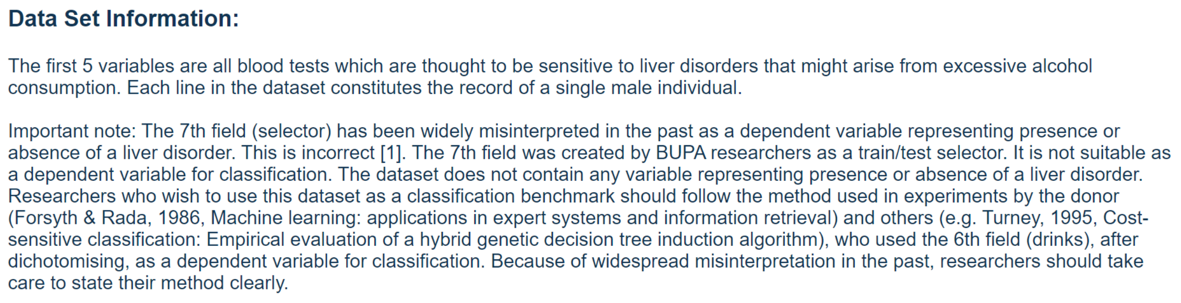
Assignment-1B
Group No- 14
Roll numbers- 19CS10037,19CS30049
Names- K Sai Santhoshi Niharika, Surkanti Akshitha

- We have used a standard ID3 algorithm for designing the decision tree.
- The data used in training was provided in .data format.
- The data set is related to liver disorders.
- We have shuffled the data and We considered the split of 60:20:20 as Train, Validation, and Test set respectively.
- The data had the following attributes:
    - mcv mean corpuscular volume
    - alkphos alkaline phosphotase
    - sgpt alanine aminotransferase
    - sgot aspartate aminotransferase
    - gammagt gamma-glutamyl transpeptidase
    - drinks number of half-pint equivalents of alcoholic beverages drunk per day
    - selector field created by the BUPA researchers to split the data into train/test sets
- We have modified the data set according to the data set information given to a new bupa.data file where the 6th field is a dependant variable for classification.



Data Set Information:

The first 5 variables are all blood tests which are thought to be sensitive to liver disorders that might arise from excessive alcohol consumption. Each line in the dataset constitutes the record of a single male individual.

Important note: The 7th field (selector) has been widely misinterpreted in the past as a dependent variable representing presence or absence of a liver disorder. This is incorrect [1]. The 7th field was created by BUPA researchers as a train/test selector. It is not suitable as a dependent variable for classification. The dataset does not contain any variable representing presence or absence of a liver disorder. Researchers who wish to use this dataset as a classification benchmark should follow the method used in experiments by the donor (Forsyth & Rada, 1986, Machine learning: applications in expert systems and information retrieval) and others (e.g. Turney, 1995, Cost-sensitive classification: Empirical evaluation of a hybrid genetic decision tree induction algorithm), who used the 6th field (drinks), after dichotomising, as a dependent variable for classification. Because of widespread misinterpretation in the past, researchers should take care to state their method clearly.

Major functions:

- Construct_tree methods for Gini and information gain: Uses the helper functions to build the Decision Tree. It is a recursive method, which chooses an attribute at each height to classify the current node into two children nodes. The attribute to be chosen is decided on the basis of the decrease in variance from the current node
- Good_attr for Gini and information gain: This function uses the above-mentioned equations and returns the attribute to be used at a given height to the construct_tree method. It uses the current data at a given node.
- 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 and MSE values.
- Randomize_select_best_tree for Gini and information gain: Provides the accuracy by averaging over 10 random 80/20 splits. The depth to be considered is provided as a parameter in construct_tree. Also returns that particular tree that provides the best test accuracy as the desired one.
- Print_decision_tree: Uses Digraph (from Graphviz package) to print the tree. Stores the output in “.gv” and “.pdf” format

- As the best test accuracy for both Gini and information gain is 100 we are printing both Gini and information gain decision trees.

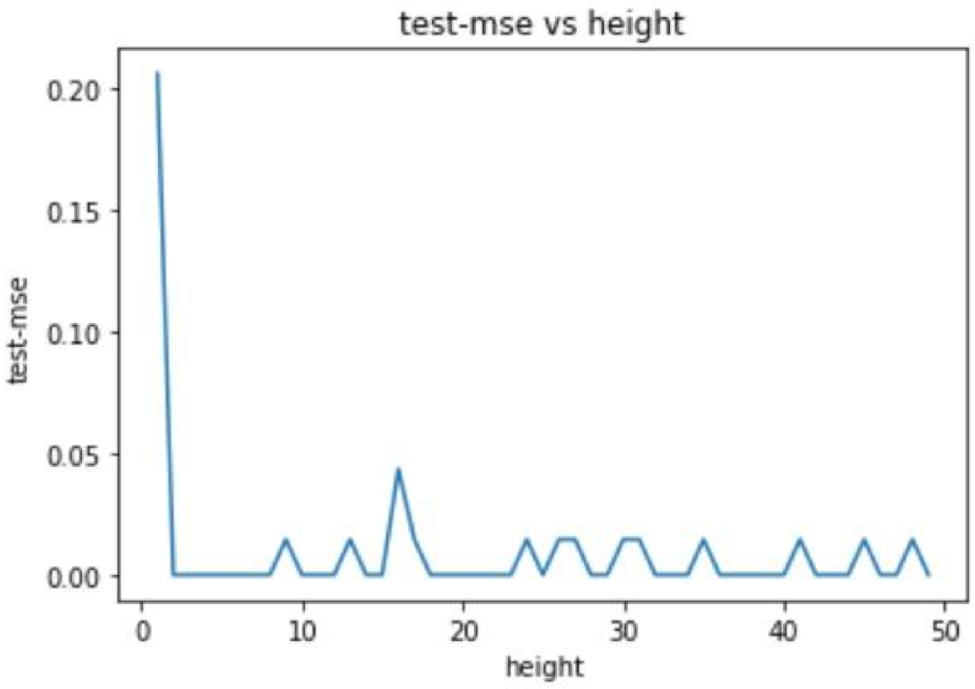
Helper functions:
- get_date: To covert date into float. Further also helps in better decision making and also allows us to consider the date as an attribute.
- build_data: Converts the DataFrame object obtained from read_csv to a collection of dictionaries. Each row is converted to a dictionary, with keys as attributes, and values as the corresponding entry in that row.
- train_test_split: Splits the data into three parts i.e, Train, Validation, and Test set in the ratio of 60:20:20.
- remove_children: Removes the children of the current node. Used in the pruning method.
- restore: Restores 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.
- variance: Calculates the variance of the provided data. Used by good_attr method.
- predict_one: Outputs the predicted value for a single data. It uses the attribute and threshold (split) value at a node and recursively moves from the root to a leaf. The predicted value is the mean value of that leaf node. Used by predict method.
- get_node: Returns the text to be printed at a particular node. Used by print_decision_tree method.
- 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.
- r2_score: Used to find accuracy.

Pruning:
- Steps:
    - We constructed the entire Decision Tree with the height provided in construct_tree. We take a bottom-up approach.
    - Pruning begins from the parent of leaf nodes. We consider a node. We temporarily remove the children of the node, say it New_Temporary_Tree. Evaluate the mean-squared error (MSE) of the New_Temporary_Tree on the validation set.
    - If the MSE 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 pruning approach is also termed reduced-error pruning. To summarize, we have used the train set for building the tree, validation set for pruning, and test set for estimating the final accuracy

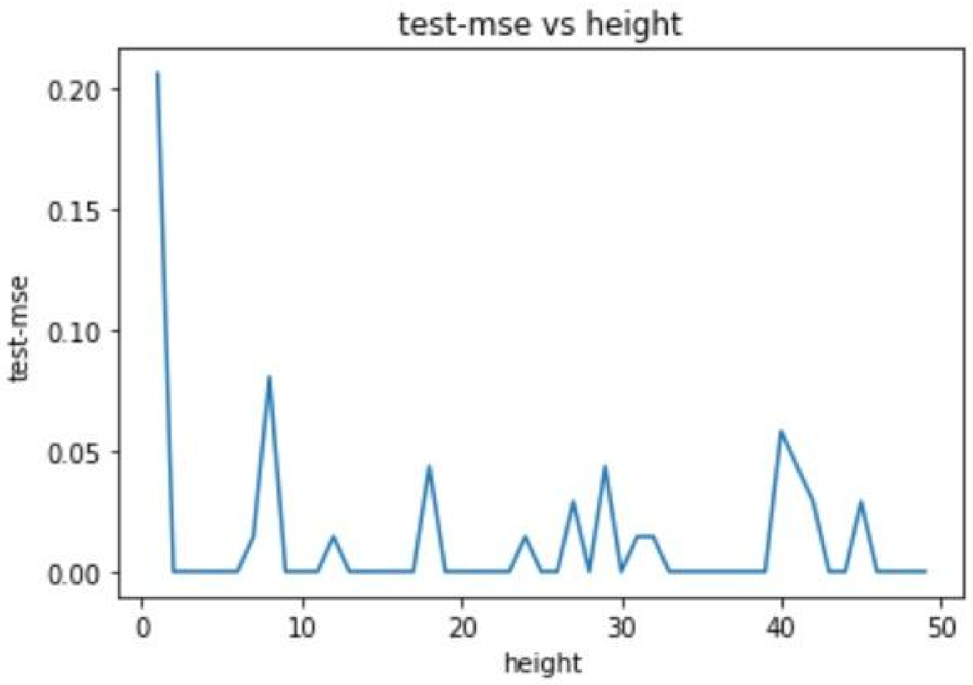
Results:

Before and after test-mse vs height

1. Gini index



2. Information gain



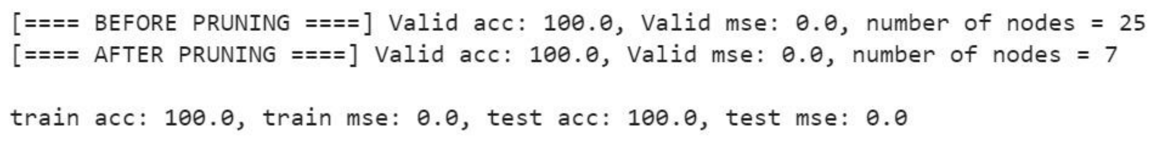
Puring results:

1. Gini index

```
[==== BEFORE PRUNING ====] Valid acc: 100.0, Valid mse: 0.0, number of nodes = 19
[==== AFTER PRUNING ====] Valid acc: 100.0, Valid mse: 0.0, number of nodes = 7

train acc: 100.0, train mse: 0.0, test acc: 100.0, test mse: 0.0
```

2. Information gain



```
[==== BEFORE PRUNING ====] Valid acc: 100.0, Valid mse: 0.0, number of nodes = 25
[==== AFTER PRUNING ====] Valid acc: 100.0, Valid mse: 0.0, number of nodes = 7

train acc: 100.0, train mse: 0.0, test acc: 100.0, test mse: 0.0
```

How to run the code :

- Download ipynb file and data set into your local machine
- Upload the above two downloaded files into the drive
- Open them in google collab from your drive
- Run every part in the notebook from the beginning to the end and the later parts will give you the desired results.
- After running the last parts you will observe two pdfs downloading in your machine i.e., a decision tree for Gini index and information gain as impurities.