unacc" decision with probability of 1 and all other m_x nodes will include "acc" decision with probability 1. Overall entropy = 0 for each of them) Path from root: s_high->p_4->b_high->m_x



Example screenshots from the dataset (for visualization purposes)

- Dataset is sorted (grouped) originally, I've shuffled the rows before starting to analyze. Random seed of the python program is set to a value in order to produce the same result on different runs. It can be adjusted accordingly if desired.

buying	maint	doors	persons	lug_boot	safety	class
vhigh	vhigh	2	2	small	low	unacc
vhigh	vhigh	2	2	small	med	unacc
vhigh	vhigh	2	2	small	high	unacc
vhigh	vhigh	2	2	med	low	unacc
vhigh	vhigh	2	2	med	med	unacc
vhigh	vhigh	2	2	med	high	unacc
vhigh	vhigh	2	2	big	low	unacc
vhigh	vhigh	2	2	big	med	unacc
vhigh	vhigh	2	2	big	high	unacc
vhigh	vhigh	2	4	small	low	unacc
vhigh	vhigh	2	4	small	med	unacc
low	high	2	4	small	med	acc
low	high	2	4	small	high	acc
low	high	2	4	med	low	unacc
low	high	2	4	med	med	acc
low	high	2	4	med	high	acc
low	high	2	4	big	low	unacc
low	high	2	4	big	med	acc
low	high	2	4	big	high	vgood
low	high	2	more	small	low	unacc
med	med	4	more	big	low	unacc
med	med	4	more	big	med	acc
med	med	4	more	big	high	vgood
med	med	5more	2	small	low	unacc
med	med	5more	2	small	med	unacc