Assignment No. 3

CODE & OUTPUT :

import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
df=pd.read_csv("path of dataset")
df.shape

[8]:
```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


for col in col_names:

    print(df[col].value_counts())
```

```
buying
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
maint
high     432
med      432
low      432
vhigh    431
Name: count, dtype: int64
doors
3        432
4        432
5more    432
2        431
Name: count, dtype: int64
persons
4       576
more    576
2       575
Name: count, dtype: int64
lug_boot
med      576
big      576
small    575
Name: count, dtype: int64
safety
med     576
high    576
low     575
Name: count, dtype: int64
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

[9]:
```
df['class'].value_counts()
```

[9]:

class	count
unacc	1209
acc	384
good	69
vgood	65

dtype: int64

[10]:
```
# check missing values in variables

df.isnull().sum()
```

[10]:

	0
buying	0
maint	0
doors	0
persons	0
lug_boot	0
safety	0
class	0

dtype: int64

[11]:
```
X = df.drop(['class'], axis=1)

y = df['class']
```

[12]:
```
# split X and y into training and testing sets

from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.33, random_state = 42)
```

[13]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

[13]:
```
((1157, 6), (570, 6))
```

[14]:
```
# check data types in X_train

X_train.dtypes
```

[14]:

	0
buying	object
maint	object
doors	object
persons	object
lug_boot	object
safety	object

dtype: object

[15]:
```
X_train.head()
```

[15]:

	buying	maint	doors	persons	lug_boot	safety
83	vhigh	vhigh	5more	2	med	low
48	vhigh	vhigh	3	more	med	med
468	high	vhigh	3	4	small	med
155	vhigh	high	3	more	med	low
1043	med	high	4	more	small	low

[19]:
```
encoder = ce.OrdinalEncoder(cols=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'])


X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

[20]:
```
X_train.head()
```

[20]:

	buying	maint	doors	persons	lug_boot	safety
83	1	1	1	1	1	1
48	1	1	2	2	1	2
468	2	1	2	3	2	2
155	1	2	2	2	1	1
1043	3	2	3	2	2	1

[21]:
```
X_test.head()
```

[21]:

	buying	maint	doors	persons	lug_boot	safety
599	2	2	3	1	3	1
932	3	1	3	3	3	1
628	2	2	1	1	3	3
1497	4	2	1	3	1	2
1262	3	4	3	2	1	1

[22]:
```
from sklearn.tree import DecisionTreeClassifier
```

[23]:
```
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


# fit the model
clf_gini.fit(X_train, y_train)
```

[23]:
```
DecisionTreeClassifier(max_depth=3, random_state=0)
```
In a Jupyter environment, please rerun this cell to show the HTML representation or trust the notebook.
On GitHub, the HTML representation is unable to render, please try loading this page with nbviewer.org.

DecisionTreeClassifier
?Documentation for DecisionTreeClassifier iFitted
```
DecisionTreeClassifier(max_depth=3, random_state=0)
```

[24]:
```
y_pred_gini = clf_gini.predict(X_test)
```

[25]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion gini index: {0:0.4f}'. format(accuracy_score(y_test, y_pred_gini)))
```
```
Model accuracy score with criterion gini index: 0.8053
```

[26]:
```
y_pred_train_gini = clf_gini.predict(X_train)

y_pred_train_gini
```

[26]:
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

[27]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))
```
```
Training-set accuracy score: 0.7848
```

[28]:
```
# print the scores on training and test set

print('Training set score: {:.4f}'.format(clf_gini.score(X_train, y_train)))

print('Test set score: {:.4f}'.format(clf_gini.score(X_test, y_test)))
```
```
Training set score: 0.7848
Test set score: 0.8053
```

[33]:
```
from sklearn.metrics import confusion_matrix
cm = confusion_matrix(y_test, y_pred_gini)

print("Confusion Matrix:")
print(cm)
```
```
Confusion Matrix:
[[ 71   0  56   0]
 [ 18   0   0   0]
 [ 11   0 388   0]
 [ 26   0   0   0]]
```

[35]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test, y_pred_gini))
```
```
              precision    recall  f1-score   support

         acc       0.56      0.56      0.56       127
        good       0.00      0.00      0.00        18
       unacc       0.87      0.97      0.92       399
       vgood       0.00      0.00      0.00        26

    accuracy                           0.81       570
   macro avg       0.36      0.38      0.37       570
weighted avg       0.74      0.81      0.77       570
```