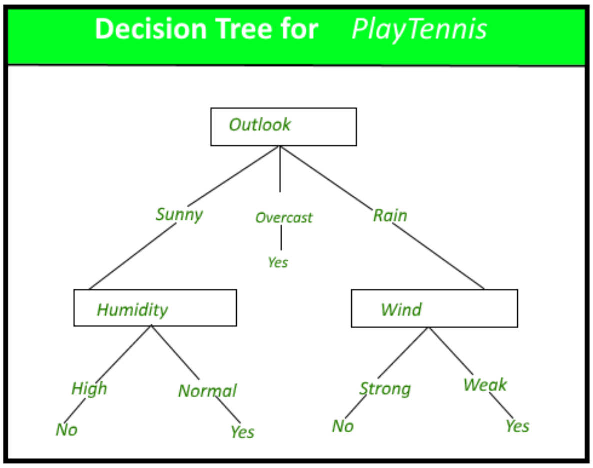
Experiment No. 3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

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. Discuss about the how categorical attributes have been dealt with during data pre-processing.

- Firstly all the unnecessary columns were drooped out and different graphs were plotted. For the categorical data , "Label Encoder" was used which converted all the categorical data into numerical data where each attribute within an entity was assigned an unique numerical value. It was used so as to establish an relationship between the attributes.

2. Discuss the hyper-parameter tunning done based on the decision tree obtained.

- It is the process of determining the right combination of hyperparameters that maximizes the models performance. It is done by carrying out multiple trials on training set. In this it is done on max_depth , min_samples_split , min_samples_leaf, max_leaf_nodes.

3. Comment on the accuracy, confusion matrix, precision, recall and F1 score obtained.

- Accuracy : 0.81.
- Confusion matrix : [[7921 380]
  [1733 1010]]
- Precision : 0.82
- Recall : 0.95
- F1 Score : 0.88

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

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

	age	workclass	fnlwgt	education	educational-num	marital-status	occupation	relation
0	25	Private	226802	11th	7	Never-married	Machine-op-inspct	Own-
1	38	Private	89814	HS-grad	9	Married-civ-spouse	Farming-fishing	Husl

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 48842 entries, 0 to 48841
Data columns (total 15 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   age              48842 non-null  int64 
 1   workclass        48842 non-null  object
 2   fnlwgt           48842 non-null  int64 
 3   education        48842 non-null  object
 4   educational-num  48842 non-null  int64 
 5   marital-status   48842 non-null  object
 6   occupation       48842 non-null  object
 7   relationship     48842 non-null  object
 8   race             48842 non-null  object
 9   gender           48842 non-null  object
 10  capital-gain     48842 non-null  int64 
 11  capital-loss     48842 non-null  int64 
 12  hours-per-week   48842 non-null  int64 
 13  native-country   48842 non-null  object
 14  income           48842 non-null  object
dtypes: int64(6), object(9)
memory usage: 5.6+ MB
```

```
df['capital-gain'].value_counts()
```

```
0        44807
15024      513
7688       410
7298       364
99999      244
         ...  
1111         1
7262         1
22040        1
1639         1
2387         1
Name: capital-gain, Length: 123, dtype: int64
```

```
df['capital-loss'].value_counts()
```

```
0       46560
1902      304
1977      253
1887      233
2415       72
        ...  
2465        1
2080        1
155         1
1911        1
2201        1
Name: capital-loss, Length: 99, dtype: int64
```

```
df.drop(['fnlwgt','race','capital-gain','capital-loss'],axis=1,inplace=True)
```

```
df.replace('?',np.nan,inplace = True)
df.dropna(inplace=True)
```

```
df.duplicated().sum()
```

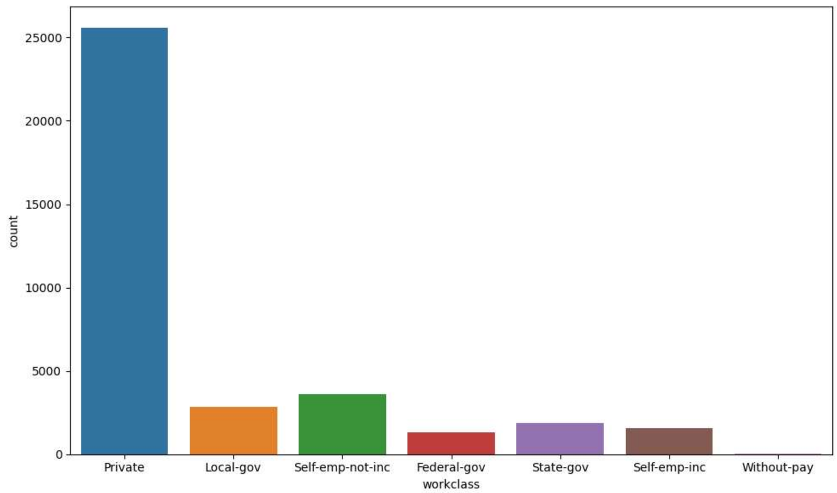
    8411

```
df=df.drop_duplicates()
```

```
df.shape
```

    (36811, 11)

```
plt.figure(figsize=(10, 6))
sns.countplot(df , x='workclass' )
plt.tight_layout()
```



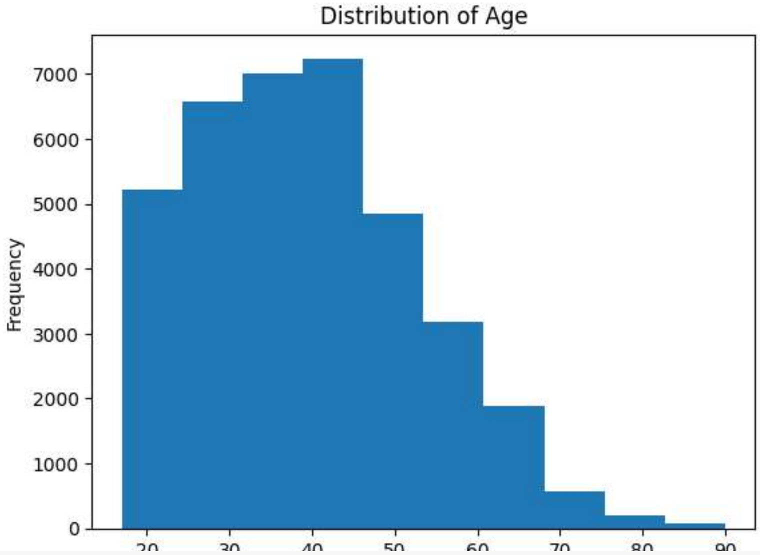

```
from sklearn import preprocessing
label_encoder = preprocessing.LabelEncoder()

df['workclass'] = label_encoder.fit_transform(df['workclass'])
df['marital-status'] = label_encoder.fit_transform(df['marital-status'])
df['occupation'] = label_encoder.fit_transform(df['occupation'])
df['relationship'] = label_encoder.fit_transform(df['relationship'])
df['gender'] = label_encoder.fit_transform(df['gender'])
df['native-country'] = label_encoder.fit_transform(df['native-country'])
df['income'] = label_encoder.fit_transform(df['income'])
df['educational-num']=label_encoder.fit_transform(df['educational-num'])
df['education'] = label_encoder.fit_transform(df['education'])
```

```
df.head()
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	gei
0	25	2	1	6	4	6	3	
1	38	2	11	8	2	4	0	
2	28	1	7	11	2	10	0	
3	44	2	15	9	2	6	0	

```
plt.hist(df['age'])
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.title('Distribution of Age')
```

```
Text(0.5, 1.0, 'Distribution of Age')
```




```
df['income'] = df['income'].astype('category')
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 36811 entries, 0 to 48841
Data columns (total 11 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   age              36811 non-null  int64
 1   workclass        36811 non-null  int64
 2   education        36811 non-null  int64
 3   educational-num  36811 non-null  int64
 4   marital-status   36811 non-null  int64
 5   occupation       36811 non-null  int64
 6   relationship     36811 non-null  int64
 7   gender           36811 non-null  int64
 8   hours-per-week   36811 non-null  int64
 9   native-country   36811 non-null  int64
 10  income           36811 non-null  category
dtypes: category(1), int64(10)
memory usage: 3.1 MB
```

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

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

```
X.head(5)
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	ge
0	25	2	1	6	4	6	3	
1	38	2	11	8	2	4	0	
2	28	1	7	11	2	10	0	
3	44	2	15	9	2	6	0	

```
y.head(5)
```

```
0    0
1    0
2    1
3    1
5    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=0.30,random_state=100)

X_train.head()
```

	age	workclass	education	educational-num	marital-status	occupation	relationship	gender	hours-per-week
37228	32	2	8	10	2	12	0	1	40
13340	22	2	3	1	4	7	2	1	40
47475	59	4	15	9	2	11	0	1	40

```
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(classification_report(y_test,y_pred_default))
```

```
              precision    recall  f1-score   support

           0       0.82      0.95      0.88      8301
           1       0.73      0.37      0.49      2743

    accuracy                           0.81     11044
   macro avg       0.77      0.66      0.69     11044
weighted avg       0.80      0.81      0.78     11044
```

```
print(confusion_matrix(y_test,y_pred_default))
```

```
[[7921  380]
 [1733 1010]]
```