```
from google.colab import files
uploaded = files.upload()
```

Choose Files No file chosen Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to

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

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

	vhigh	vhigh.1	2	2.1	small	low	unacc
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

```
df.shape
```

```
(1727, 7)
```

```
col_names=['buying','maint','doors','persons','lug_boot','safety','class']
df.columns=col_names
col_names
```

```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

```
df.head()
```

	buying	maint	doors	persons	lug_boot	safety	class
0	vhigh	vhigh	2	2	small	med	unacc
1	vhigh	vhigh	2	2	small	high	unacc
2	vhigh	vhigh	2	2	med	low	unacc
3	vhigh	vhigh	2	2	med	med	unacc
4	vhigh	vhigh	2	2	med	high	unacc

```
df.shape
```

```
(1727, 7)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1727 entries, 0 to 1726
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1727 non-null   object
 1   maint     1727 non-null   object
 2   doors     1727 non-null   object
 3   persons   1727 non-null   object
 4   lug_boot  1727 non-null   object
 5   safety    1727 non-null   object
 6   class     1727 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
df.describe()
```

	buying	maint	doors	persons	lug_boot	safety	class
count	1727	1727	1727	1727	1727	1727	1727
unique	4	4	4	3	3	3	4
top	high	high	3	4	med	med	unacc
freq	432	432	432	576	576	576	1209

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

```
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)#dataset ka 33% will be testing set and random state is sp
```

```
X_train.shape, X_test.shape
```

```
((1157, 6), (570, 6))
```

```
import category_encoders as ce
```

```
---------------------------------------------------------------------------
ModuleNotFoundError                       Traceback (most recent call last)
<ipython-input-13-6021c897e3cf> in <cell line: 0>()
----> 1 import category_encoders as ce

ModuleNotFoundError: No module named 'category_encoders'

---------------------------------------------------------------------------
NOTE: If your import is failing due to a missing package, you can
manually install dependencies using either !pip or !apt.

To view examples of installing some common dependencies, click the
"Open Examples" button below.
---------------------------------------------------------------------------
```

OPEN EXAMPLES

```
!pip install category_encoders
import category_encoders as ce
```

```
Collecting category_encoders
  Downloading category_encoders-2.8.0-py3-none-any.whl.metadata (7.9 kB)
Requirement already satisfied: numpy>=1.14.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.26.4)
Requirement already satisfied: pandas>=1.0.5 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (2.2.2)
Requirement already satisfied: patsy>=0.5.1 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.0.1)
Requirement already satisfied: scikit-learn>=1.6.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.6.1)
Requirement already satisfied: scipy>=1.0.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (1.13.1)
Requirement already satisfied: statsmodels>=0.9.0 in /usr/local/lib/python3.11/dist-packages (from category_encoders) (0.14.4)
Requirement already satisfied: python-dateutil>=2.8.2 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2024.2)
Requirement already satisfied: tzdata>=2022.7 in /usr/local/lib/python3.11/dist-packages (from pandas>=1.0.5->category_encoders) (2025.1
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_encoders) (1
Requirement already satisfied: threadpoolctl>=3.1.0 in /usr/local/lib/python3.11/dist-packages (from scikit-learn>=1.6.0->category_encod
Requirement already satisfied: packaging>=21.3 in /usr/local/lib/python3.11/dist-packages (from statsmodels>=0.9.0->category_encoders) (
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.11/dist-packages (from python-dateutil>=2.8.2->pandas>=1.0.5->category
Downloading category_encoders-2.8.0-py3-none-any.whl (85 kB)
   ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 85.7/85.7 kB 2.2 MB/s eta 0:00:00
Installing collected packages: category_encoders
Successfully installed category_encoders-2.8.0
```

```
import category_encoders as ce
```

```
encoder=ce.OrdinalEncoder(cols=['buying','maint','doors','persons','lug_boot','safety'])#transformation
X_train=encoder.fit_transform(X_train)
X_test=encoder.transform(X_test)
```

```
X_train.head()
```

	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

```
X_test.head()
```

	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

```
from sklearn.tree import DecisionTreeClassifier
```

```
clf_gini=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=0)
clf_gini.fit(X_train,y_train) #gives decision tree model as output
```

```
▾          DecisionTreeClassifier          i ?
DecisionTreeClassifier(max_depth=3, random_state=0)
```

```
y_pred_gini=clf_gini.predict(X_test)
```

```
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
```

```
y_pred_train_gini=clf_gini.predict(X_train)
y_pred_train_gini
```

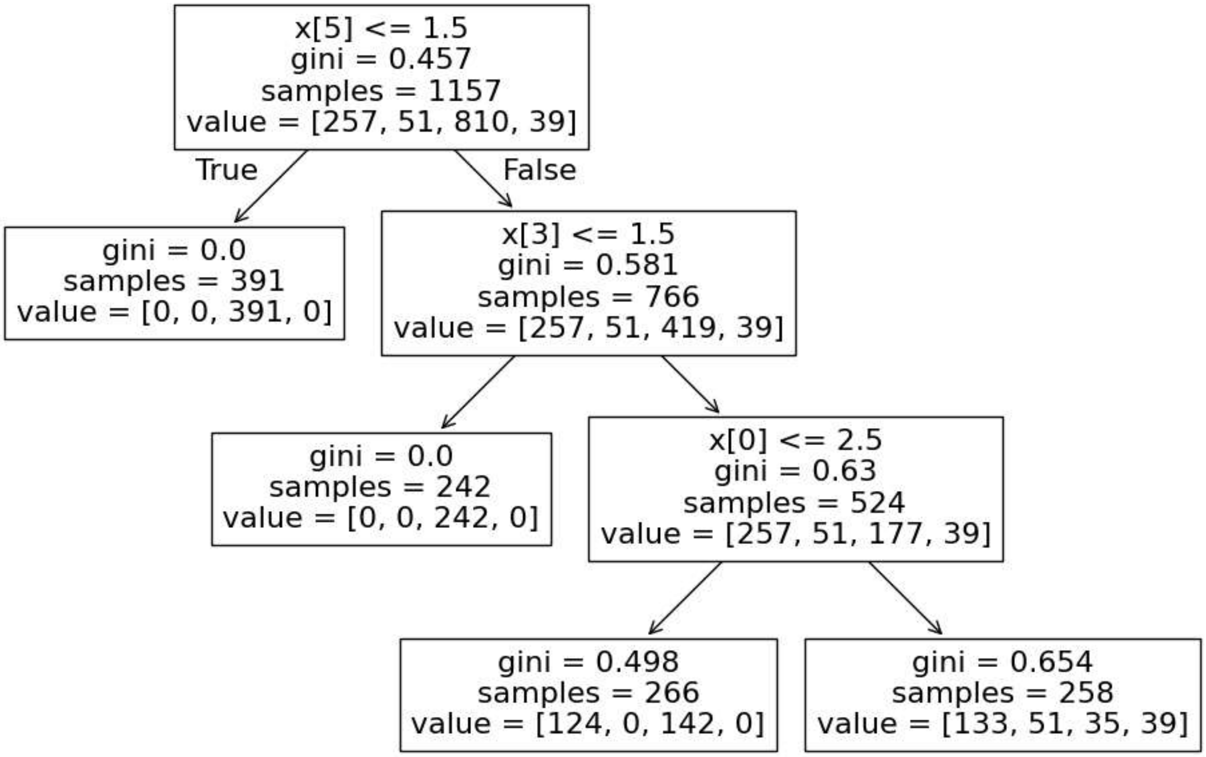
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

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

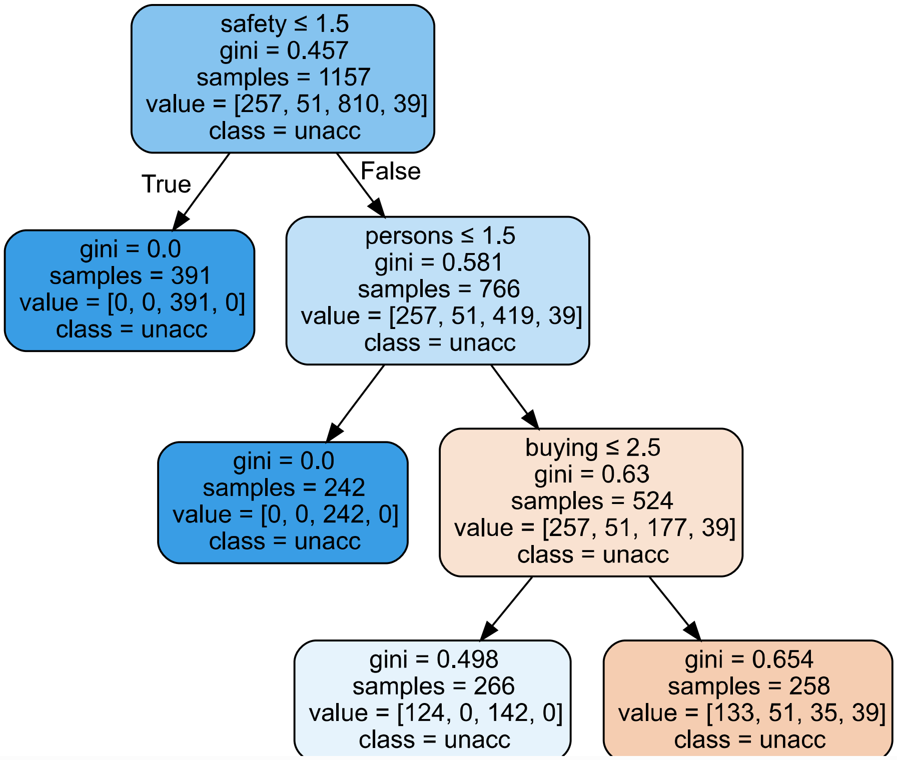
```
Training-set accuracy score: 0.7848
```

```
plt.figure(figsize=(12,8))
from sklearn import tree
tree.plot_tree(clf_gini.fit(X_train,y_train))
```

```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\ngini = 0.457\nsamples = 1157\nvalue = [257, 51, 810, 39]'),
 Text(0.16666666666666666, 0.625, 'gini = 0.0\nsamples = 391\nvalue = [0, 0, 391, 0]'),
 Text(0.25, 0.75, 'True  '),
 Text(0.5, 0.625, 'x[3] <= 1.5\ngini = 0.581\nsamples = 766\nvalue = [257, 51, 419, 39]'),
 Text(0.41666666666666663, 0.75, '  False'),
 Text(0.3333333333333333, 0.375, 'gini = 0.0\nsamples = 242\nvalue = [0, 0, 242, 0]'),
 Text(0.6666666666666666, 0.375, 'x[0] <= 2.5\ngini = 0.63\nsamples = 524\nvalue = [257, 51, 177, 39]'),
 Text(0.5, 0.125, 'gini = 0.498\nsamples = 266\nvalue = [124, 0, 142, 0]'),
 Text(0.8333333333333334, 0.125, 'gini = 0.654\nsamples = 258\nvalue = [133, 51, 35, 39]')]
```



```
import graphviz
dot_data = tree.export_graphviz(clf_gini, out_file=None, feature_names=X_train.columns, class_names=y_train, filled=True, rounded=True, speci
graph = graphviz.Source(dot_data)
graph
```



```
clf_entropy=DecisionTreeClassifier(criterion='entropy',max_depth=3,random_state=0)
clf_entropy.fit(X_train,y_train) #gives decision tree model as output
```

```
DecisionTreeClassifier

DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

```
y_pred_entropy=clf_entropy.predict(X_test)
```

```
from sklearn.metrics import accuracy_score
print('Model accuracy score with criterion entropy index: {0:0.4f}'.format(accuracy_score(y_test, y_pred_entropy)))
```

```
Model accuracy score with criterion entropy index: 0.8053
```

```
y_pred_train_entropy=clf_entropy.predict(X_train)
y_pred_train_entropy
```

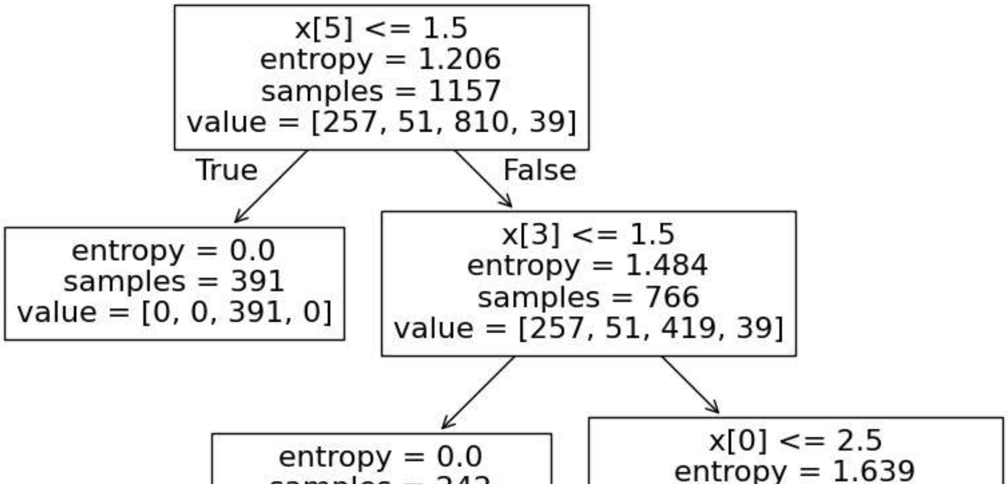
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

```
print('Training-set accuracy score: {0:0.4f}'.format(accuracy_score(y_train,y_pred_train_entropy)))
```

```
Training-set accuracy score: 0.7848
```

```
plt.figure(figsize=(12,8))
from sklearn import tree
tree.plot_tree(clf_entropy.fit(X_train,y_train))
```

```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\nentropy = 1.206\nsamples = 1157\nvalue = [257, 51, 810, 39]'),
 Text(0.16666666666666666, 0.625, 'entropy = 0.0\nsamples = 391\nvalue = [0, 0, 391, 0]'),
 Text(0.25, 0.75, 'True  '),
 Text(0.5, 0.625, 'x[3] <= 1.5\nentropy = 1.484\nsamples = 766\nvalue = [257, 51, 419, 39]'),
 Text(0.41666666666666663, 0.75, '  False'),
 Text(0.3333333333333333, 0.375, 'entropy = 0.0\nsamples = 242\nvalue = [0, 0, 242, 0]'),
 Text(0.6666666666666666, 0.375, 'x[0] <= 2.5\nentropy = 1.639\nsamples = 524\nvalue = [257, 51, 177, 39]'),
 Text(0.5, 0.125, 'entropy = 0.997\nsamples = 266\nvalue = [124, 0, 142, 0]'),
 Text(0.8333333333333334, 0.125, 'entropy = 1.758\nsamples = 258\nvalue = [133, 51, 35, 39]')]
```



```
import graphviz
dot_data = tree.export_graphviz(clf_entropy, out_file=None, feature_names=X_train.columns, class_names=y_train, filled=True, rounded=True, s
graph = graphviz.Source(dot_data)
graph
```