# MACHINE LEARNING

## Assignment-1

## Group - 042

Chukka Jaswanth Kumar (20CS10021)

Sourabh S Das (20CS30051)

## CS60050

# DATA SET:

This dataset contains information about insurance payment of a customer based on certain measurements of different attributes. The dataset contains information about demographics (gender, age, region code type), Vehicles (Vehicle Age, Damage), Policy (Premium, sourcing channel). The dataset has the following attributes:

- **id** - Unique ID for the customer
- **Gender** -Gender of the customer
- **Age** - Age of the customer
- **Driving_license** - Customer's Driving license
- **Region_Code** - Unique code for the region of the customer
- **Previously_Insured** - Customer's Vehicle Insurance
- **Vehicle_Age** - Age of the Vehicle
- **Vehicle_Damage** - Customer's vehicle damage in the past
- **Annual_Premium** - The amount customer needs to pay as a premium in the year
- **PolicySalesChannel** - Anonymized Code for the channel of outreaching to the customer
- **Vintag**e - Number of days customer has been associated with the company
- **Response** - Customer is interested or not

The target function is a boolean-valued classification of whether or not the customer is interested in Vehicle insurance or not.

# Decision Tree

## Procedure:

We have used the **ID3 algorithm** for building the Decision tree. We have used the Information Gain measure for selecting the best parameter for splitting the data.
The data provided was initially modified such that the data columns having continuous values were discretized to almost **2** groups for easier tree building, testing and pruning. We have split the data **80%** for training and **20%** for testing. From the training data, we have stored **20%** for validation in case it is the best split, and use the remaining data for building the tree.

Out of 10 iterations, we take the model with the best test accuracy. Then we use the validation data set that was stored for Reduced Error Pruning. Then we plot the accuracy vs depth graph and print the tree in an hierarchical manner.

## Functions/Classes:

1. **Node class**: It is a class object for a node. It stores the boolean value of whether the node is a leaf or a non leaf. In either case, the best value is stored.
   In case of a non-leaf node, it also stores the attribute on the basis of which the split was made and the list of child nodes.

2. **ID3**: It is a function that uses the ID3 algorithm for building the tree. So, it makes a node leaf if it is pure with value **1** or **0** or if its list of attributes is exhausted, else it builds a tree for each node at that level recursively by finding the split by the **IG** measure.
   The data, best value, best split attribute, remaining attributes list, parent node - these are passed into the function for their recursive use.

3. **IG**: It is a function that takes in the data and the index of the attribute on the basis of which split will be made and returns the weighted entropy for that split by employing the Information Gain measure.

   **For $N$ classes (E = Entropy):** $E = -\sum_{i=1}^{n} p_i \log_2(p_i)$
   **Gain = E(parent) - $\sum$[E(child) $*$ weight of the child]**

   The split will be made on the feature with highest Gain.

4. **test:** It tests the data using the function check for all the testing dataset and compares with actual value and returns the accuracy.

5. **check**: It returns the best value on the basis of the node reached.

6. **DFS depth:** It recursively checks for the maximum depth of the tree.

7. **DFS_pruning**: It recursively applies the function for non-leaf child nodes. Then it removes the subtree and compares the accuracy before and after.
   Based on that it decides on whether to remove or keep the subtree. Thus it employs Reduced Error Pruning. It uses the previously stored validation data for this purpose.

8. **depth_acc _test:** It is quite similar to the test function, except it is used while calculating accuracy till particular depth.

9. **depth_acc _check:** It is quite similar to the check function, except it also assumes the nodes at the required depth to be leaf nodes. Thus the nodes at further depths are unvisited.

10. **print _tree:** It recursively prints tab spaces based on the depth and best value that occurs at that node and if it is non-leaf, it also prints the best attribute that was used for splitting.

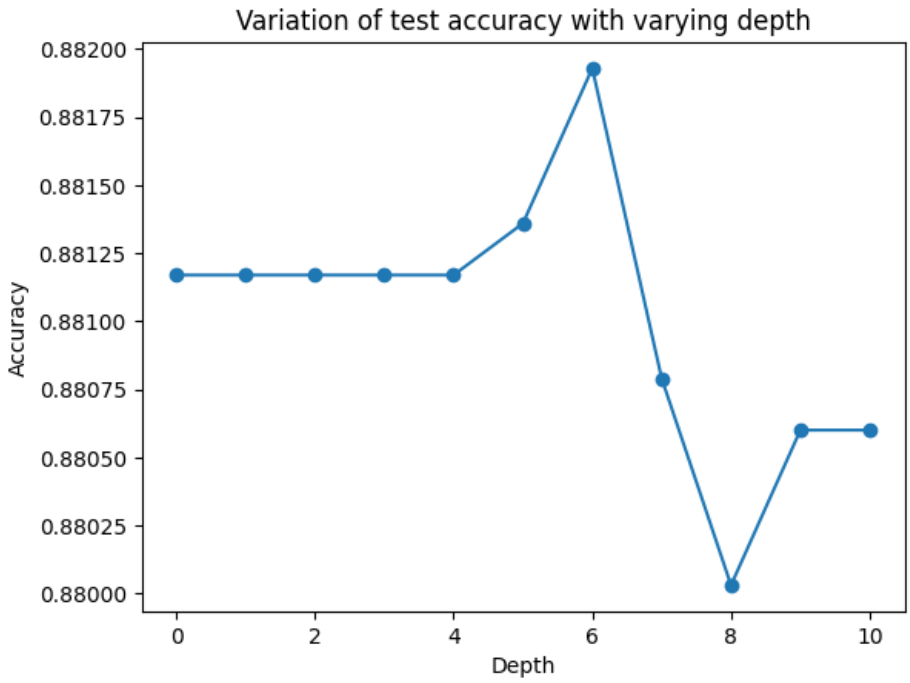
# Results:

## Before Pruning Accuracy

| Iteration No. | Accuracy |
|---|---|
| 1 | 87.22% |
| 2 | 87.26% |
| 3 | 87.64% |
| 4 | 87.45% |
| 5 | 87.93% |
| 6 | 87.68% |
| 7 | 87.21% |
| **8** | **88.06%** |
| 9 | 87.43% |
| 10 | 87.66% |

* Best accuracy obtained = **88.06%**
* Depth of the tree = **10**

## Pre-pruning Graph Plot:

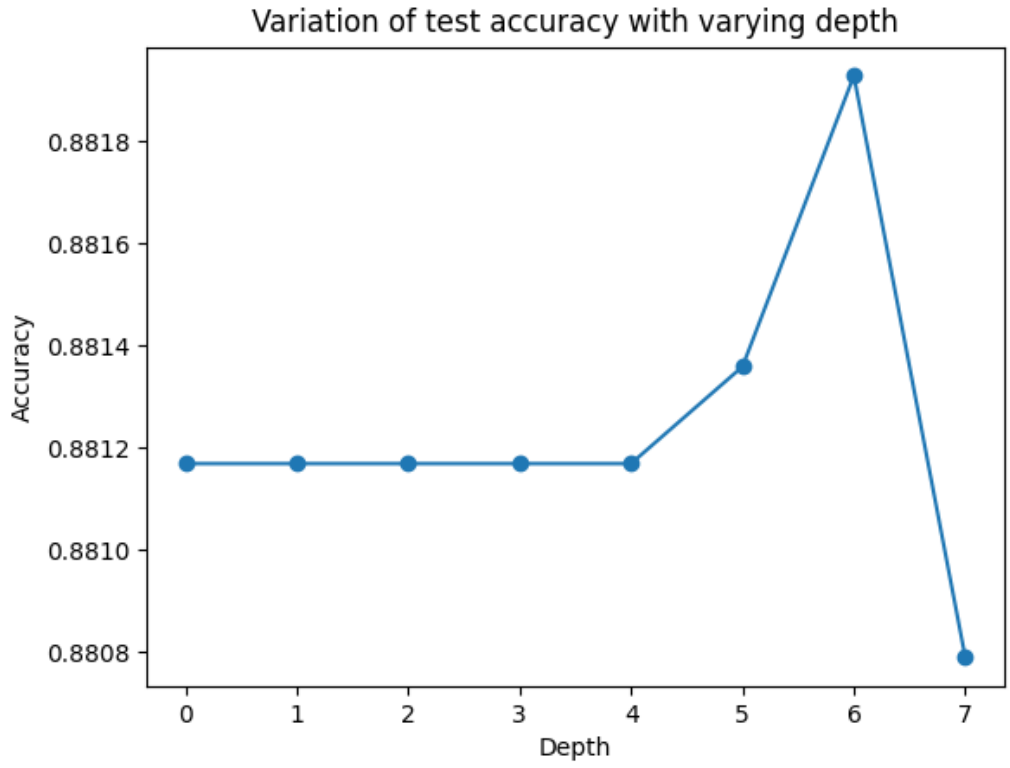


### After Pruning Accuracy v/s Depth

| Depth level | Accuracy |
|---|---|
| 0 | 88.12% |
| 1 | 88.12% |
| 2 | 88.12% |
| 3 | 88.12% |
| 4 | 88.12% |
| 5 | 88.14% |
| 6 | 88.19% |
| 7 | 88.08% |

* Clearly, the depth of the tree has reduced from **10** to **7**, thus implying, the nodes which made the model less accurate were pruned off.

### Graph Plot after Pruning:



# Naive Bayesian Classifier

## Procedure:

We have made a Naive Bayes classifier. The data provided was initially modified such that the data columns having continuous values were discretized to almost $10$ groups and the categorical variables were encoded.
The sample with max outliers were found and removed. Then it was split into $80\%$ for training and $20\%$ for testing.
Now, the training data was split into $10$ parts ($10$ fold cross validation). In each iteration the relevant split bucket was used for testing (validation) and the remaining $9$ parts were used for training. The model with the best accuracy was selected. It was tested on the test split to get the accuracy. Then the model was trained using Laplacian correction as well.The test accuracy was found after all this process.

## Functions used:

1. **train:** It takes in the training data and returns 3 lists of data. The first list is the list of dictionaries of probabilities of occurrence of all unique values in a column given that the Response is **1** (**P(**$X_i$**|1)**).
   The second list is the list of dictionaries of probabilities of occurrence of all unique values in a column given that the Response is **0** (**P(**$X_i$**|0)**).
   The third list is the list of dictionaries of probabilities of occurrence of all unique values in a column (**P(**$X_i$**)**). The unique values are the keys for the dictionaries.

2. **test:** It takes in the test data, the list of dictionaries of probabilities of positive values, negative values and the unique values. Based on the provided data, it calculates the probabilities of the result being $1$ or $0$. The value having the max probability is considered as the predicted value. Then it is compared with the actual value to give the final accuracy.
   The probabilities of the results are calculated using the formula :

   $$\mathbf{P(}Y_i \mid X_1, X_2, \ldots, X_n\mathbf{)} = \frac{\mathbf{P(}X_1, X_2, \ldots, X_n \mid Y_i\mathbf{)} * \mathbf{P(}Y_i\mathbf{)}}{\mathbf{P(}X_1, X_2, \ldots, X_n\mathbf{)}}$$

   Now since the variables $X_i$ are independent of each other, we can simply multiply their individual probabilities.

3. **train_laplace:** This function is the same as the original train function, except it also handles the probabilities which are **0**.

   $$\mathbf{P(}X_i \mid Y_i\mathbf{)} = \frac{\textbf{count of samples satisfying both } X_i \textbf{ and } Y_i + \alpha}{\textbf{count of samples satisfying } Y_i + \mathbf{K}\alpha}$$

   Here $\alpha$ is generally considered to be **1** and **K** is the number of unique values that can occur for that parameter.

## Results:

| Iteration No. | Accuracy Obtained on validation set |
|---|---|
| 1 | 87.96% |
| 2 | 87.21% |
| **3** | **88.40%** |
| 4 | 87.62% |
| 5 | 88.00% |
| 6 | 87.87% |
| 7 | 87.69% |
| 8 | 87.59% |
| 9 | 86.96% |
| 10 | 87.98% |

* The model in the third iteration was selected.

* The **accuracy** obtained on the test set for this model is : **87.95%**

* Then the model was trained using Laplacian correction and the **accuracy** obtained on the test set for this model is : **87.95%**

Now, since the accuracies obtained by both models with or without Laplacian correction is the same, this means there is negligible amount of data that gives **0** probability while training without Laplacian correction.