DECISION TREE CLASSIFIER TUTORIAL

In [1]:
```
#importing libraries
import numpy as np #linear algebra
import pandas as pd #data processing csv file
import matplotlib.pyplot as plt #data visulization
import seaborn as sns #statistical data visulization

%matplotlib inline
```

In [2]:
```
import warnings

warnings.filterwarnings('ignore')
```

In [3]:
```
#import dataset

data=pd.read_csv(r"C:\Users\Achal Raghorte\Downloads\car_evaluation.csv")
```

In [4]:
```
data
```

Out[4]:

	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
...	...	...	...	...	...	...	...
1722	low	low	5more	more	med	med	good
1723	low	low	5more	more	med	high	vgood
1724	low	low	5more	more	big	low	unacc
1725	low	low	5more	more	big	med	good
1726	low	low	5more	more	big	high	vgood

1727 rows × 7 columns

exploratory data analysis

In [5]:
```
#view dimensions of dataset
data.shape
```

Out[5]: (1727, 7)

we can see that are 1727 instances and 7 variables in the data set

In [6]:
```
# view top 5 rows

data.head()
```

Out[6]:

	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

In [7]:
```
data.columns
```

Out[7]:
```
Index(['vhigh', 'vhigh.1', '2', '2.1', 'small', 'low', 'unacc'], dtype='objec
t')
```

rename column names

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

data.columns=col_names

col_names
```

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

In [9]:
```
# lets again preview the dataset

data.head()
```

Out[9]:

	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

now we have see that columns names are renamed. now the columns have meaningful names.

view summary of dataset

In [10]:
```
data.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
```

frequency distribution of values in variables

now i will check the frequency counts of categorical variables

In [11]:
```
col_names=['buying' ,'maint' ,'doors', 'persons','lug_boot','safety','class']

for col in col_names:

 print(data[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
```

we can see that 'doors' and 'person' is seen like a categorical in nature thats why i am treet like a categorical variables

summary of variables

#there are 7 variables in the dataset. All the variables are of categorical data type. #These are given by buying, maint, doors, persons, lug_boot, safety and class. #class is the target variable.

explore class variable

In [12]:
```
data['class'].value_counts()
```

Out[12]:
```
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

the class target variable is ordinal in nature

check missing values in variables

In [13]:
```
data.isnull().sum()
```

Out[13]:
```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

We can see that there are no missing values in the dataset. I have checked the frequency distribution of values previously. It also confirms that there are no missing values in the dataset.

declare the feature vector and target variable

In [14]:
```
x=data.drop(['class'],axis=1)
y=data['class']
```

In [15]:
```python
print(x)
print(y)
```

```
     buying  maint  doors persons lug_boot safety
0     vhigh  vhigh      2       2    small    med
1     vhigh  vhigh      2       2    small   high
2     vhigh  vhigh      2       2      med    low
3     vhigh  vhigh      2       2      med    med
4     vhigh  vhigh      2       2      med   high
...     ...    ...    ...     ...      ...    ...
1722    low    low  5more    more      med    med
1723    low    low  5more    more      med   high
1724    low    low  5more    more      big    low
1725    low    low  5more    more      big    med
1726    low    low  5more    more      big   high

[1727 rows x 6 columns]
0       unacc
1       unacc
2       unacc
3       unacc
4       unacc
        ...
1722     good
1723    vgood
1724    unacc
1725     good
1726    vgood
Name: class, Length: 1727, dtype: object
```

split the data into seprate training and test set

In [16]:
```python
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
```

In [17]:
```python
# check shape of x_train , x_test

x_train.shape , x_test.shape
```

Out[17]:
```
((1157, 6), (570, 6))
```

feature engineering

In [18]:
```
# check datatypes in x_train
x_train.dtypes
```

Out[18]:
```
buying      object
maint       object
doors       object
persons     object
lug_boot    object
safety      object
dtype: object
```

encode categorical variable

In [19]:
```
x_train.head()
```

Out[19]:

	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

import category encoders

In [20]:
```
import category_encoders as ce
```

In [21]:
```
# encode variables with ordinal encoding
encoder=ce.OrdinalEncoder(cols=['buying' ,'maint' ,'doors','persons','lug_boot

x_train=encoder.fit_transform(x_train)
x_test=encoder.transform(x_test)
```

In [22]:
```
x_train.head()
```

Out[22]:

	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

In [23]:
```python
x_test.head()
```

Out[23]:

	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

decision tree classifier with criterion gini index

In [24]:
```python
# import decision tree classifier

from sklearn.tree import DecisionTreeClassifier
clf_gini=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=0)

#fit the model

clf_gini.fit(x_train,y_train)
```

Out[24]:

```
DecisionTreeClassifier

DecisionTreeClassifier(max_depth=3, random_state=0)
```

predict the test set result with criterion gini index

In [25]:
```python
y_pred_gini=clf_gini.predict(x_test)
```

check accuracy score with criterion gini index

In [26]:
```python
from sklearn.metrics import accuracy_score

print('model accuracy score with gini index:{0:0.4f}'.format(accuracy_score(y_
```

```
model accuracy score with gini index:0.8053
```

compare train set and test set accuracy

In [27]:
```
y_pred_train_gini=clf_gini.predict(x_train)

y_pred_train_gini
```

Out[27]:
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

In [28]:
```
print('training set accuracy score:{0:0.4f}'.format(accuracy_score(y_train,y_p
```

```
training set accuracy score:0.7848
```

check for overfitting and underfitting

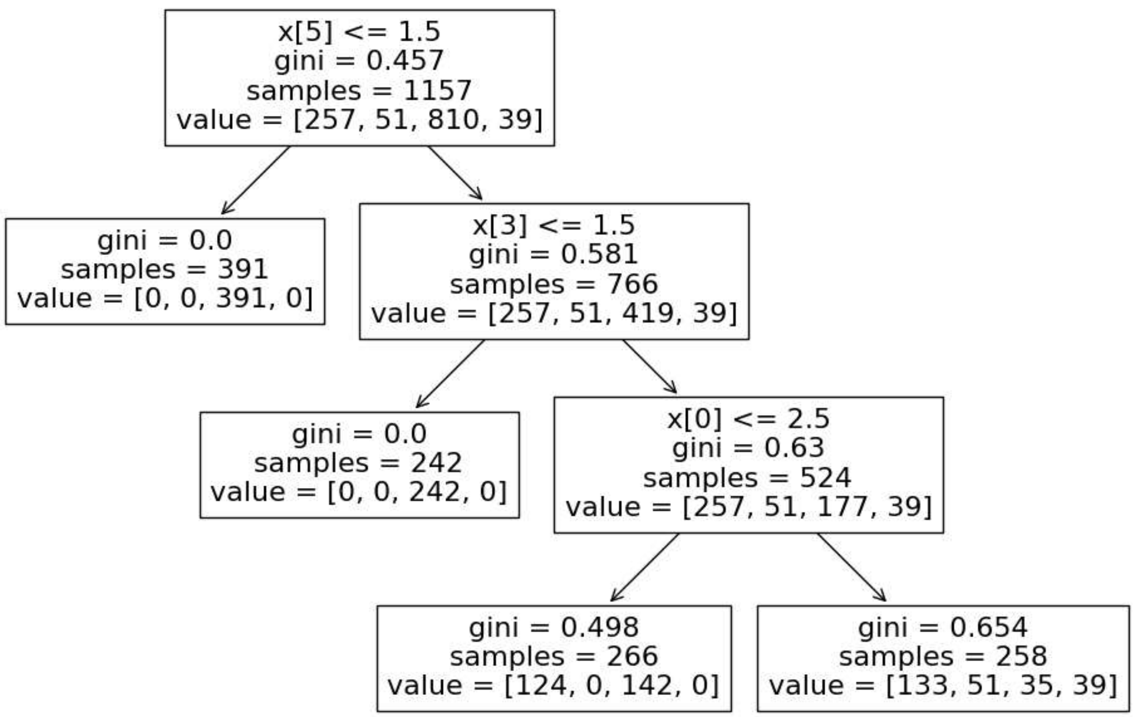
In [29]:
```
#print the score on traning and test set

print('training set score:{0:0.4f}'.format(clf_gini.score(x_train,y_train)))

print('test set score:{0:0.4f}'.format(clf_gini.score(x_test,y_test)))
```

```
training set score:0.7848
test set score:0.8053
```

visualize decission trees

In [30]:
```
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_gini.fit(x_train,y_train))
```

Out[30]:
```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\ngini = 0.457\nsamples = 1157\n
value = [257, 51, 810, 39]'),
 Text(0.16666666666666666, 0.625, 'gini = 0.0\nsamples = 391\nvalue = [0, 0,
391, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.5\ngini = 0.581\nsamples = 766\nvalue = [257, 5
1, 419, 39]'),
 Text(0.3333333333333333, 0.375, 'gini = 0.0\nsamples = 242\nvalue = [0, 0, 2
42, 0]'),
 Text(0.6666666666666666, 0.375, 'x[0] <= 2.5\ngini = 0.63\nsamples = 524\nva
lue = [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]')]
```

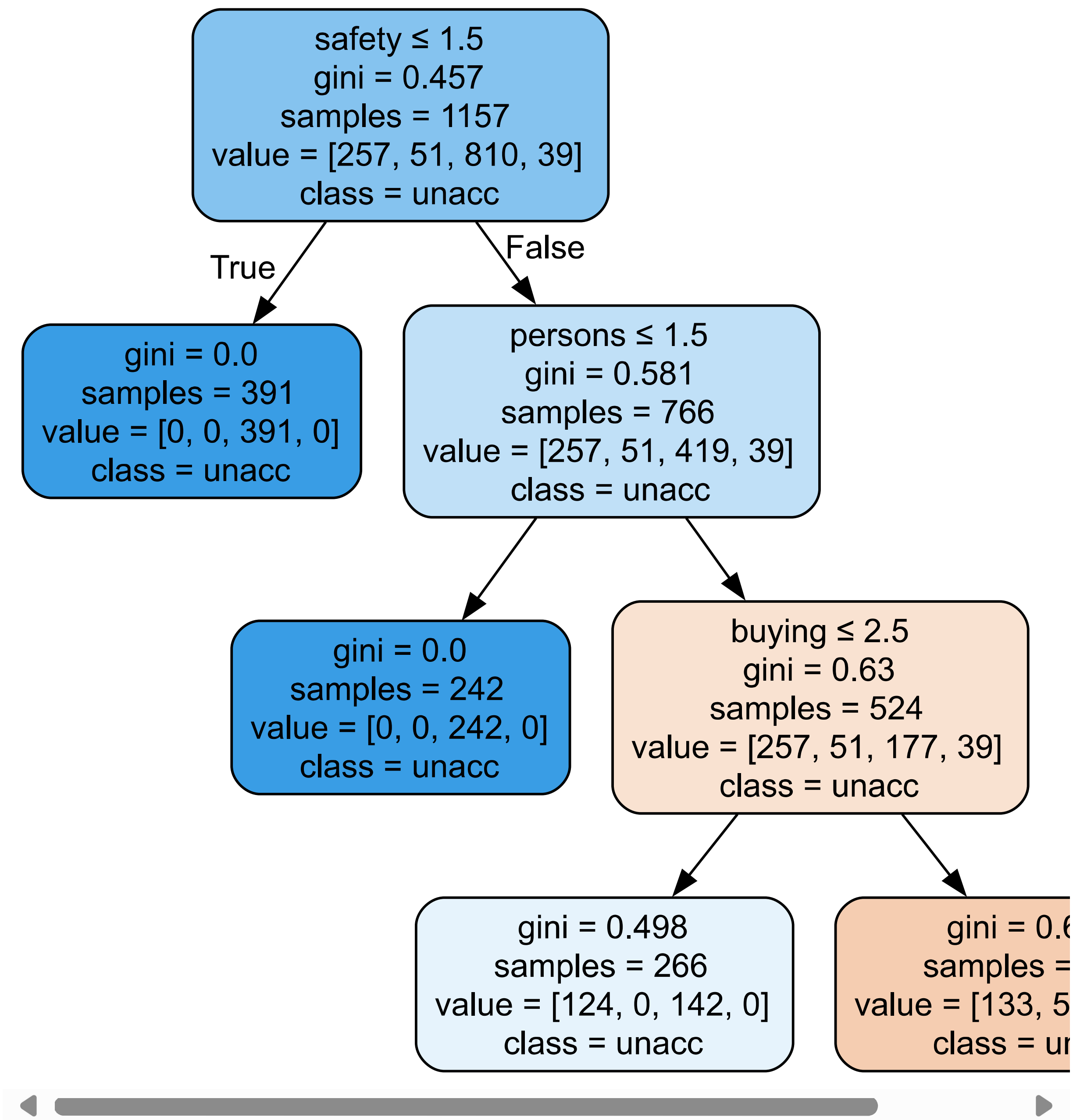


visualize decision trees with graphviz

In [31]:
```
#!pip install graphviz
```

In [32]:
```
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,
                                special_characters=True)

graph = graphviz.Source(dot_data)

graph
```

Out[32]:



decision tree classifier with criterion entropy

In [33]:
```
# instantiate the decesiontreeclassifier model with criterion entropy
clf_en = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state

# Fit the model
clf_en.fit(x_train, y_train)
```

Out[33]:
```
DecisionTreeClassifier

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

predict the test set results with criterion entropy

In [34]:
```
y_pred_en = clf_en.predict(x_test)
```

check accuracy score with criterion entropy

In [35]:
```
print('model accuracy score with criterion entropy:{0:0.4f}'.format(accuracy_s
```

```
model accuracy score with criterion entropy:0.8053
```

compare train and test set accuracy

In [36]:
```
y_pred_train_en=clf_en.predict(x_train)
y_pred_train_en
```

Out[36]:
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

In [37]:
```
print('training set accuracy score:{0:0.4f}'.format(accuracy_score(y_train,y_p
```

```
training set accuracy score:0.7848
```

check for overfitting and underfitting

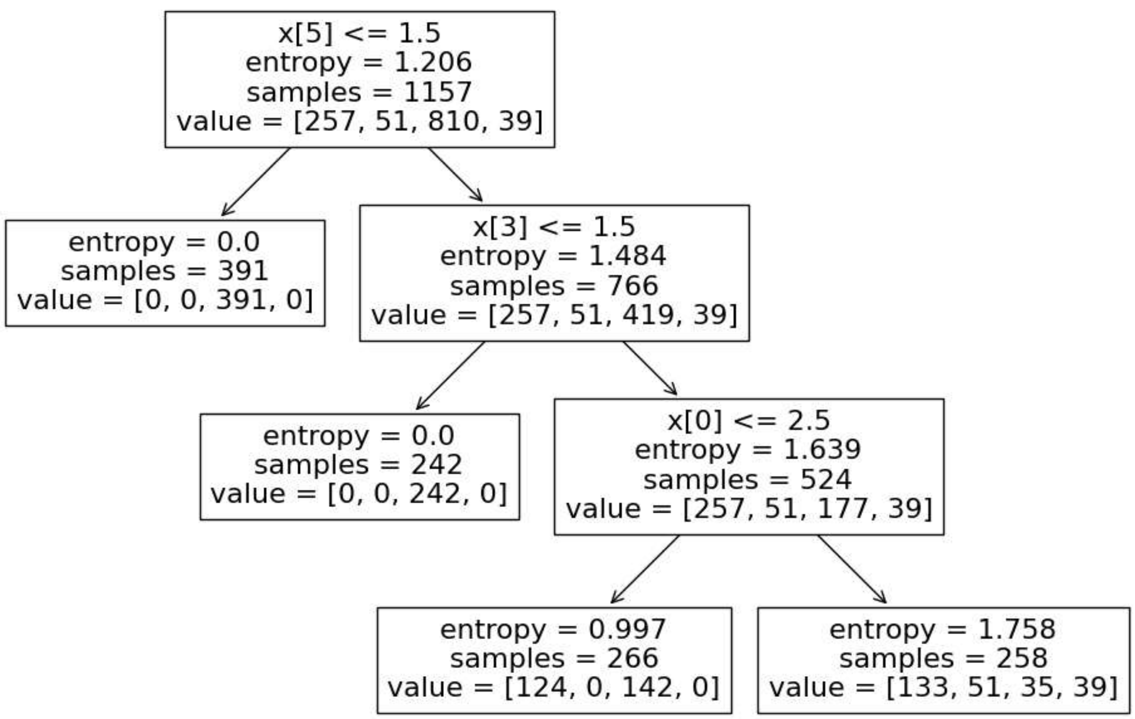
In [38]:
```
#print the scores on training and test set

print('Training set score:{0:0.4f}'.format(clf_en.score(x_train,y_train)))

print('test set score:{0:0.4f}'.format(clf_en.score(x_test,y_test)))
```

```
Training set score:0.7848
test set score:0.8053
```

visualize decision trees

In [39]:
```
plt.figure(figsize=(12,8))
from sklearn import tree
tree.plot_tree(clf_en.fit(x_train,y_train))
```

Out[39]:
```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\nentropy = 1.206\nsamples = 115
7\nvalue = [257, 51, 810, 39]'),
 Text(0.16666666666666666, 0.625, 'entropy = 0.0\nsamples = 391\nvalue = [0,
0, 391, 0]'),
 Text(0.5, 0.625, 'x[3] <= 1.5\nentropy = 1.484\nsamples = 766\nvalue = [257,
51, 419, 39]'),
 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 = [13
3, 51, 35, 39]')]
```



confusion matrix

In [40]:
```
#print the confusion matrix and slice it into four pieces

from sklearn.metrics import confusion_matrix

cm=confusion_matrix(y_test,y_pred_en)

print('Confusion matrix\n\n' , cm)
```

```
Confusion matrix

 [[ 71   0  56   0]
 [ 18   0   0   0]
 [ 11   0 388   0]
 [ 26   0   0   0]]
```

classification report ¶

In [41]:
```
from sklearn.metrics import classification_report

print(classification_report(y_test,y_pred_en))
```

```
              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
```

In [ ]: