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:

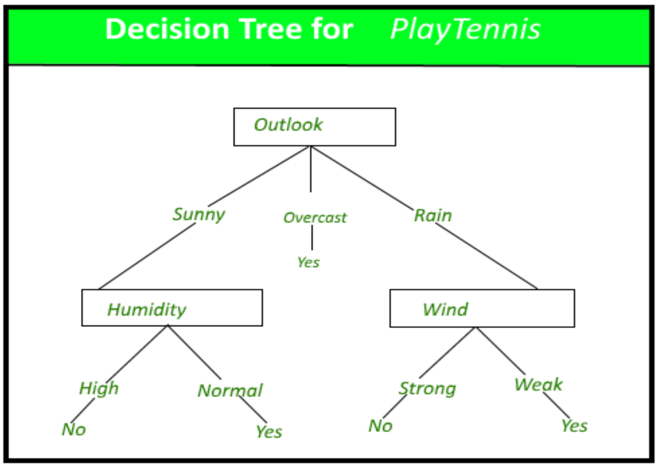
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model.

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:

```
import os

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

# To ignore warning messages

import warnings

warnings.filterwarnings('ignore')

# Adult dataset path

adult_dataset_path = "../input/adult_dataset.csv"
```

```
# Function for loading adult dataset

def load_adult_data(adult_path=adult_dataset_path):

    csv_path = os.path.join(adult_path)

    return pd.read_csv(csv_path)

# Calling load adult function and assigning to a new variable df

df = load_adult_data()

# load top 3 rows values from adult dataset

df.head(3)
```

:

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0

```
print ("Rows     : " ,df.shape[0])

print ("Columns  : " ,df.shape[1])

print ("\nFeatures : \n" ,df.columns.tolist())

print ("\nMissing values :  ", df.isnull().sum().values.sum())

print ("\nUnique values :  \n",df.nunique())
```

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

pull top 5 row values to understand the data and how it's look like

df.head()

checking "?" total values present in particular 'workclass' feature

df_check_missing_workclass = (df['workclass']=='?').sum()

df_check_missing_workclass

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	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

checking "?" total values present in particular 'occupation' feature

df_check_missing_occupation = (df['occupation']=='?').sum()

df_check_missing_occupation

```
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

```
age               0.000000
workclass         5.638647
fnlwgt            0.000000
education         0.000000
education.num     0.000000
marital.status    0.000000
occupation        5.660146
relationship      0.000000
race              0.000000
sex               0.000000
capital.gain      0.000000
capital.loss      0.000000
hours.per.week    0.000000
native.country    1.790486
income            0.000000
dtype: float64
```

contain any missing value as '?'

df.apply(lambda x: x !='?',axis=1).sum()

```
age               32561
workclass         30725
fnlwgt            32561
education         32561
education.num     32561
marital.status    32561
occupation        30718
relationship      32561
race              32561
sex               32561
capital.gain      32561
capital.loss      32561
hours.per.week    32561
native.country    31978
income            32561
dtype: int64
```

df = df[df['workclass'] !='?']

df.head()

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain
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_categorical = df.select_dtypes(include=['object'])

df_categorical.apply(lambda x: x=='?',axis=1).sum()

df = df[df['occupation'] !='?']

df = df[df['native.country'] !='?']

dropping the "?"s from occupation and native.country

df = df[df['occupation'] !='?']

df = df[df['native.country'] !='?']

```
workclass            0
education            0
marital.status       0
occupation           7
relationship         0
race                 0
sex                  0
native.country     556
income               0
dtype: int64
```

check the dataset whether cleaned or not?

df.info()

from sklearn import preprocessing

encode categorical variables using label Encoder

select all categorical variables

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

df_categorical.head()

	workclass	education	marital.status	occupation	relationship	race	sex	native.country	income
1	Private	HS-grad	Widowed	Exec-managerial	Not-in-family	White	Female	United-States	<=50K
3	Private	7th-8th	Divorced	Machine-op-inspct	Unmarried	White	Female	United-States	<=50K
4	Private	Some-college	Separated	Prof-specialty	Own-child	White	Female	United-States	<=50K
5	Private	HS-grad	Divorced	Other-service	Unmarried	White	Female	United-States	<=50K
6	Private	10th	Separated	Adm-clerical	Unmarried	White	Male	United-States	<=50K

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.head()

df.info()

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

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation
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()

from sklearn.model_selection import train_test_split

X = df.drop('income',axis=1)

Putting response/dependent variable/feature to y

y = df['income']

X.head(3)

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupation
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

y.head(3)

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

Splitting the data into train and test

X_train,X_test,y_train,y_test =
train_test_split(X,y,test_size=0.30,random_state=99)

X_train.head()

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclass	education	marital.status	occupa
24351	42	289636	9	0	0	46	2	11	2	13
15626	37	52465	9	0	0	40	1	11	4	7
4347	38	125933	14	0	0	40	0	12	2	9
23972	44	183829	13	0	0	38	5	9	4	0
26843	35	198841	11	0	0	35	2	8	0	12

from sklearn.tree import DecisionTreeClassifier

Fitting the decision tree with default hyperparameters, apart from

max_depth which is 5 so that we can plot and read the tree.

dt_default = DecisionTreeClassifier(max_depth=5)

dt_default.fit(X_train,y_train)

```
DecisionTreeClassifier(class_weight=None, criterion='gini', max_depth=5,
                       max_features=None, max_leaf_nodes=None,
                       min_impurity_decrease=0.0, min_impurity_split=None,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort=False,
                       random_state=None, splitter='best')
```

From sklearn metricimport

classification_report,confusion_matrix,accuracy_score

making predictions

y_pred_default = dt_default.predict(X_test)

```
              precision    recall  f1-score   support

           0       0.86      0.95      0.91      6867
           1       0.78      0.52      0.63      2182

    accuracy                           0.85      9049
   macro avg       0.82      0.74      0.77      9049
weighted avg       0.84      0.85      0.84      9049
```

print(confusion_matrix(y_test,y_pred_default))

print(accuracy_score(y_test,y_pred_default))

```
[[6553  314]
 [1038 1144]]
0.8505912255497845
```

!pip install pydotplus

from IPython.display import Image

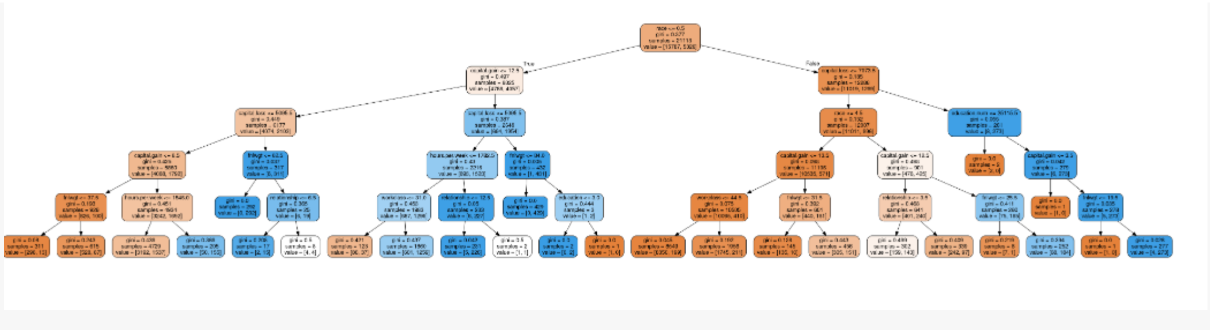
from sklearn.externals.six import StringIO

from sklearn.tree import export_graphviz

import pydotplus,graphviz

Putting features

```
features = list(df.columns[1:])

features

dot_data = StringIO()

 export_graphviz(dt_default, out_file=dot_data,

 feature_names=features, filled=True,rounded=True)

graph = pydotplus.graph_from_dot_data(dot_data.getvalue())

Image(graph.create_png())
```



Conclusion:

1. Discuss about the how categorical attributes have been dealt with during data pre-processing.
   - Categorical attribute handling methods include label encoding, one-hot encoding, frequency encoding, target encoding, binary encoding, embedding layers (for neural networks), and handling missing values appropriately.

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

- Hyperparameter tuning involves adjusting parameters like max_depth, min_samples_leaf, min_samples_split, criterion, max_features, min_impurity_decrease, ccp_alpha, and others.
- The tuning process aims to find the optimal combination of hyperparameters that balances model complexity and predictive performance.
- Techniques like Grid Search or Random Search can be used for systematic hyperparameter exploration.
- Careful tuning can improve the Decision Tree's accuracy and prevent overfitting or underfitting.

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

	precision	recall	f1-score	support
0	0.85	0.96	0.90	6867
1	0.77	0.47	0.59	2182
accuracy			0.84	9049
macro avg	0.81	0.71	0.74	9049
weighted avg	0.83	0.84	0.82	9049