In [ ]:

```
# EX NO:8          Perform Analaysis on Decision Tree Classifier     Bharath Kumar S
# Date:11/03/24                                                      URK22AI1030
```

AIM: The aim of this project is to develop a Decision Tree classification model for the Social_Network dataset using scikit-learn. The model will predict whether an individual will purchase a product or not based on their gender, age, and estimated salary. The project will involve encoding categorical values, dividing the data into training and testing sets, performing classification with entropy and information gain, and analyzing the performance of the classifier using various performance measures. Additionally, the constructed decision tree will be displayed.

DESCRIPTION: In this project, we will build a Decision Tree classification model using scikit-learn for the Social_Network dataset. The dataset contains information about individuals including their gender, age, estimated salary, and whether they purchased a particular product or not.

To begin, we will use the columns 'Gender', 'Age', and 'EstimatedSalary' as the independent variables, and the target variable will be 'Purchased'. The categorical values of the target column ('Yes-Y' and 'No-N') will be encoded into numerical values.

Next, we will divide the data into training and testing sets, with 75% of the data used for training and 25% for testing.

We will perform the classification using both entropy and information gain as criteria to construct the decision tree.

To evaluate the performance of the classifier, we will analyze various performance measures such as the confusion matrix, accuracy, recall, precision, specificity, f-score, and Receiver Operating Characteristic (ROC) curve. These measures will help us assess the effectiveness of the model in predicting whether an individual will make a purchase or not.

Finally, we will display the constructed decision tree, which will provide insights into the rules and conditions used by the model to make predictions.

In [ ]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.tree import DecisionTreeClassifier
from sklearn.preprocessing import LabelEncoder
from sklearn import metrics
from sklearn import tree
import matplotlib.pyplot as plt
```

In [ ]:

```
# Load the dataset
data = pd.read_csv('/Social_Network (1).csv')
```

In [ ]:

```
# Select the independent variables and the target variable
X = data[['Gender', 'Age', 'EstimatedSalary']]
y = data['Purchased']
```

In [ ]:

```
# Encode the categorical target variable to numerical values
label_encoder = LabelEncoder()
y = label_encoder.fit_transform(y)
```

In [ ]:

```
# Divide the data into training and testing sets
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_state=42)
```

In [ ]:

```
# Create the Decision Tree classifier using entropy
classifier = DecisionTreeClassifier(criterion='entropy')
```

```
classifier.fit(X_train, y_train)
```

Out[ ]:

```
▾ DecisionTreeClassifier
DecisionTreeClassifier(criterion='entropy')
```

In [ ]:

```
# Make predictions on the test set
y_pred = classifier.predict(X_test)
```

In [ ]:

```
# Calculate various performance measures
confusion_matrix = metrics.confusion_matrix(y_test, y_pred)
accuracy = metrics.accuracy_score(y_test, y_pred)
recall = metrics.recall_score(y_test, y_pred)
precision = metrics.precision_score(y_test, y_pred)
specificity = metrics.recall_score(y_test, y_pred, pos_label=0)
f1_score = metrics.f1_score(y_test, y_pred)
```

In [ ]:

```
# Display the performance measures
print("Confusion Matrix:")
print(confusion_matrix)
print("Accuracy:", accuracy)
print("Recall:", recall)
print("Precision:", precision)
print("Specificity:", specificity)
print("F1 Score:", f1_score)
```

```
Confusion Matrix:
[[59  8]
 [ 4 29]]
Accuracy: 0.88
Recall: 0.8787878787878788
Precision: 0.7837837837837838
Specificity: 0.8805970149253731
F1 Score: 0.8285714285714285
```

In [ ]:

```
# Plot the Decision Tree
plt.figure(figsize=(12, 12))
tree.plot_tree(classifier, feature_names=X.columns, class_names=label_encoder.classes_, fille
plt.show()
```