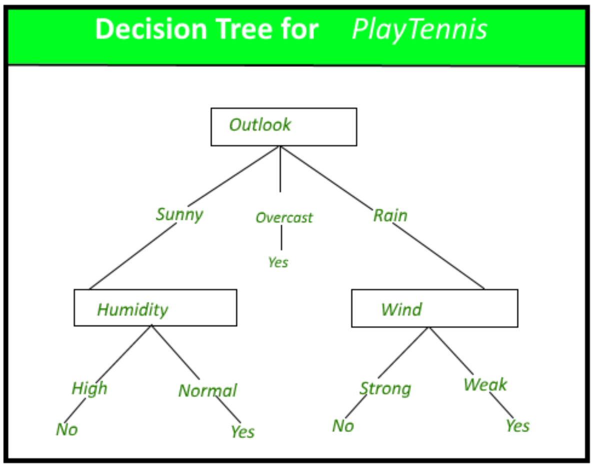
Experiment No. 3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:07/08/23
Date of Submission:20/08/23

Aim: Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model.

Objective: To perform various feature engineering tasks, apply Decision Tree Algorithm on the given dataset and maximize the accuracy, Precision, Recall, F1 score. Improve the performance by performing different data engineering and feature engineering tasks.

Theory:

Decision Tree is the most powerful and popular tool for classification and prediction. A Decision tree is a flowchart-like tree structure, where each internal node denotes a test on an attribute, each branch represents an outcome of the test, and each leaf node (terminal node) holds a class label.



Dataset:

Predict whether income exceeds $50K/yr based on census data. Also known as "Adult" dataset.

Attribute Information:

Listing of attributes:

>50K, <=50K.

age: continuous.

workclass: Private, Self-emp-not-inc, Self-emp-inc, Federal-gov, Local-gov, State-gov, Without-pay, Never-worked.

fnlwgt: continuous.

education: Bachelors, Some-college, 11th, HS-grad, Prof-school, Assoc-acdm, Assoc-voc, 9th, 7th-8th, 12th, Masters, 1st-4th, 10th, Doctorate, 5th-6th, Preschool.

education-num: continuous.

marital-status: Married-civ-spouse, Divorced, Never-married, Separated, Widowed, Married-spouse-absent, Married-AF-spouse.

occupation: Tech-support, Craft-repair, Other-service, Sales, Exec-managerial, Prof-specialty, Handlers-cleaners, Machine-op-inspct, Adm-clerical, Farming-fishing, Transport-moving, Priv-house-serv, Protective-serv, Armed-Forces.

relationship: Wife, Own-child, Husband, Not-in-family, Other-relative, Unmarried.

race: White, Asian-Pac-Islander, Amer-Indian-Eskimo, Other, Black.

sex: Female, Male.

capital-gain: continuous.

capital-loss: continuous.

hours-per-week: continuous.

native-country: United-States, Cambodia, England, Puerto-Rico, Canada, Germany, Outlying-US(Guam-USVI-etc), India, Japan, Greece, South, China, Cuba, Iran, Honduras, Philippines, Italy, Poland, Jamaica, Vietnam, Mexico, Portugal, Ireland, France, Dominican-Republic, Laos, Ecuador, Taiwan, Haiti, Columbia, Hungary, Guatemala, Nicaragua, Scotland, Thailand, Yugoslavia, El-Salvador, Trinadad&Tobago, Peru, Hong, Holand-Netherlands.

Code:

Conclusion:

1. **Dealing with Categorical Attributes during Data Pre-processing**:

   Managing Categorical Attributes During Data Pre-processing:

   The values of the following categories columns are converted into unique number labels using label encoding. This numerical representation is required by many machine learning methods that need numerical input data. Encoding these categorical characteristics prepares the data for machine learning model training.

   1. The term "workclass" refers to their job position.
   2. The word "education" indicates their greatest degree of schooling.
   3. "marital-status" indicates their marital status.
   4. "occupation" displays their work functions.
   5. "relationship" describes their familial situation.
   6. "race" usually refers to their racial heritage.
   7. The word "sex" denotes gender.
   8. "native-country" frequently refers to their country of origin or citizenship. Certain columns are eliminated during data pre-processing in the code you gave.

2. The following columns are specifically removed:

1. Channel: This column is removed with the data.drop(labels=(['Channel','Region']),axis=1,inplace=True) function. The Channel column appears to have been deleted from the dataset.

2. Region: The Region column, like the Channel column, is discarded using the same line of code. This column is also deleted from the dataset. **Hyperparameter Tuning:**

The Decision Tree classifier is hyperparameter tuned in this code:

The Decision Tree classifier is built with a maximum depth of 5 specified: DecisionTreeClassifier(max_depth=5). Because it influences the depth of the tree, this is a type of hyperparameter adjustment.This code sample, however, does not show a comprehensive hyperparameter tweaking procedure. In reality, more extensive approaches such as grid search or random search can be used to find the optimum hyperparameters. Only the max_depth is changed here.

3. Evaluation Metrics for Classification Models Confusion Matrix:

It accurately identified 4310 cases as negative (0) and 767 instances as positive (1), but it also projected 243 positives and 713 negatives incorrectly. Performance Metrics: The accuracy for positive predictions (1) is 0.76 lower than in Model 1, while the recall is 0.52 higher. This model has an F1-score of 0.62. This model has an overall accuracy of 0.84.

	precision	recall	f1-score	support
0	0.86	0.95	0.90	4553
1	0.76	0.52	0.62	1480
accuracy			0.84	6033
macro avg	0.81	0.73	0.76	6033
weighted avg	0.83	0.84	0.83	6033

Importing lib

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
```

Load the dataset

```
df = pd.read_csv('adult.csv')
df.head(10)
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	rela
0	90	?	77053	HS-grad	9	Widowed	?	No
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	No
2	66	?	186061	Some-college	10	Widowed	?	U
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	U
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	U
6	38	Private	150601	10th	6	Separated	Adm-	U

Understanding Dataset

```
print ("Total Rows     : " ,df.shape[0])
dataset_row = df.shape[0]
print ("Total Columns  : " ,df.shape[1])
print ("\nFeatures : \n" ,df.columns.tolist())
print ("\nMissing values :  ", df.isnull().sum().values.sum())
print ("\nUnique values :  \n",df.nunique())
```

```
Total Rows     :  32561
Total Columns  :  15

Features :
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num', 'marital.status', 'occupation', 'relationship', 'race', 'sex', 'capita

Missing values :   0

Unique values :
 age                  73
workclass             9
fnlwgt            21648
education            16
education.num        16
marital.status        7
occupation           15
relationship          6
race                  5
sex                   2
capital.gain        119
capital.loss         92
hours.per.week       94
native.country       42
income                2
dtype: int64
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 32561 entries, 0 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             32561 non-null  int64
 1   workclass       32561 non-null  object
 2   fnlwgt          32561 non-null  int64
 3   education       32561 non-null  object
 4   education.num   32561 non-null  int64
```

```
 5   marital.status  32561 non-null  object
 6   occupation      32561 non-null  object
 7   relationship    32561 non-null  object
 8   race            32561 non-null  object
 9   sex             32561 non-null  object
 10  capital.gain    32561 non-null  int64
 11  capital.loss    32561 non-null  int64
 12  hours.per.week  32561 non-null  int64
 13  native.country  32561 non-null  object
 14  income          32561 non-null  object
dtypes: int64(6), object(9)
memory usage: 3.7+ MB
```

```
df.describe()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32561.000000
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	40.437456
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	12.347429
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	1.000000
25%	28.000000	1.178270e+05	9.000000	0.000000	0.000000	40.000000
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	40.000000
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	45.000000
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	99.000000

Missing Values

```
df_missing = (df=='?').sum()
print(df_missing)
```

```
age                  0
workclass         1836
fnlwgt               0
education            0
education.num        0
marital.status       0
occupation        1843
relationship         0
race                 0
sex                  0
capital.gain         0
capital.loss         0
hours.per.week       0
native.country     583
income               0
dtype: int64
```

percent_missing = (df=='?').sum() * 100/len(df) percent_missing

```
#droping row having missing values from dataset
df = df[df['workclass'] !='?']
df = df[df['occupation'] !='?']
df = df[df['native.country'] !='?']
df.head()
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	cap
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	

```
df_missing = (df=='?').sum()
print(df_missing)
```

```
age               0
workclass         0
fnlwgt            0
education         0
education.num     0
marital.status    0
```

```
occupation        0
relationship      0
race              0
sex               0
capital.gain      0
capital.loss      0
hours.per.week    0
native.country    0
income            0
dtype: int64
```

```
print ("Total Rows after droping rows : " ,df.shape[0])
print("Numbers of rows drop: ", dataset_row -df.shape[0])
```

```
Total Rows after droping rows :  30162
Numbers of rows drop:  2399
```

Data Preparation

```
from sklearn import preprocessing
```

```
df_categorical = df.select_dtypes(include=['object'])
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	nati
1	Private	HS-grad	Widowed	Exec-managerial	Not-in-family	White	Female	U
3	Private	7th-8th	Divorced	Machine-op-inspct	Unmarried	White	Female	U
4	Private	Some-college	Separated	Prof-specialty	Own-child	White	Female	U

```
le = preprocessing.LabelEncoder()
df_categorical = df_categorical.apply(le.fit_transform)
df_categorical.head()
```

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	2	11	6	3	1	4	0	38	0
3	2	5	0	6	4	4	0	38	0
4	2	15	5	9	3	4	0	38	0
5	2	11	0	7	4	4	0	38	0
6	2	0	5	0	4	4	1	38	0

```
df = df.drop(df_categorical.columns,axis=1)
df = pd.concat([df,df_categorical],axis=1)
df['income'] = df['income'].astype('category')
df.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation	relation
1	82	132870	9	0	4356	18	2	11	6	3	
3	54	140359	4	0	3900	40	2	5	0	6	
4	41	264663	10	0	3900	40	2	15	5	9	
5	34	216864	9	0	3770	45	2	11	0	7	
6	38	150601	6	0	3770	40	2	0	5	0	

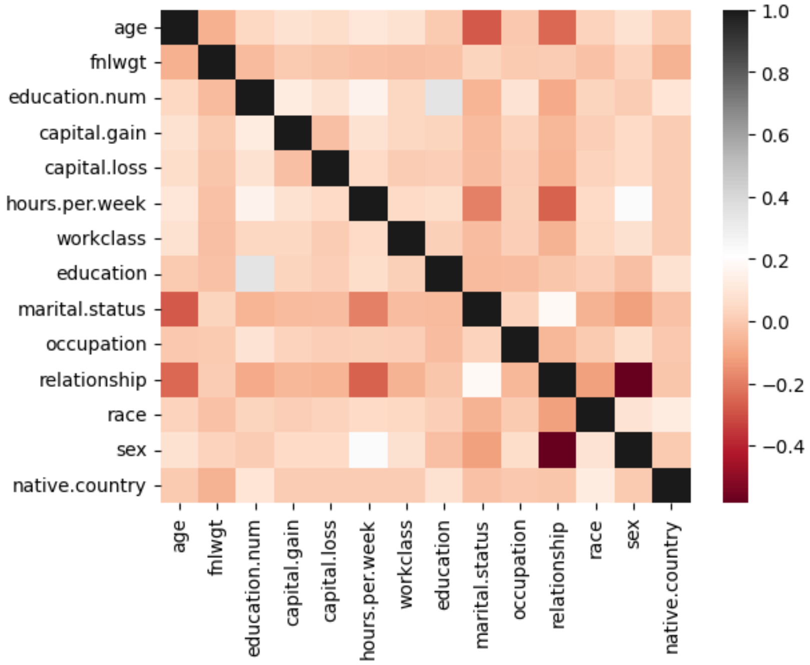
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 30162 entries, 1 to 32560
Data columns (total 15 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   age             30162 non-null  int64
 1   fnlwgt          30162 non-null  int64
 2   education.num   30162 non-null  int64
 3   capital.gain    30162 non-null  int64
 4   capital.loss    30162 non-null  int64
 5   hours.per.week  30162 non-null  int64
 6   workclass       30162 non-null  int64
```

```
 7   education       30162 non-null  int64
 8   marital.status  30162 non-null  int64
 9   occupation      30162 non-null  int64
 10  relationship    30162 non-null  int64
 11  race            30162 non-null  int64
 12  sex             30162 non-null  int64
 13  native.country  30162 non-null  int64
 14  income          30162 non-null  category
dtypes: category(1), int64(14)
memory usage: 3.5 MB
```

```
sns.heatmap(df.corr(), cmap = 'RdGy')
```

```
<ipython-input-101-b22fcbbd6ef9>:1: FutureWarning: The default value of numeric_only in DataFrame.corr is deprecated. In a future ve
  sns.heatmap(df.corr(), cmap = 'RdGy')
<Axes: >
```



Spliting dataset

```
from sklearn.model_selection import train_test_split
```

```
X = df.drop('income',axis=1)
X = X.drop('sex',axis=1)
y = df['income']
```

```
X.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation	relatio
1	82	132870	9	0	4356	18	2	11	6	3	
3	54	140359	4	0	3900	40	2	5	0	6	
4	41	264663	10	0	3900	40	2	15	5	9	
5	34	216864	9	0	3770	45	2	11	0	7	
6	38	150601	6	0	3770	40	2	0	5	0	

```
y.head()
```

```
1    0
3    0
4    0
5    0
6    0
Name: income, dtype: category
Categories (2, int64): [0, 1]
```

```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.20)
```

Appling Decision Tree Algo

```
from sklearn.tree import DecisionTreeClassifier
```

```
dt_default = DecisionTreeClassifier(max_depth=5)
dt_default.fit(X_train,y_train)
```

```
▾      DecisionTreeClassifier
DecisionTreeClassifier(max_depth=5)
```

```
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

```
y_pred_default = dt_default.predict(X_test)
print("confusion matrix\n",confusion_matrix(y_test,y_pred_default))
print(classification_report(y_test,y_pred_default))
```

```
confusion matrix
 [[4310  243]
 [ 713  767]]
              precision    recall  f1-score   support

           0       0.86      0.95      0.90      4553
           1       0.76      0.52      0.62      1480

    accuracy                           0.84      6033
   macro avg       0.81      0.73      0.76      6033
weighted avg       0.83      0.84      0.83      6033
```

```
print("accuracy score: ",accuracy_score(y_test,y_pred_default))
```

```
accuracy score:  0.8415382065307475
```