In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline

import os
for dirname, _, filenames in os.walk('/kaggle/input'):
    for filename in filenames:
        print(os.path.join(dirname, filename))
```

In [2]:
```
import warnings

warnings.filterwarnings('ignore')
```

In [3]:
```
df=pd.read_csv("/Users/akheruddinahmed/Downloads/car_evaluation.csv")
```

In [4]:
```
df.shape
```

Out[4]: (1727, 7)

In [5]:
```
df.head()
```

Out[5]:

	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 [6]:
```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']


df.columns = col_names

col_names
```

Out[6]:
```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

In [7]:
```
df.head()
```

Out[7]:

	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

In [8]:
```
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
```

In [9]:
```
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
```

In [10]:
```
df['class'].value_counts()
```

Out[10]:
```
class
unacc    1209
acc       384
good       69
vgood      65
Name: count, dtype: int64
```

In [11]:
```
df.isnull().sum()
```

Out[11]:
```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

In [12]:
```
X = df.drop(['class'], axis=1)

y = df['class']
```

In [13]:
```
# 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)
```

In [14]:
```
# check the shape of X_train and X_test

X_train.shape, X_test.shape
```

Out[14]:
```
((1157, 6), (570, 6))
```

In [15]:
```
# check data types in X_train

X_train.dtypes
```

Out[15]:
```
buying      object
maint       object
doors       object
persons     object
lug_boot    object
safety      object
dtype: object
```

In [16]:
```
X_train.head()
```

Out[16]:

	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

In [17]:
```
# import category encoders

import category_encoders as ce
```

In [18]:
```
# encode variables with ordinal encoding

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


X_train = encoder.fit_transform(X_train)

X_test = encoder.transform(X_test)
```

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

Out[19]:

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

Out[20]:

	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

In [21]:
```
# import DecisionTreeClassifier

from sklearn.tree import DecisionTreeClassifier
```

In [22]:
```
# instantiate the DecisionTreeClassifier model with criterion gini index

clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=0)


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

Out[22]:
```
DecisionTreeClassifier(max_depth=3, random_state=0)
```

In [23]:
```
y_pred_gini = clf_gini.predict(X_test)
```

In [24]:
```
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
```

In [25]:
```
y_pred_train_gini = clf_gini.predict(X_train)

y_pred_train_gini
```

Out[25]:
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

In [26]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_gini)))
```

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

In [27]:
```
# 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)))
```

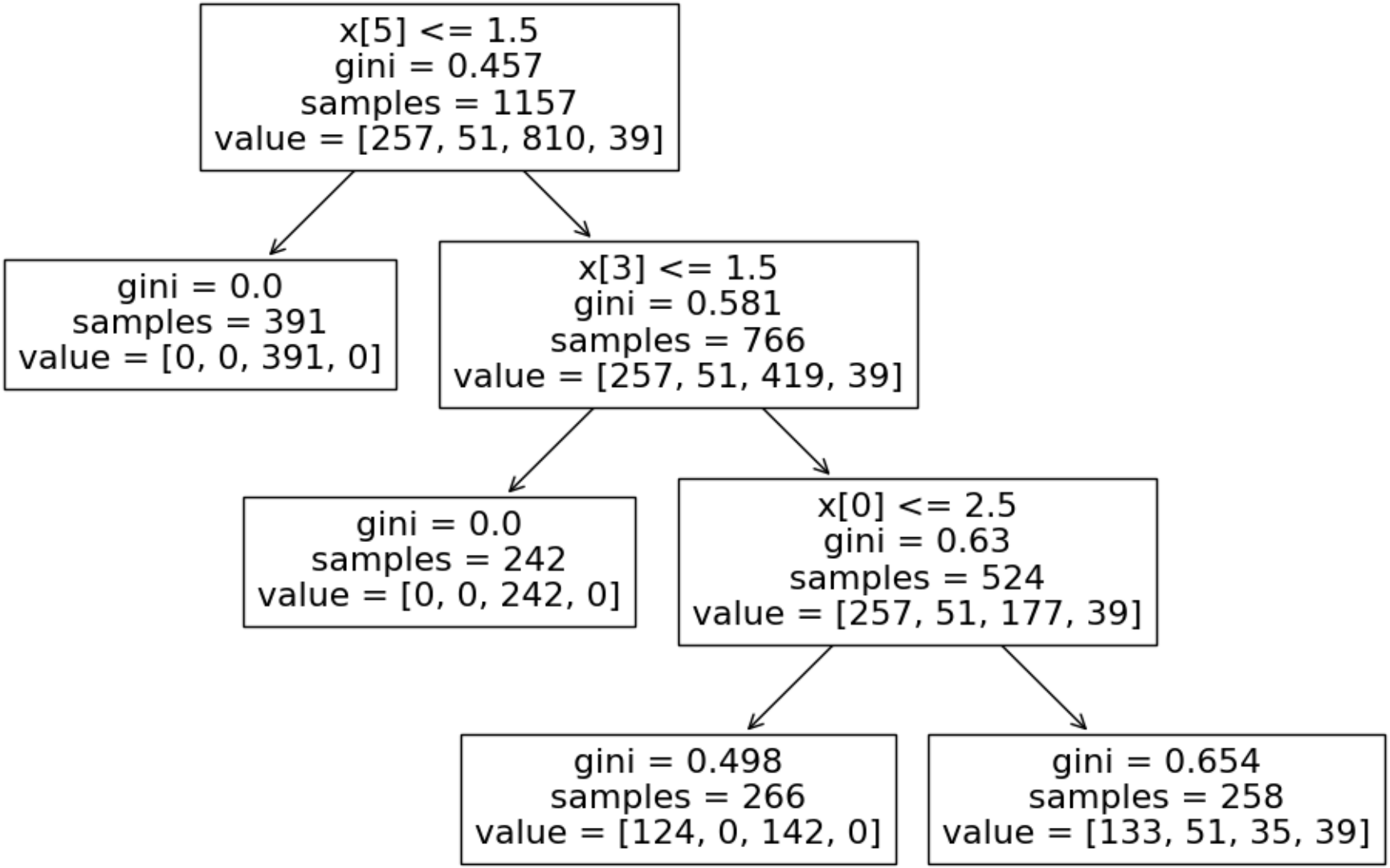
```
Training set score: 0.7848
Test set score: 0.8053
```

In [28]:
```
plt.figure(figsize=(12,8))

from sklearn import tree

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

Out[28]:
```
[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(0.5, 0.625, 'x[3] <= 1.5\ngini = 0.581\nsamples = 766\nvalue = [257, 51, 419, 39]'),
 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]')]
```

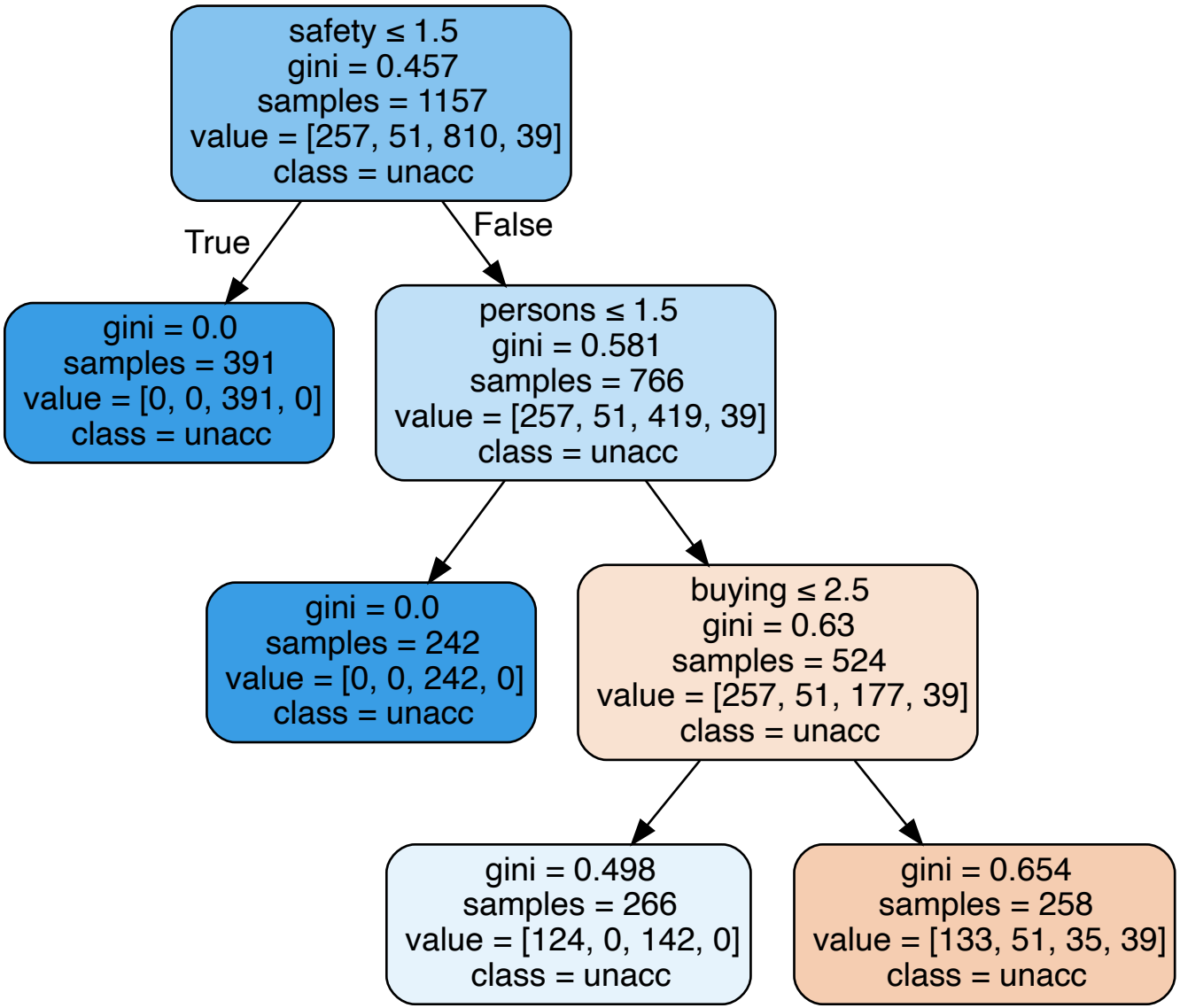

x[5] <= 1.5
gini = 0.457
samples = 1157
value = [257, 51, 810, 39]
gini = 0.0
samples = 391
value = [0, 0, 391, 0]
x[3] <= 1.5
gini = 0.581
samples = 766
value = [257, 51, 419, 39]
gini = 0.0
samples = 242
value = [0, 0, 242, 0]
x[0] <= 2.5
gini = 0.63
samples = 524
value = [257, 51, 177, 39]
gini = 0.498
samples = 266
value = [124, 0, 142, 0]
gini = 0.654
samples = 258
value = [133, 51, 35, 39]

In [29]:
```
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[29]:


safety ≤ 1.5
gini = 0.457
samples = 1157
value = [257, 51, 810, 39]
class = unacc
True
False
gini = 0.0
samples = 391
value = [0, 0, 391, 0]
class = unacc
persons ≤ 1.5
gini = 0.581
samples = 766
value = [257, 51, 419, 39]
class = unacc
gini = 0.0
samples = 242
value = [0, 0, 242, 0]
class = unacc
buying ≤ 2.5
gini = 0.63
samples = 524
value = [257, 51, 177, 39]
class = unacc
gini = 0.498
samples = 266
value = [124, 0, 142, 0]
class = unacc
gini = 0.654
samples = 258
value = [133, 51, 35, 39]
class = unacc

In [30]:
```
# instantiate the DecisionTreeClassifier model with criterion entropy

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


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

Out[30]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=0)
```

In [31]:
```
y_pred_en = clf_en.predict(X_test)
```

In [32]:
```
from sklearn.metrics import accuracy_score

print('Model accuracy score with criterion entropy: {0:0.4f}'. format(accuracy_score(y_test, y_pred_en)))
```

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

In [33]:
```
y_pred_train_en = clf_en.predict(X_train)

y_pred_train_en
```

Out[33]:
```
array(['unacc', 'unacc', 'unacc', ..., 'unacc', 'unacc', 'acc'],
      dtype=object)
```

In [34]:
```
print('Training-set accuracy score: {0:0.4f}'. format(accuracy_score(y_train, y_pred_train_en)))
```

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

In [35]:
```
# print the scores on training and test set

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

print('Test set score: {:.4f}'.format(clf_en.score(X_test, y_test)))
```

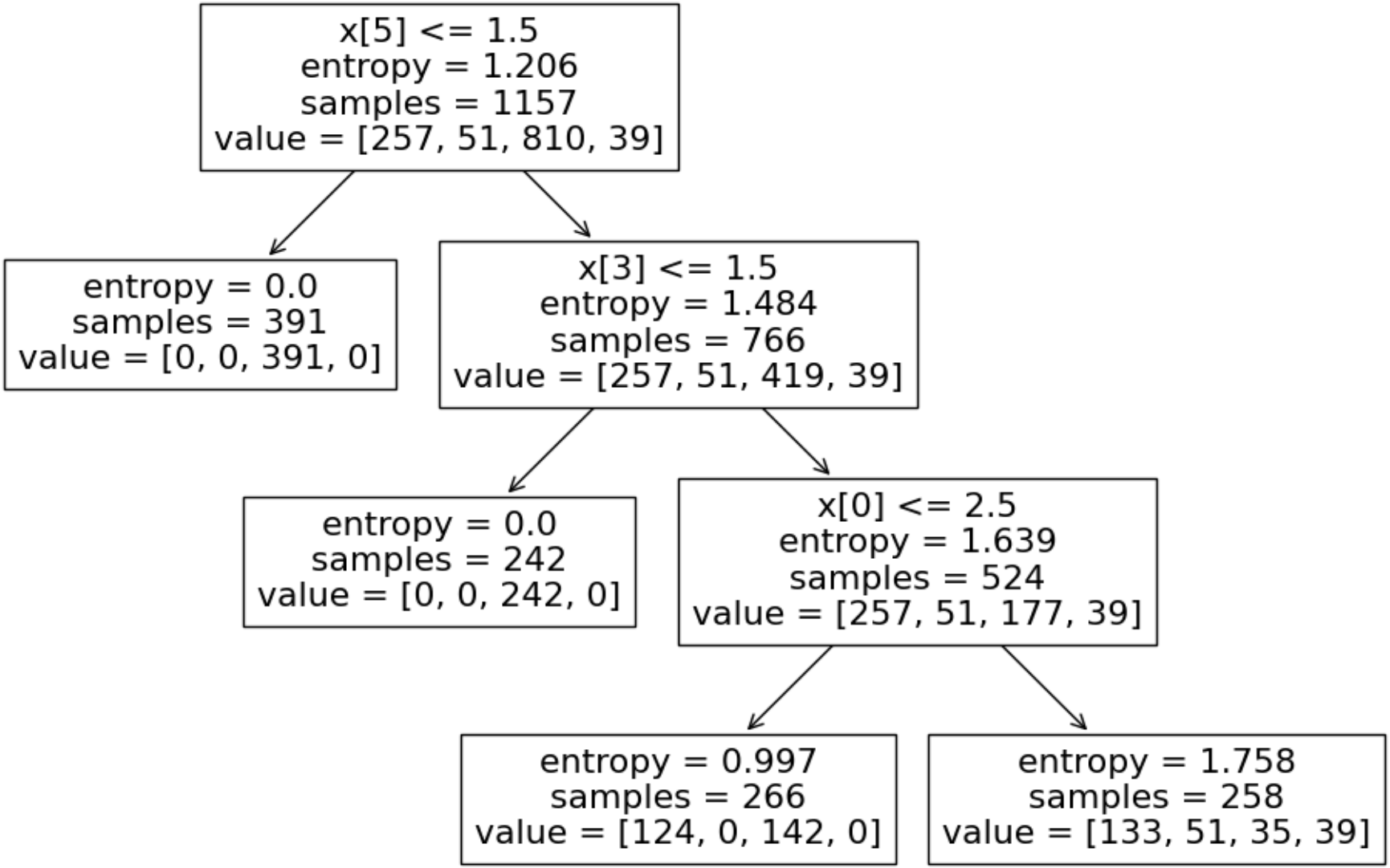
```
Training set score: 0.7848
Test set score: 0.8053
```

In [36]:
```
plt.figure(figsize=(12,8))

from sklearn import tree

tree.plot_tree(clf_en.fit(X_train, y_train))
```

Out[36]:
```
[Text(0.3333333333333333, 0.875, 'x[5] <= 1.5\nentropy = 1.206\nsamples = 1157\nvalue = [257, 51, 810, 3
9]'),
 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 = [133, 51, 35, 39]')]
```

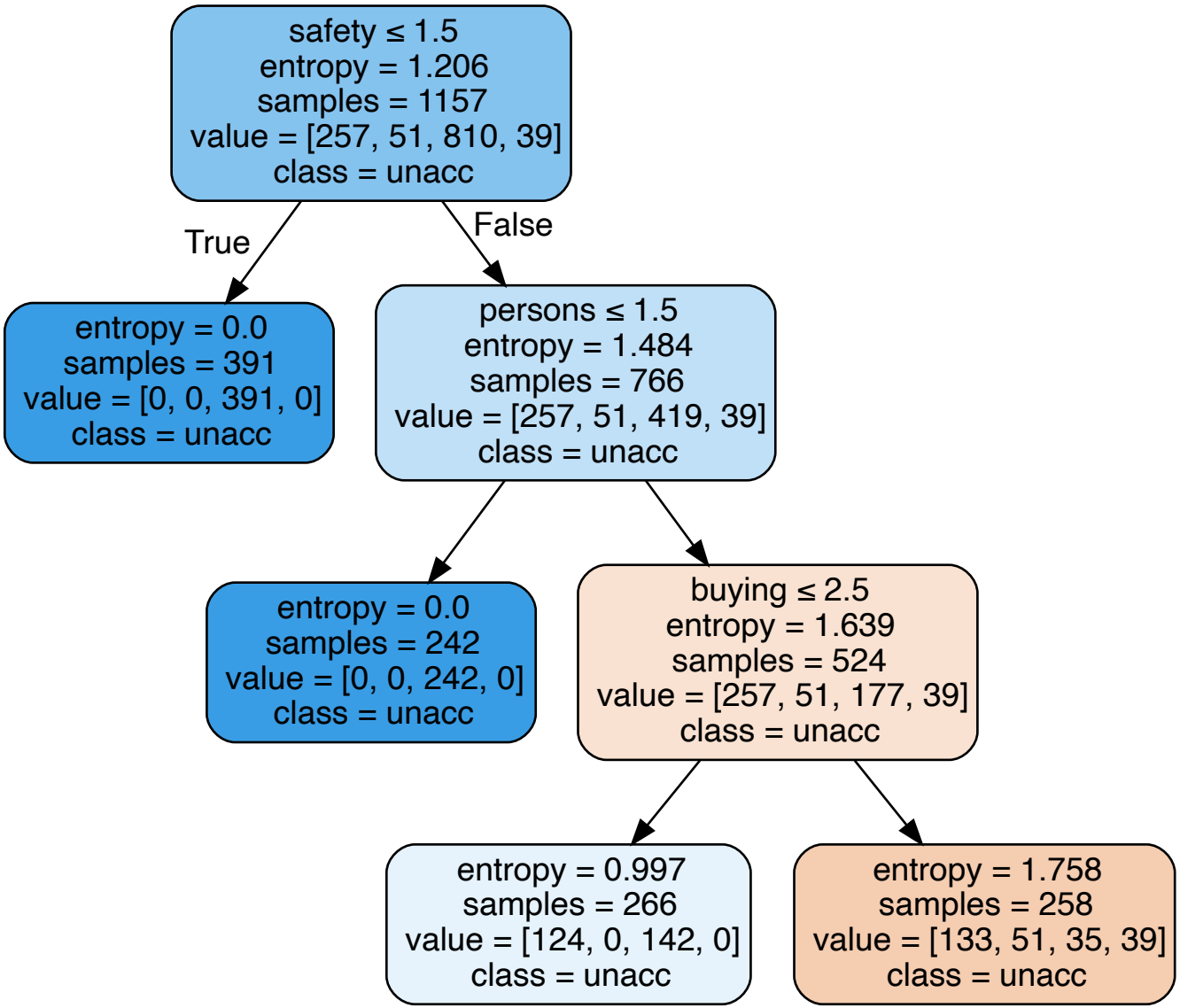

x[5] <= 1.5
entropy = 1.206
samples = 1157
value = [257, 51, 810, 39]
entropy = 0.0
samples = 391
value = [0, 0, 391, 0]
x[3] <= 1.5
entropy = 1.484
samples = 766
value = [257, 51, 419, 39]
entropy = 0.0
samples = 242
value = [0, 0, 242, 0]
x[0] <= 2.5
entropy = 1.639
samples = 524
value = [257, 51, 177, 39]
entropy = 0.997
samples = 266
value = [124, 0, 142, 0]
entropy = 1.758
samples = 258
value = [133, 51, 35, 39]

In [37]:
```
import graphviz
dot_data = tree.export_graphviz(clf_en, 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[37]:


safety ≤ 1.5
entropy = 1.206
samples = 1157
value = [257, 51, 810, 39]
class = unacc
True
False
entropy = 0.0
samples = 391
value = [0, 0, 391, 0]
class = unacc
persons ≤ 1.5
entropy = 1.484
samples = 766
value = [257, 51, 419, 39]
class = unacc
entropy = 0.0
samples = 242
value = [0, 0, 242, 0]
class = unacc
buying ≤ 2.5
entropy = 1.639
samples = 524
value = [257, 51, 177, 39]
class = unacc
entropy = 0.997
samples = 266
value = [124, 0, 142, 0]
class = unacc
entropy = 1.758
samples = 258
value = [133, 51, 35, 39]
class = unacc

In [38]:
```
# 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]]
```

In [39]:
```
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
```