IMPLEMENTATION OF DECISION TREE USING ID3 ALGORITHM

In [5]:
```
# Implementation of decision tree classifier
# Importing the required packages
import numpy as np
import pandas as pd
from sklearn.metrics import confusion_matrix
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import export_graphviz
from sklearn.metrics import accuracy_score
from sklearn.metrics import classification_report
```

In [6]:
```
def importdata():
    data = pd.read_csv('Sample_Data.csv')
    # Printing the dataset shape
    print ("Dataset Length: ", len(data))
    print ("Dataset Shape: ", data.shape)
    # Printing the dataset obseravtions
    print ("Dataset: ",data.head())
    return data
```

In [7]:
```
# Function to split the dataset
def splitdataset(data):
    # Separating the target variable
    X = data.iloc[:, 1:3].values
    Y = data.iloc[:, 4].values
    # Splitting the dataset into train and test
    X_train, X_test, y_train, y_test = train_test_split(
    X, Y, test_size = 0.25, random_state = 100)
    return X, Y, X_train, X_test, y_train, y_test
```

In [8]:
```
# Function to perform training with giniIndex.
def train_using_gini(X_train, X_test, y_train):
    # Creating the classifier object
    clf_gini = DecisionTreeClassifier(criterion = "gini",

    random_state = 100,max_depth=4, min_samples_leaf=2)

    # Performing training
    clf_gini.fit(X_train, y_train)
    return clf_gini
```

In [9]:
```
# Function to perform training with entropy.
def train_using_entropy(X_train, X_test, y_train):
    # Decision tree with entropy
    clf_entropy = DecisionTreeClassifier(

    criterion = "entropy", random_state = 100,
    max_depth = 4, min_samples_leaf = 2)

    # Performing training
    clf_entropy.fit(X_train, y_train)
    return clf_entropy
```

In [10]:
```
# Function to make predictions
def prediction(X_test, clf_object):
    # Predicton on test with giniIndex
    y_pred = clf_object.predict(X_test)
    print("Predicted values:")
    print(y_pred)
    return y_pred
```

In [11]:
```
# Function to calculate accuracy
def cal_accuracy(y_test, y_pred):
    print("Confusion Matrix: ",
    confusion_matrix(y_test, y_pred))
    print ("Accuracy : ",
    accuracy_score(y_test,y_pred)*100)
    print("Report : ",
    classification_report(y_test, y_pred))
```

In [12]:
```
# Driver code
def main():
    # Building Phase
    data = importdata()
    ## data.info()
    data['Gender'],_ = pd.factorize(data['Gender'])
    data['PurchasedTheProduct'],class_names = pd.factorize(data['PurchasedTheProduct'])
    ## data.info()
    Xf = data.iloc[:,1:-1]
    X, Y, X_train, X_test, y_train, y_test = splitdataset(data)
    clf_gini = train_using_gini(X_train, X_test, y_train)
    clf_entropy = train_using_entropy(X_train, X_test, y_train)

    # Operational Phase
    print("Results Using Gini Index:")
    # Prediction using gini
    y_pred_gini = prediction(X_test, clf_gini)
    cal_accuracy(y_test, y_pred_gini)
    print("Results Using Entropy:")
    # Prediction using entropy
    y_pred_entropy = prediction(X_test, clf_entropy)
    cal_accuracy(y_test, y_pred_entropy)
```

In [13]:
```
# Calling main function
if __name__=="__main__":
    main()
```

```
Dataset Length:  400
Dataset Shape:  (400, 5)
Dataset:       User ID  Gender  Age  Salary  PurchasedTheProduct
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
Results Using Gini Index:
Predicted values:
[0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0
 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0]
Confusion Matrix:  [[62  3]
 [12 23]]
Accuracy :  85.0
Report :                precision    recall  f1-score   support

           0       0.84      0.95      0.89        65
           1       0.88      0.66      0.75        35

    accuracy                           0.85       100
   macro avg       0.86      0.81      0.82       100
weighted avg       0.85      0.85      0.84       100

Results Using Entropy:
Predicted values:
[0 1 0 0 0 1 0 0 0 0 0 0 0 0 1 1 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 1 1 1 1 1 0
 0 0 0 0 0 0 1 1 0 0 0 0 1 0 0 0 0 1 0 1 1 0 1 0 1 0 0 0 0 1 0 0 0 0 1 1 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 0 0 1 0 1 0 0 0 0 0]
Confusion Matrix:  [[62  3]
 [12 23]]
Accuracy :  85.0
Report :                precision    recall  f1-score   support

           0       0.84      0.95      0.89        65
           1       0.88      0.66      0.75        35

    accuracy                           0.85       100
   macro avg       0.86      0.81      0.82       100
weighted avg       0.85      0.85      0.84       100
```

In [ ]: