\\## **Building Decision Tree using sklearn**

Data set : users.csv

Step 1: Loading the data

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
#reading data from csv file
user=pd.read_csv('/content/drive/MyDrive/Subjects/ Python_AI_ML_DL_NLP/3  Machine Learning/2. Supervised Machine Learning/2_Classifi
user.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 99 entries, 0 to 98
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   User ID          99 non-null     int64
 1   Gender           99 non-null     object
 2   Age              99 non-null     int64
 3   EstimatedSalary  99 non-null     int64
 4   Purchased        99 non-null     int64
dtypes: int64(4), object(1)
memory usage: 4.0+ KB
```

```
user.head()
```

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	12	19000	0
1	15810944	Male	11	20000	0
2	15668575	Female	1	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	10	76000	0

```
user['Gender'].unique()
```

```
array(['Male', 'Female'], dtype=object)
```

Setting the predictors and target:

```
#selecting predicator attributes
X=user.columns.drop("Purchased")
#selecting target attribute
y=user['Purchased']
```

Step 2: Feature engineering - Encoding

```
# Encoding all the predictor variables to convert the categorical values to numerical values.
user_encoded = pd.get_dummies(user[X])
print("Total number of predictors after encoding = ", len(user_encoded.columns))
# Printing the list of columns after encoding to understand the encoding process
user_encoded.columns
user_encoded
```

```
Total number of predictors after encoding =  5
```

	User ID	Age	EstimatedSalary	Gender_Female	Gender_Male
0	15624510	12	19000	0	1
1	15810944	11	20000	0	1
2	15668575	1	43000	1	0
3	15603246	27	57000	1	0
4	15804002	10	76000	0	1
...	...	...	...	...	...
94	15786993	45	83000	1	0
95	15709441	35	44000	1	0
96	15710257	4	25000	1	0
97	15582492	5	123000	0	1
98	15575694	35	73000	0	1

99 rows x 5 columns

Step 3: Splitting the dataset into train and test data

```
#Impotbthe required function
from sklearn.model_selection import train_test_split
#splitting data into train and test datasets
X_train,X_test,y_train,y_test = train_test_split(user_encoded, y, test_size=0.15, random_state=0)
# Printing the shape of the resulting datasets
print("Shape of X_train and y_train are:", X_train.shape, "and", y_train.shape, " respectively")
print("Shape of X_test and y_test are:", X_test.shape, "and", y_test.shape, " respectively")

```

```
Shape of X_train and y_train are: (84, 5) and (84,)  respectively
Shape of X_test and y_test are: (15, 5) and (15,)  respectively
```

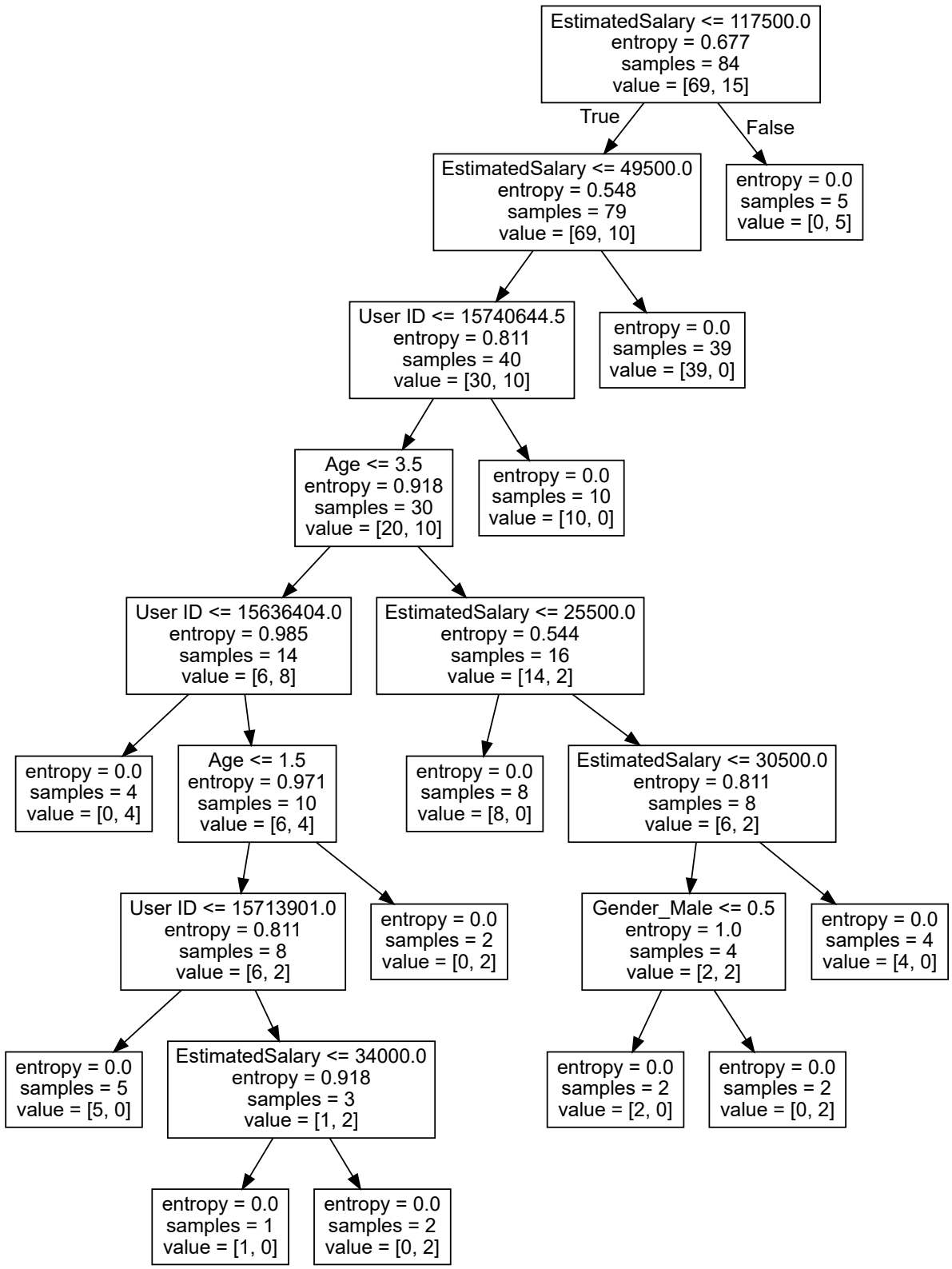
Step 4: Building the model

```
#from scipy.sparse import random
#Importing required class
from sklearn.tree import DecisionTreeClassifier
#create object
model=DecisionTreeClassifier(criterion='entropy',random_state=1)
model.fit(X_train,y_train)
```

```
DecisionTreeClassifier(criterion='entropy', random_state=1)
```

```
# Predicting target values using the model built on training data
train_predictions = model.predict(X_train)
test_predictions = model.predict(X_test)
```

```
# Importing the required libraries (Ensure that they are already installed.)
from sklearn.tree import export_graphviz
import graphviz
# Generating the tree
data = export_graphviz(model, out_file=None,
                       feature_names=user_encoded.columns)
graph = graphviz.Source(data)
graph
```



Step 5: Evaluate model performance on train and test sets

```
# Getting the accuracy on train data
train_accuracy = model.score(X_train,y_train)
print("Accuracy of the model on train data = ",train_accuracy)
# Getting the accuracy on test data
test_accuracy = model.score(X_test,y_test)
print("Accuracy of the model on test data = ",test_accuracy)
```

```
Accuracy of the model on train data =  1.0
Accuracy of the model on test data =  0.8
```

You can observe that the training accuracy is 100% and the test accuracy is approximately 80%.

This could mean that the model is overfit to the training data and is not a good approximation of the input to output mapping.