Vidyavardhini's College of Engineering & Technology Department of Computer Engineering

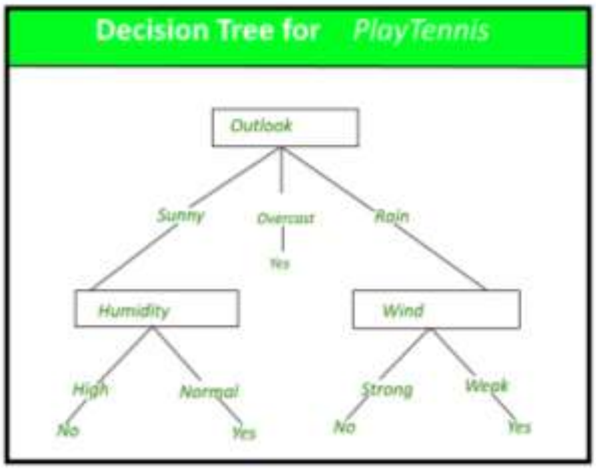
Experiment No. 3
Apply Decision Tree Algorithm on Adult Census Income Dataset and analyze the performance of the model
Date of Performance:
Date of Submission:

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

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

def load_adult_data(adult_path=adult_dataset_path):

    csv_path = os.path.join(adult_path)

    return pd.read_csv(csv_path)
```

df = load_adult_data()

df.head(3)

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
0	90	?	77053	HS-grad	9	Widowed	?	Not-in-family	White	Female	0	4356	40
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18
2	66	?	186061	Some-college	10	Widowed	?	Unmarried	Black	Female	0	4356	40

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

df.head()

	age	workclass	fnlwgt	education	education.num	marital.status	occupation	relationship	race	sex	capital.gain	capital.loss	hours.per.week
1	82	Private	132870	HS-grad	9	Widowed	Exec-managerial	Not-in-family	White	Female	0	4356	18
3	54	Private	140359	7th-8th	4	Divorced	Machine-op-inspct	Unmarried	White	Female	0	3900	40
4	41	Private	264663	Some-college	10	Separated	Prof-specialty	Own-child	White	Female	0	3900	40
5	34	Private	216864	HS-grad	9	Divorced	Other-service	Unmarried	White	Female	0	3770	45
6	38	Private	150601	10th	6	Separated	Adm-clerical	Unmarried	White	Male	0	3770	40

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

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

from sklearn import preprocessing

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()

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

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

from sklearn.model_selection import 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	workclass	education	marital.status	occupation	relationship	race	sex	nati
1	82	132870	9	0	4356	18	2	11	6	3	1	4	0	
3	54	140359	4	0	3900	40	2	5	0	6	4	4	0	
4	41	264663	10	0	3900	40	2	15	5	9	3	4	0	

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

from sklearn.tree import DecisionTreeClassifier

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
   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
```

from IPython.display import Image

from sklearn.externals.six import StringIO

from sklearn.tree import export_graphviz

import graphviz

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']
```

Tuning max_depth

from sklearn.model_selection import KFold

from sklearn.model_selection import GridSearchCV

n_folds = 5

parameters = {'max_depth': range(1, 40)}

dtree = DecisionTreeClassifier(criterion = "gini",random_state = 100)

```
tree = GridSearchCV(dtree, parameters,cv=n_folds,scoring="accuracy")

tree.fit(X_train, y_train)
```

Tuning min_samples_leaf

```
from sklearn.model_selection import KFold

from sklearn.model_selection import GridSearchCV

n_folds = 5

parameters = {'min_samples_leaf': range(5, 200, 20)}

dtree = DecisionTreeClassifier(criterion = "gini",random_state = 100)

tree = GridSearchCV(dtree, parameters cv=n_folds, scoring="accuracy")

tree.fit(X_train, y_train)
```

Tuning min_samples_split

```
from sklearn.model_selection import KFold

from sklearn.model_selection import GridSearchCV

n_folds = 5

parameters = {'min_samples_split': range(5, 200, 20)}

dtree = DecisionTreeClassifier(criterion = "gini", random_state = 100)

tree = GridSearchCV(dtree, parameters, cv=n_folds, scoring="accuracy")
```

tree.fit(X_train, y_train)

Grid Search to Find Optimal Hyperparameters

```
param_grid = {
    'max_depth': range(5, 15, 5),
    'min_samples_leaf': range(50, 150, 50),
    'min_samples_split': range(50, 150, 50),
    'criterion': ["entropy", "gini"]
}
n_folds = 5
dtree = DecisionTreeClassifier()
grid_search = GridSearchCV(estimator = dtree, param_grid = param_grid,cv = n_folds, verbose = 1)
grid_search.fit(X_train,y_train)
print("best accuracy", grid_search.best_score_)
print(grid_search.best_estimator_)
```

best accuracy 0.8514659214701843

```
# model with optimal hyperparameters

clf_gini = DecisionTreeClassifier(criterion = "gini",

random_state = 100, max_depth=10,

 min_samples_leaf=50, min_samples_split=50)

clf_gini.fit(X_train, y_train)

clf_gini.score(X_test,y_test)
```

Accuracy: 0.850922753895458

```
clf_gini = DecisionTreeClassifier(criterion = "gini", random_state = 100,
max_depth=3, min_samples_leaf=50, min_samples_split=50)

clf_gini.fit(X_train, y_train)

print(clf_gini.score(X_test,y_test))
```

0.8393192617968837

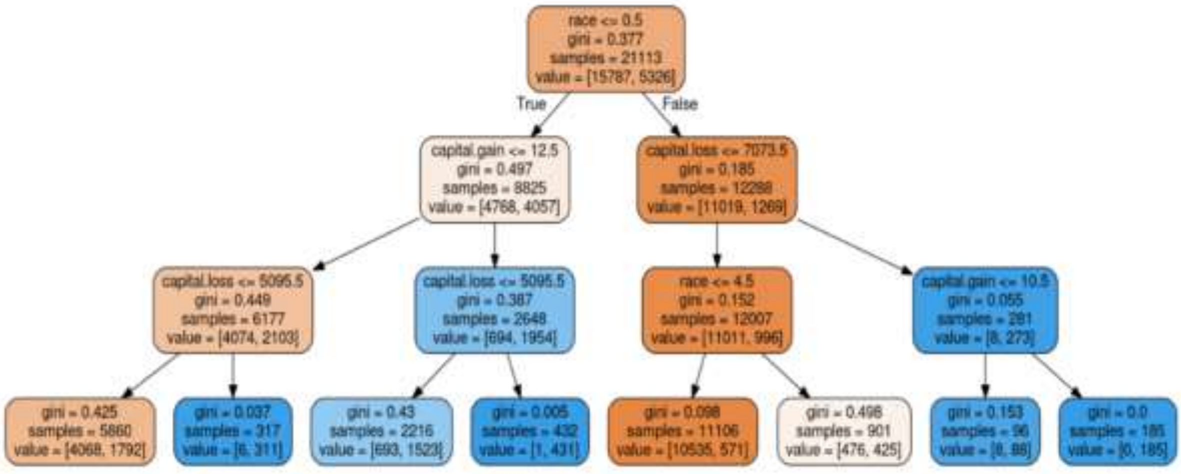
```
dot_data = StringIO()

export_graphviz(clf_gini,
out_file=dot_data,feature_names=features,filled=True,rounded=True)

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

Image(graph.create_png())
```

#Decision Tree Considering max_depth = 3



from sklearn.metrics import classification_report,confusion_matrix

y_pred = clf_gini.predict(X_test)

print(classification_report(y_test, y_pred))

	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

print(confusion_matrix(y_test,y_pred))

```
[[6564  303]
 [1151 1031]]
```

Conclusion:

1. How categorical attributes have been dealt with during data pre-processing?

➢ Since those models (being mathematical equations) can process only numeric variables. Categorical values cannot be directly used. We need to encode the categorical variables into a standard format so that sklearn can understand them and build the tree. We'll do that using the LabelEncoder() class, which comes with sklearn.preprocessing.

2. Hyper-Parameter tunning done based on the decision tree.

➢ Hyperparameter tuning (or hyperparameter optimization) is the process of determining the right combination of hyperparameters that maximizes the model performance. It works by running multiple trials in a single training process. In this implementation it done on max_depth, min_samples_leaf and min_samples_split.

3. 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