DECISION TREE CLASSIFICATIO N ALGORITHM

DECISION TREE CLASSIFICATION ALGORITHM

Supervised learning technique used for **classification** and regression

Tree internal nodes: features of the dataset

Branches: decision rules

Leaf nodes: outcomes

Two types of nodes

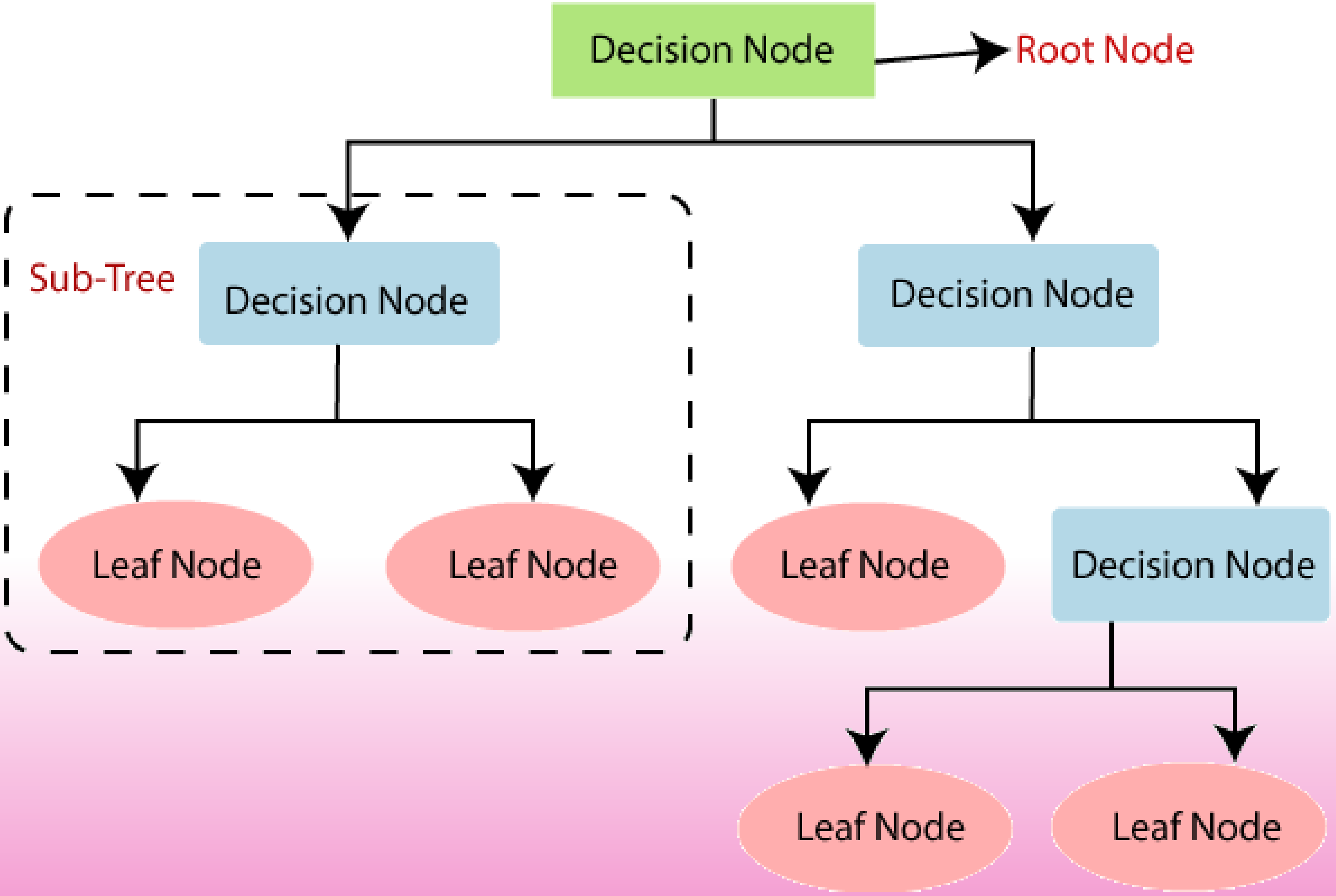
- Decision nodes : make decision and have multiple branches
- Leaf nodes : output of those decisions

DECISION TREE CLASSIFICATION ALGORITHM

- The decisions or the test are performed based on features of the given dataset.
- It is a graphical representation for getting all the possible solutions to a problem/decision based on given conditions.
- To build tree: CART (Classification and Regression Tree Algorithm)
- Tree asks a question based on “yes or no” further split into subtrees

DECISION TREE CLASSIFICATION ALGORITHM

A decision tree can contain categorical data (YES/NO) as well as numeric data.


Decision Node
Root Node
Sub-Tree
Decision Node
Decision Node
Leaf Node
Leaf Node
Leaf Node
Decision Node
Leaf Node
Leaf Node

REASONS TO USE DECISION TREE

Mimic human thinking ability while making decisions

Easy to understand

Logic is easy to understand due to tree structure

DECISION TREE TERMINOLOGIES

Root Node: Root node is from where the decision tree starts. It represents the entire dataset, which further gets divided into two or more homogeneous sets.

Leaf Node: Leaf nodes are the final output node, and the tree cannot be segregated further after getting a leaf node.

Splitting: Splitting is the process of dividing the decision node/root node into sub-nodes according to the given conditions.

DECISION TREE TERMINOLOGIES

Branch/Sub Tree: A tree formed by splitting the tree.

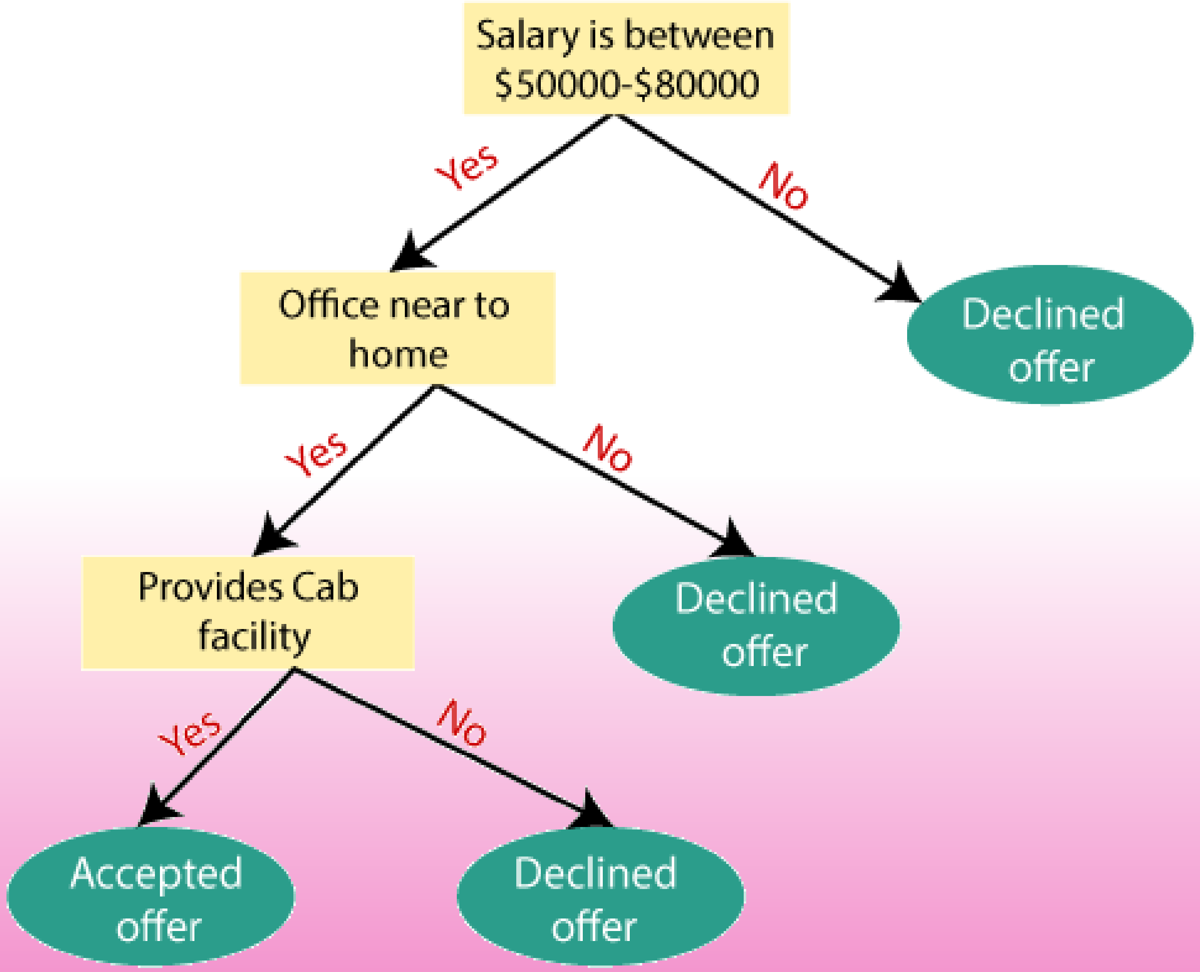
Pruning: Pruning is the process of removing the unwanted branches from the tree.

Parent/Child node: The root node of the tree is called the parent node, and other nodes are called the child nodes.

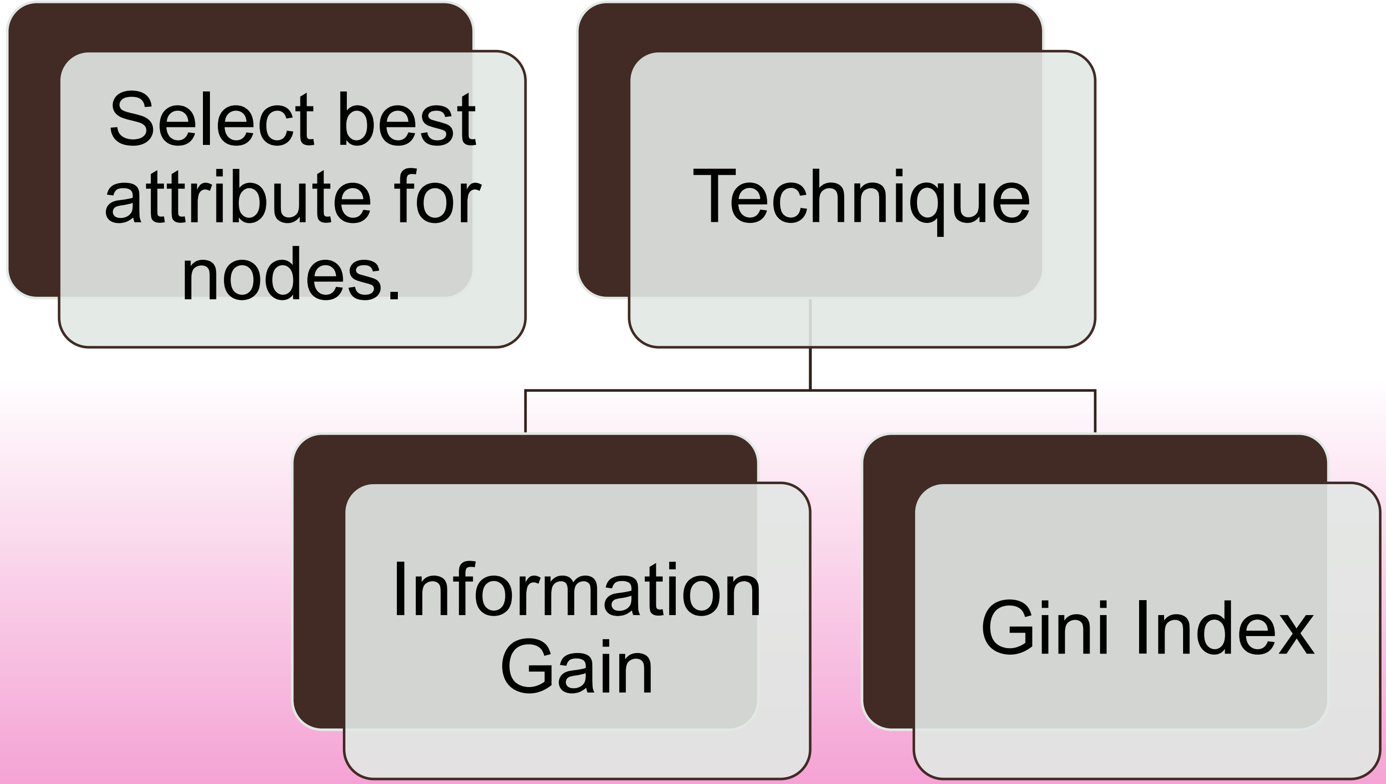
WORKING – COMPARE AT ROOT AND JUMP TO NEXT NODE BASED ON COMPARISON

- **Step-1:** Begin the tree with the root node, says S, which contains the complete dataset.
- **Step-2:** Find the best attribute in the dataset using **Attribute Selection Measure (ASM).**
- **Step-3:** Divide the S into subsets that contains possible values for the best attributes.
- **Step-4:** Generate the decision tree node, which contains the best attribute.
- **Step-5:** Recursively make new decision trees using the subsets of the dataset created in step -3. Continue this process until a stage is reached where you cannot further classify the nodes and called the final node as a leaf node.

EXAMPLE – ACCEPT JOB OFFER OR DECLINE



ATTRIBUTE SELECTION MEASURES (ASM)



INFORMATION GAIN

Information gain is the measurement of changes in entropy after the segmentation of a dataset based on an attribute.

It calculates how much information a feature provides us about a class.

According to the value of information gain, we split the node and build the decision tree.

A decision tree algorithm always tries to maximize the value of information gain, and a node/attribute having the highest information gain is split first.

INFORMATION GAIN

- Information Gain= Entropy(S) -
  [(Weighted Avg) *Entropy(each feature)
- Entropy: Entropy is a metric to measure the impurity in a given attribute.
- Entropy(s)= -P(yes)log2 P(yes)- P(no) log2 P(no)
- **S= Total number of samples**
- **P(yes)= probability of yes**
- **P(no)= probability of no**

GINI INDEX

Gini index is a measure of impurity or purity used while creating a decision tree in the CART(Classification and Regression Tree) algorithm.

An attribute with the low Gini index should be preferred as compared to the high Gini index.

It only creates binary splits, and the CART algorithm uses the Gini index to create binary splits.

GINI INDEX - FORMULA

Gini Index= 1- $\sum_j P_j^2$

PRUNING

- *Pruning is a process of deleting the unnecessary nodes from a tree in order to get the optimal decision tree.*
- A too-large tree increases the risk of overfitting
- If a small tree may not capture all the important features of the dataset.
- So decreasing the learning tree size without reducing accuracy is known as Pruning.
- **Cost Complexity Pruning**
- **Reduced Error Pruning.**

ADVANTAGES OF DECISION TREE

It is simple to understand as it follows the same process which a human follow while making any decision in real-life.

It can be very useful for solving decision-related problems.

It helps to think about all the possible outcomes for a problem.

There is less requirement of data cleaning compared to other algorithms.

The decision tree contains lots of layers, which makes it complex.

It may have an overfitting issue, which can be resolved using the **Random Forest algorithm.**

For more class labels, the computational complexity of the decision tree may increase.

DISADVANTAGES

PYTHON IMPLEMENTATION

- Data Pre-processing step
- Fitting a Decision-Tree algorithm to the Training set
- Predicting the test result
- Test accuracy of the result(Creation of Confusion matrix)
- Visualizing the test set result

#feature Scaling
from sklearn.preprocessing import StandardScaler
st_x= StandardScaler()
x_train= st_x.fit_transform(x_train)
x_test= st_x.transform(x_test)

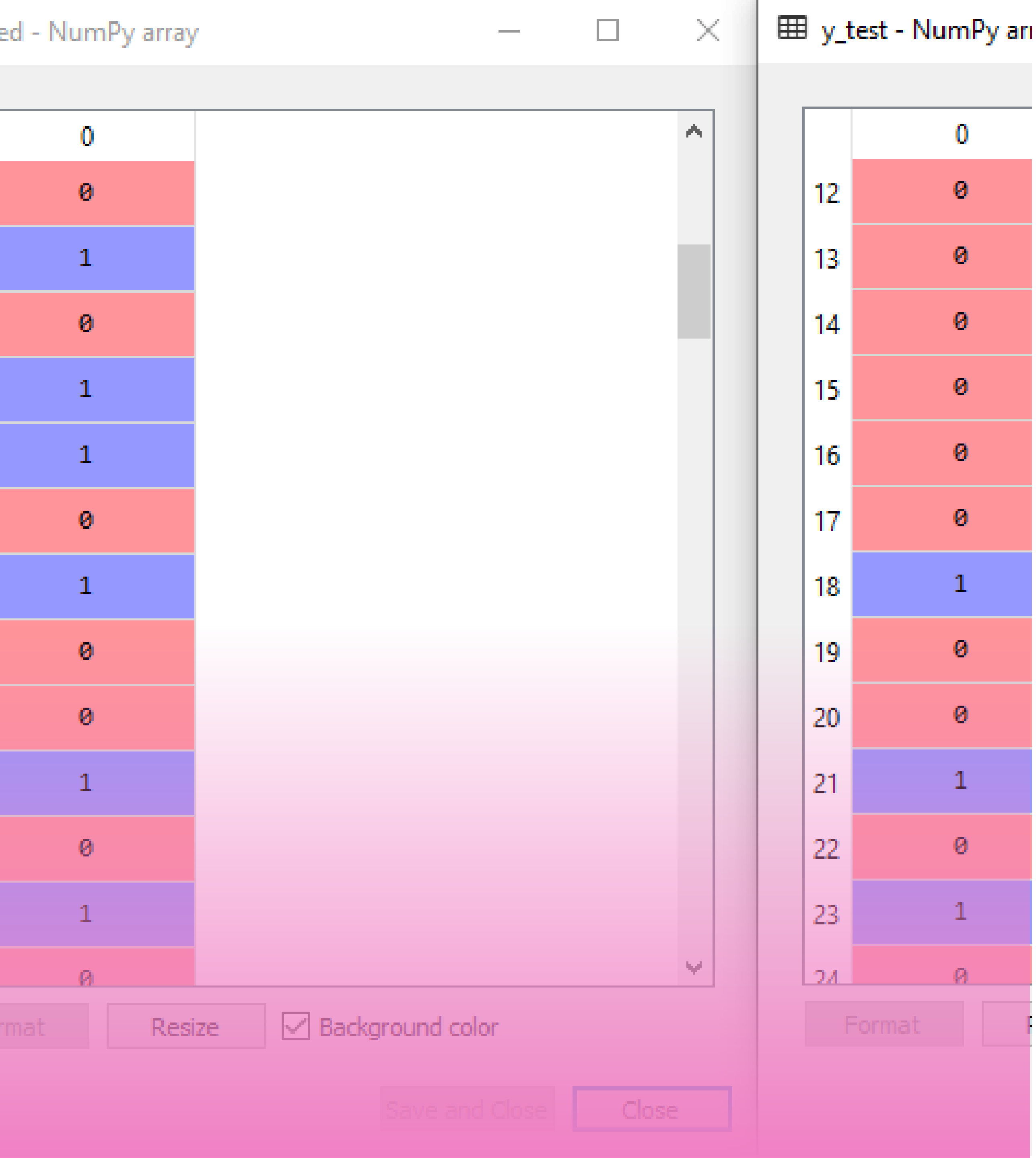
data_set - DataFrame

Index	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
5	15728773	Male	27	58000	0
6	15598044	Female	27	84000	0
7	15694829	Female	32	150000	1
8	15600575	Male	25	33000	0
9	15727311	Female	35	65000	0
10	15570769	Female	26	80000	0
11	15606274	Female	26	52000	0
12	15746139	Male	20	86000	0
13	15704987	Male	32	18000	0
14	15628972	Male	18	82000	0
15	15697686	Male	29	80000	0

Format Resize Background color Column min/max Save and Close Close

PYTHON IMPLEMENTATION

- From sklearn.tree import DecisionTreeClassifier
- classifier= DecisionTreeClassifier(criterion='entropy', random_state=0)
- classifier.fit(x_train, y_train)



PYTHON IMPLEMENTATION

```
#Predicting the test set result
y_pred= classifier.predict(x_test)
```

PYTHON IMPLEMENTATION

- #Creating the Confusion matrix

from sklearn.metrics import confusion_matrix

cm= confusion_matrix(y_test, y_pred)

cm - NumPy array

	0	1
0	62	6
1	3	29

		Actual	
		Dog	Not Dog
Predicted	Dog	True Positive (TP)	False Positive (FP)
	Not Dog	False Negative (FN)	True Negative (TN)

PYTHON IMPLEMENTATION

- #Creating the Confusion matrix

```
from sklearn.metrics import confusion_matrix

cm= confusion_matrix(y_test, y_pred)
```

cm - NumPy array

	0	1
0	62	6
1	3	29