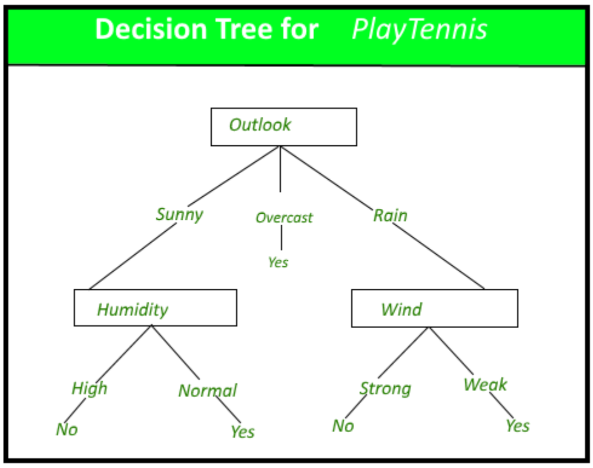
Experiment No.3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance: 07/08/2023
Date of Submission: 20/8/2023

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:

Following categorical columns are processed using label encoding to convert their values into unique numerical labels. This numerical representation is essential for many machine learning algorithms that require input data to be in numerical format. By encoding these categorical features, you make the data suitable for training machine learning models.

1. "workclass" describes their employment status.
2. "education" tells us their highest level of education.
3. "marital-status" shows their marital situation.
4. "occupation" reveals their job roles.
5. "relationship" details their family status.
6. "race" typically notes their racial background.
7. "sex" indicates gender.
8. "native-country" often specifies their country of origin or citizenship.

In the code you provided, certain columns are dropped during the data pre-processing steps. Specifically, the following columns are dropped:

1. Channel: This column is dropped using the data.drop(labels=(['Channel','Region']),axis=1,inplace=True) line of code. It appears that the Channel column is removed from the dataset.
2. Region: Similar to the Channel column, the Region column is also dropped using the same line of code. This column is removed from the dataset as well.

2. Hyperparameter Tuning:

In this code, hyperparameter tuning is applied to the Decision Tree classifier:

The Decision Tree classifier is created with a specified maximum depth of 5: DecisionTreeClassifier(max_depth=5). This is a form of hyperparameter tuning as it controls the depth of the tree.However, this code snippet does not demonstrate an extensive hyperparameter tuning process. In practice, more comprehensive methods like grid search or random search can be employed to systematically search for the best hyperparameters. Here, only the max_depth is adjusted.

3. **Evaluation Metrics for Classification Models**

	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

Confusion Matrix: It correctly predicted 4310 instances as negative (0) and 767 instances as positive (1), but it had 243 false positive predictions and 713 false negatives.

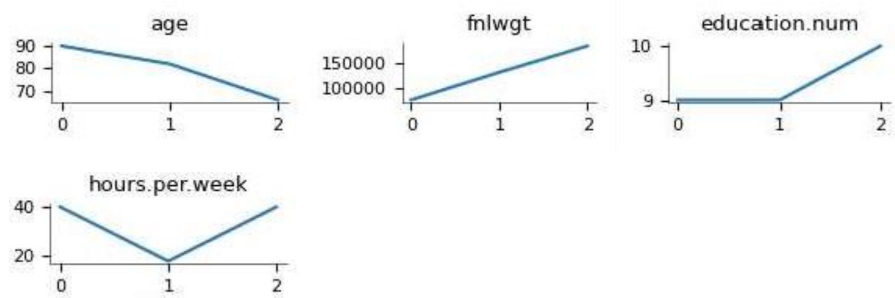
Performance Metrics: The precision for positive predictions (1) is lower at 0.76 compared to Model 1, but the recall is slightly better at 0.52. The F1-score for this model is 0.62.

Accuracy: The overall accuracy of this model is 0.84.

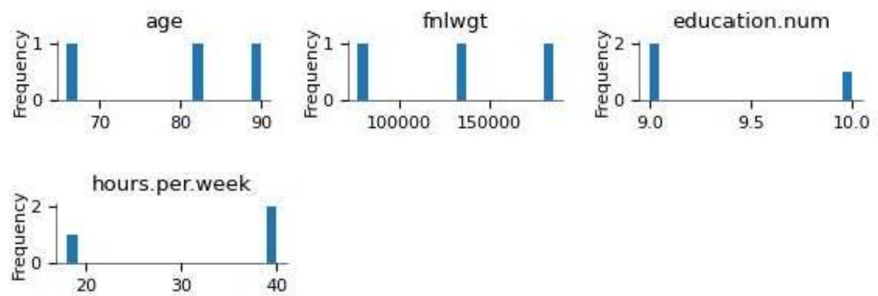
```
importos
import numpyasnp
importpandasaspd
importmatplotlíb.pyplotasplt
import seaborn as sns
%matplotlibinline
import warníngs
warníngs.fílterwarníngs('ígnore')


adult_dataset_path=”/content/adult.csv”

def load_adult_data(adult_path=adult_dataset_path):
    csv_path = os.path.join(adu1t_path)
    returnpd.read_csv(csv_path)


df=load_adult_data()

df.head(3)
```

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	re1ati‹
0	90	?	77053	HS-grad	9	Widowed	?	Not-ir
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-ir
2	66	?	186061	Some-college	10	Widowed	?	Um

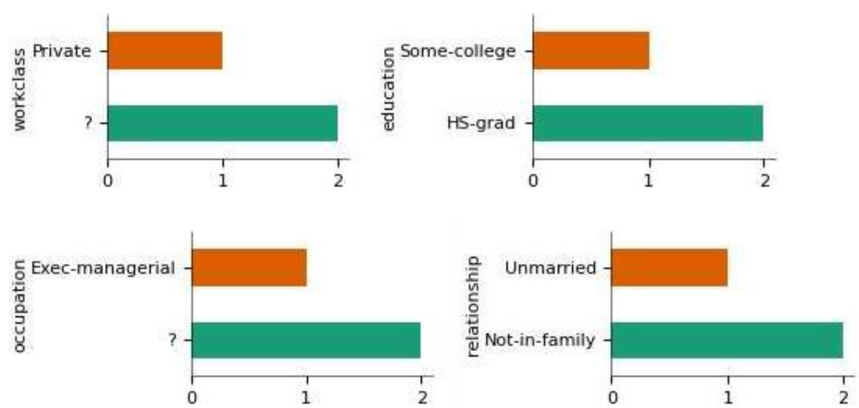
Values



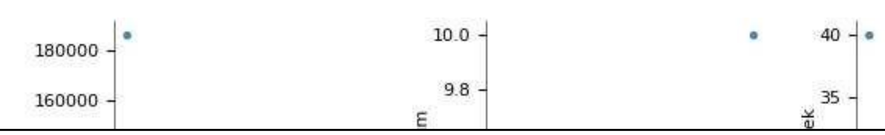
Distributions



Categoricaldistributions



2-ddistributions



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

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

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

Rows 3256
Columns 1

2-dcategoricaldistributions

```
Features
 ['age', 'workclass', 'fnlwgt', 'education', 'education.num','marital.status','occupation', 'relationship','rac

Missing values    0

Uniquevalues:
 ageworkc1a            73
ssfnlwgt               9
education          21648
education.num         16
marital.status        16
occupation             7
relationship          15
race                   6
sex                    5
capital.gain           2
capital.loss         119
hours.per.week        92
native.country        94
income                42
dtype:int64            2
```

occupation

df.info()

```
<class'pandas.core.frame.DataFrame'>
       d       1entries,0to32560
RangeIndex:3256   15columns):
DataColumn  (total  Non-NullCount    Dtype

 0   age             32561non-null   int64
 1   workclass       32561non-null   object
 2   fnlwgt          32561non-null   int64
 3   education       32561non-null   object
,4   education.num   32561non-null   int64
 5   marital.status32561non-null     object
 6   occupation      32561non-null   object
 7   relationship    32561non-null   object
 8   race            32561non-null   object
 9   sex             32561non-null   object
 10  capital.gain    32561non-null   int64
 11  capital.loss    32561non-null   int64
 12  hours.per.week32561non-null     int64
 13  native.country32561non-null     object
 14  income          32561non-null   object
dtypes: int64(6), object(9)
memoryusage:3.7+MB
```

Faceted distributions

df.describe()

	age	fnlwgt	education.num	capital.gain	capital.loss	hour•
count	32561.000000	3.256100e+04	32561.000000	32561.000000	32561.000000	32
mean	38.581647	1.897784e+05	10.080679	1077.648844	87.303830	
std	13.640433	1.055500e+05	2.572720	7385.292085	402.960219	
min	17.000000	1.228500e+04	1.000000	0.000000	0.000000	
25°/»	28.000000	1.178270e+05	9.000000	0.000000	0.000000	
50%	37.000000	1.783560e+05	10.000000	0.000000	0.000000	
75%	48.000000	2.370510e+05	12.000000	0.000000	0.000000	
max	90.000000	1.484705e+06	16.000000	99999.000000	4356.000000	

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

```
1836
```

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

```
1843
```

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

```
age               0.000000
workc1ass         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
```

```
df.apply(lambdax: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	relatil
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-ir
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Uni
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Ov
5	34	Private	216864	HS-grad	9	Divorced	Other-service	UnrUn
6	38	Private	150601	10th	6	Separated	Adm-clerical	r

```
df_categorical = df.select_dtypes(include=['object'])
df_categorical.apply(lambdax:x=='?',axis=1).sum()
```

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

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index:30162entries,1to32560
Datacolumns (total 15columns):
 #   Column          Non-NullCountDtype

 0   age             30162 non-null  int64
 1   workclass       30162 non-null  object
 2   fnlwgt          30162 non-null  int64
 3   education       30162 non-null  object
 4   education.num   30162 non-null  int64
 5   marital.status  30162 non-null  object
 6   occupation      30162 non-null  object
 7   relationship    30162 non-null  object
 8   race            30162 non-null  object
 9   sex             30162 non-null  object
 10  capital.gain    30162 non-null  int64
 11  capital.loss    30162 non-null  int64
 12  hours.per.week  30162 non-null  int64
 13  native.country  30162 non-null  object
 14  income          30162 non-null  object
dtypes:int64(6),object(9) memory
usage: 3.7+ MB
```

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

	workclass	education	marital.status	occupation	relationship	race	sexnative
1	Private	HS-grad	Widowed	Exec-managerial	Not-in-family	White	FemaleUni
3	Private	7th-8th	Divorced	Machine-op-inspct	Unmarried	White	FemaleUniOwn-
4	Private	Some-college	Separated	Prof-specialty	childWhiteFemaleUni		

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

	workclass	education	marital.status	occupation	relationship	race	sexnative.con
1	2	11	6	3	1	4	0
3	2	5	0	6	4	4	0
4	2	15	5	9	3	4	0
5	2	11	0	7	4	4	0
6	2	0	5	0	4	4	1

```
df = df.drop(df_categorical.co1umns,axis=1)
df=pd.concat([df,df_categorical],axis=1)
df.head()
```

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

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index:30162entries,1to32560
Datacolumns (total 15columns):
 #   Column          Non-NullCountDtype

 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   30162non-nullint64
 11  race           30162non-nullint64
 12  sex            30162non-nullint64
 13  native.country30162non-nullint64
 14  income         30162non-nullint64
dtypes:int64(15)
memoryusage:3.7MB
```

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

```
df.info()
```

```
<class  'pandas.core.frame.DataFrame'>
Int64Index:30162entries,1to32560
Datacolumns (total 15columns):
 #ColumnNon-Null CountDtype

 0   age            30162non-nullint64
 1   fnlwgt         30162non-nullint64
 2   education.num  30162non-nullint64
 3   capital.gain   30162non-nullint64
 4   capital.loss   30162non-nullint64
 5   hours.per.week30162non-nullint64
 6   workclass      30162non-nullint64
 7   education      30162non-nullint64
 8   marital.status30162non-nullint64
 9   occupation     30162non-nullint64
 10  relationship   30162non-nullint64
 11  race           30162non-nullint64
 12  sex            30162non-nullint64
 13  native.country30162non-nullint64
 14  income         30162non-nullcategory
dtypes:category(1),int64(14)
memoryusage:3.5MB
```

```
from sklearn.model_selectionimport train_test_split
```

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

```
X.head(3)
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclasse‹
1	82	132870	9	0	4356	18	2
3	54	140359	4	0	3900	40	2
4	41	264663	10	0	3900	40	2

```
y.head(3)
```

```
1    0
3    0
4    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=99)

X_train.head()
```

	age	fnlwgt	education.num	capital.gain	capital.loss	hours.per.week	workclas
24351	42	289636	9	0	0	46	
15626	37	52465	9	0	0	40	
4347	38	125933	14	0	0	40	
23972	44	183829	13	0	0	38	
26843	35	198841	11	0	0	35	

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

```
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       0.86      0.95      0.91      6867
           1       0.78      0.52      0.63      2182

    accuracy                           0.85      9049
    macroavg       0.82      0.74      0.77      9049
 weightedavg       0.84      0.85      0.84      9049
```

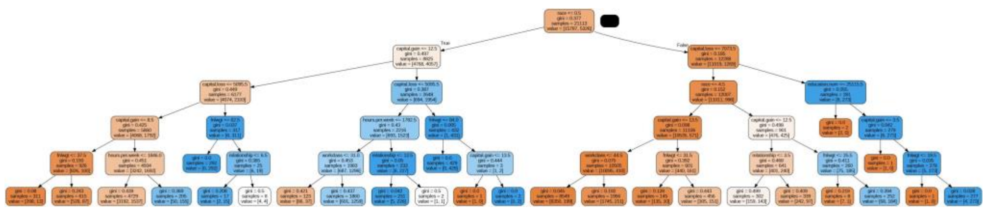
```
print(confusion_matrix(y_test,y_pred_default))
print(accuracy_score(y_test,y_pred_default))
```

```
[[6553314]
 [10391143]]
0.8504807161012267
```

```
fromIPython.displayimportImage
from six import StringIo
fromsklearn.tree importexport_graphviz
import pydotplus,graphviz

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

```
['fnlwgt',
 'education.num',
 'capital.gain',
 'capital.loss',
 'hours.per.week',
 'workclass',
 'education',
 'marital.status',
 'occupation',
```

```
        'relationship',
        'race',
        'sex',
        'native.country',
        'income']


dotdata=StringIo()
exportgraphviz(dtdefault,outfile=dotdata,
                featurenames=features,filled=True,rounded=True)

graph=pydotplus.graphfromdotdata(dotdata.getvalue())
Image(graph.create png())
```



```
pipinstallStringIo
```

```
ERROR:CouldnotfindaversionthatsatisfiestherequirementStringlo(fromversions:none) ERROR:
Nomatching distribution found forStringIo
```

2s completedat12:10PM zx