Aryaman Gautam

J001

In [1]:
```
import os
print(os.getcwd())
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

```
C:\Users\Usha Gautam\Desktop\Sem 5\ML\New folder
```

In [2]:
```
df = pd.read_csv('car_evaluation.csv', header = None)
```

In [3]:
```
df.head()
```

Out[3]:

	0	1	2	3	4	5	6
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc

In [4]:
```
col_names = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
df.columns = col_names
col_names
```

Out[4]:
```
['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class']
```

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

Out[5]:

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

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   buying    1728 non-null   object
 1   maint     1728 non-null   object
 2   doors     1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [7]:
```
for i in col_names:
    print(df[i].value_counts())
```

```
med      432
low      432
high     432
vhigh    432
Name: buying, dtype: int64
med      432
low      432
high     432
```

```
vhigh    432
Name: maint, dtype: int64
2        432
5more    432
3        432
4        432
Name: doors, dtype: int64
2       576
more    576
4       576
Name: persons, dtype: int64
med      576
big      576
small    576
Name: lug_boot, dtype: int64
med     576
low     576
high    576
Name: safety, dtype: int64
unacc    1210
acc       384
good       69
vgood      65
Name: class, dtype: int64
```

In [8]:
```
df.shape
```

Out[8]:
```
(1728, 7)
```

In [9]:
```
X = df.drop(['class'],axis = 1)
y = df['class']
```

In [10]:
```
from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(X,y,test_size=0.3,random_state=42)
```

In [11]:
```
from sklearn.preprocessing import OrdinalEncoder
enc = OrdinalEncoder()
X_train = enc.fit_transform(X_train)
X_test = enc.transform((X_test))
```

Gini index as criterion

In [12]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [13]:
```
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_state=42)
clf_gini.fit(X_train, y_train)
```

Out[13]:
```
DecisionTreeClassifier(max_depth=3, random_state=42)
```

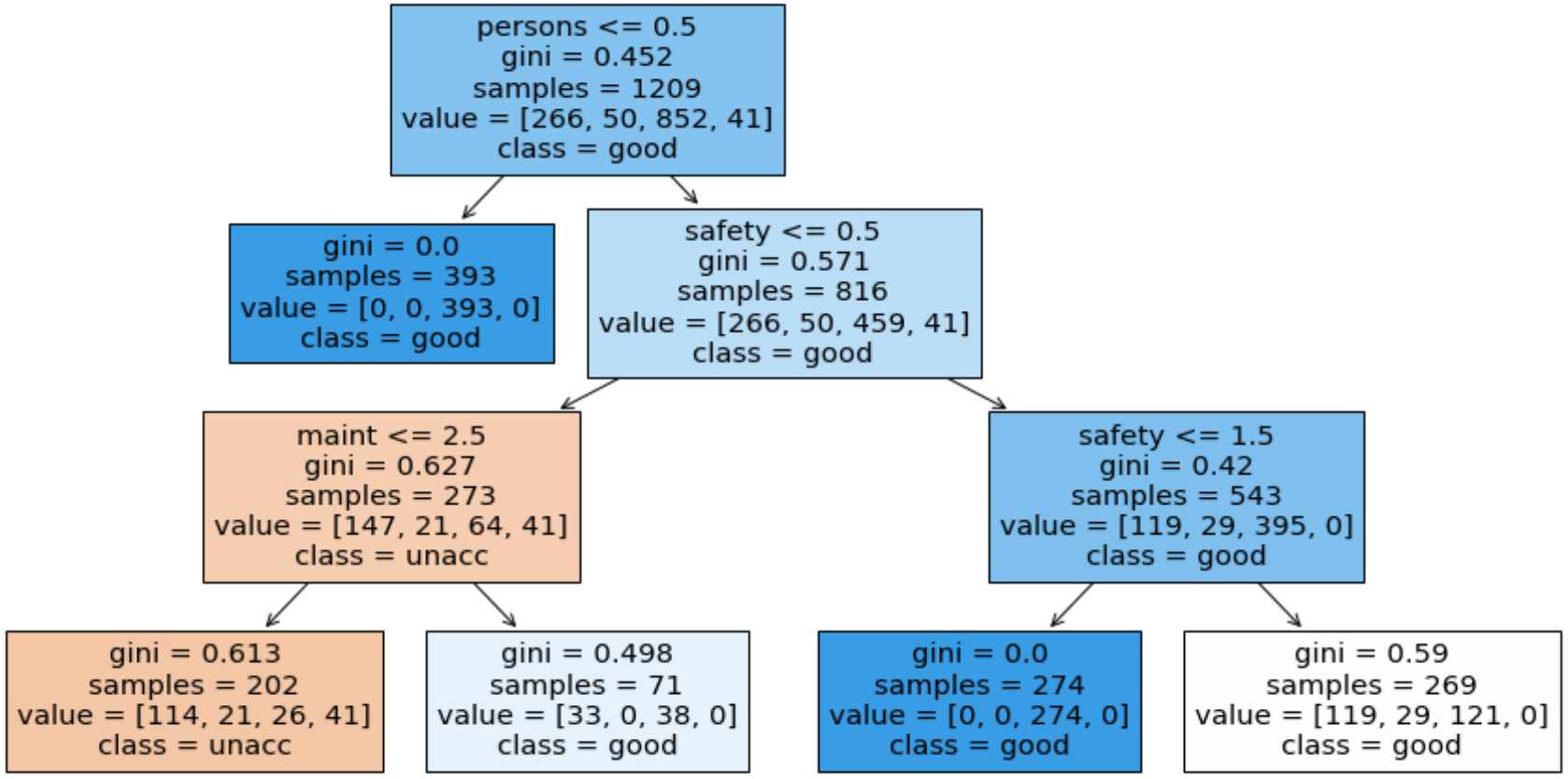
In [14]:
```
y_pred = clf_gini.predict(X_test)
```

In [15]:
```
from sklearn.metrics import accuracy_score

print(f'Model with gini index gives an accuracy of: {accuracy_score(y_test, y_pred)}')
```

```
Model with gini index gives an accuracy of: 0.7572254335260116
```

In [16]:
```
from sklearn import tree
plt.figure(figsize=(15,8))
tree.plot_tree(clf_gini,
               feature_names=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'],
               class_names= list(set(y_train)),
               filled = True)
plt.show()
```



In [17]:

```
# Check for underfitting

print(f'Training set score: {clf_gini.score(X_train,y_train)}')
print(f'Test set score: {clf_gini.score(X_test,y_test)}')
```

```
Training set score: 0.7775020678246485
Test set score: 0.7572254335260116
```

Entropy as criterion

In [18]:

```
clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
```

```
clf_entropy.fit(X_train, y_train)
```

Out[18]:
```
DecisionTreeClassifier(criterion='entropy', max_depth=3, random_state=42)
```

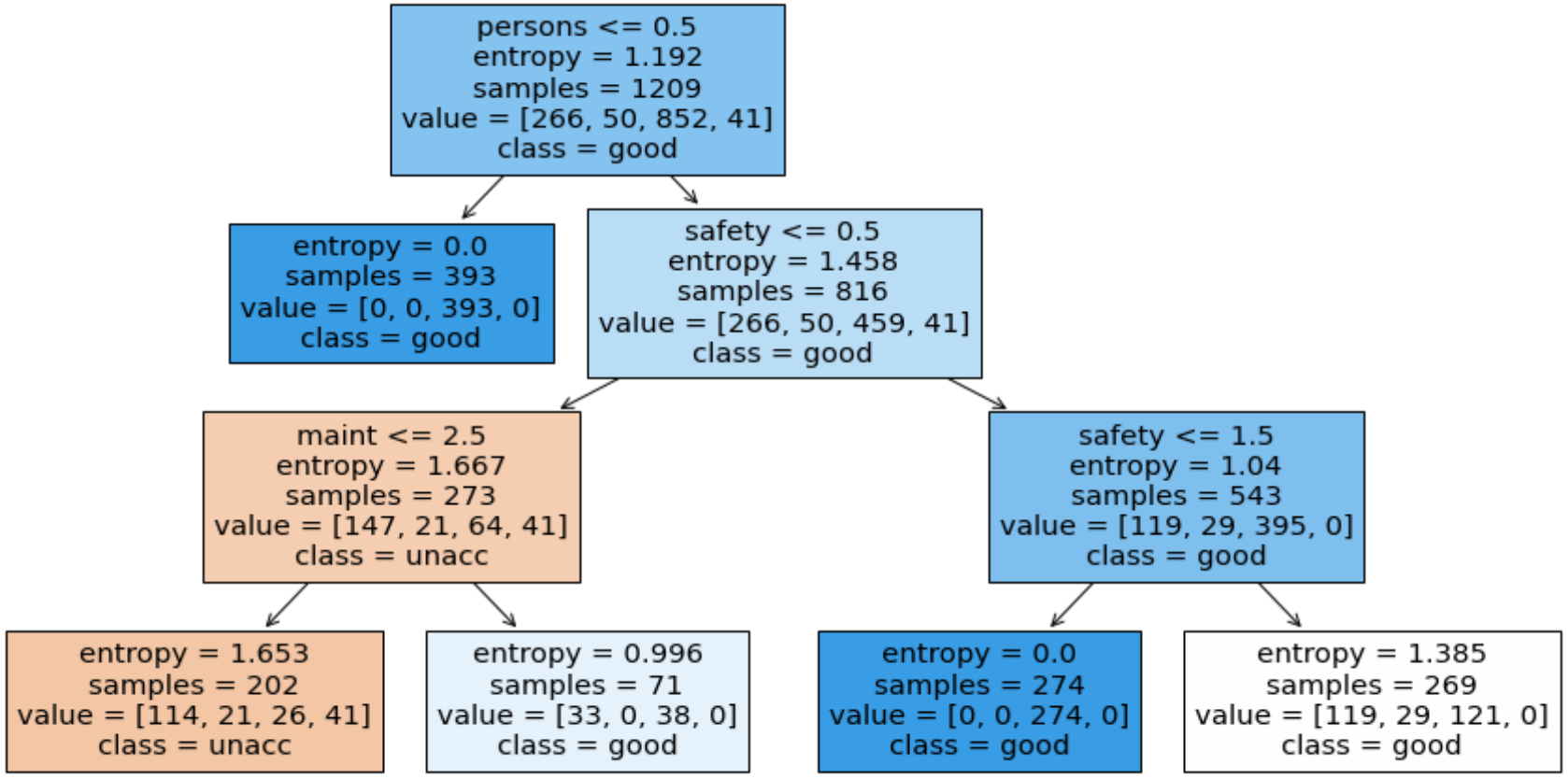
In [19]:
```
y_pred = clf_entropy.predict(X_test)
```

In [20]:
```
from sklearn.metrics import accuracy_score

print(f'Model with gini index gives an accuracy of: {accuracy_score(y_test, y_pred)}')
```

```
Model with gini index gives an accuracy of: 0.7572254335260116
```

In [21]:
```
plt.figure(figsize=(15,8))
tree.plot_tree(clf_entropy,
               feature_names=['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety'],
               class_names= list(set(y_train)),
               filled = True)
plt.show()
```



In [22]:

```
# Check for underfitting

print(f'Training set score: {clf_entropy.score(X_train,y_train)}')
print(f'Test set score: {clf_entropy.score(X_test,y_test)}')
```

```
Training set score: 0.7775020678246485
Test set score: 0.7572254335260116
```

In [23]:

```
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, y_pred)
```

In [24]:
```
print(cm)
```

```
[[ 44   0  74   0]
 [  9   0  10   0]
 [  9   0 349   0]
 [ 24   0   0   0]]
```

In [25]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         acc       0.51      0.37      0.43       118
        good       0.00      0.00      0.00        19
       unacc       0.81      0.97      0.88       358
       vgood       0.00      0.00      0.00        24

    accuracy                           0.76       519
   macro avg       0.33      0.34      0.33       519
weighted avg       0.67      0.76      0.71       519
```

```
C:\Users\Usha Gautam\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Usha Gautam\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
C:\Users\Usha Gautam\anaconda3\lib\site-packages\sklearn\metrics\_classification.py:1245: UndefinedMetricWarning: Pre
cision and F-score are ill-defined and being set to 0.0 in labels with no predicted samples. Use `zero_division` para
meter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

Grid Search CV

In [34]:
```
from sklearn.model_selection import train_test_split
from sklearn.svm import SVC # SVM model with kernels
from sklearn.model_selection import GridSearchCV

from sklearn.metrics import accuracy_score

from sklearn.metrics import confusion_matrix
from sklearn.metrics import classification_report
```

In [27]:
```
title_list = ['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class value']

car = pd.read_csv("car_evaluation.csv", names=title_list, index_col=None)

car
```

Out[27]:

	buying	maint	doors	persons	lug_boot	safety	class value
0	vhigh	vhigh	2	2	small	low	unacc
1	vhigh	vhigh	2	2	small	med	unacc
2	vhigh	vhigh	2	2	small	high	unacc
3	vhigh	vhigh	2	2	med	low	unacc
4	vhigh	vhigh	2	2	med	med	unacc
...	...	...	...	...	...	...	...
1723	low	low	5more	more	med	med	good
1724	low	low	5more	more	med	high	vgood

	buying	maint	doors	persons	lug_boot	safety	class value
1725	low	low	5more	more	big	low	unacc
1726	low	low	5more	more	big	med	good
1727	low	low	5more	more	big	high	vgood

1728 rows × 7 columns

In [30]:
```
car.describe(),print("\n\n\n"),car.info(),print("\n\n\n"),car.shape
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1728 entries, 0 to 1727
Data columns (total 7 columns):
 #   Column       Non-Null Count  Dtype 
---  ------       --------------  ----- 
 0   buying       1728 non-null   object
 1   maint        1728 non-null   object
 2   doors        1728 non-null   object
 3   persons      1728 non-null   object
 4   lug_boot     1728 non-null   object
 5   safety       1728 non-null   object
 6   class value  1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

Out[30]:
```
(       buying maint doors persons lug_boot safety class value
 count    1728  1728  1728    1728     1728   1728        1728
 unique      4     4     4       3        3      3           4
 top       med   med     2       2      med    med       unacc
 freq      432   432   432     576      576    576        1210,
 None,
 None,
 None,
 (1728, 7))
```

In [31]:
```
X = car.drop(['class value'], axis=1)
y = car['class value']

X, y
```

Out[31]:
```
(     buying  maint  doors persons lug_boot safety
 0     vhigh  vhigh      2       2    small    low
 1     vhigh  vhigh      2       2    small    med
 2     vhigh  vhigh      2       2    small   high
 3     vhigh  vhigh      2       2      med    low
 4     vhigh  vhigh      2       2      med    med
 ...     ...    ...    ...     ...      ...    ...
 1723    low    low  5more    more      med    med
 1724    low    low  5more    more      med   high
 1725    low    low  5more    more      big    low
 1726    low    low  5more    more      big    med
 1727    low    low  5more    more      big   high

 [1728 rows x 6 columns],
 0       unacc
 1       unacc
 2       unacc
 3       unacc
 4       unacc
         ...
 1723     good
 1724    vgood
 1725    unacc
 1726     good
 1727    vgood
 Name: class value, Length: 1728, dtype: object)
```

In [32]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size = 0.4, random_state = 42)
```

In [37]:
```
columns_encode = []
columns_encode.append(title_list)
columns_encode
```

Out[37]:
```
[['buying', 'maint', 'doors', 'persons', 'lug_boot', 'safety', 'class value']]
```

In [38]:
```
ordinal_encoder = OrdinalEncoder()

X_train = ordinal_encoder.fit_transform(X_train, columns_encode)
X_test = ordinal_encoder.transform(X_test)
```

In [44]:
```
svm = SVC()
```

In [46]:
```
parameters = {'criterion':['gini','entropy'],
              'max_depth':[1,2,3,4,5,6,7,8,9,10]
             }

default_tr = tree.DecisionTreeClassifier(random_state=42)

gs_tree = GridSearchCV(default_tr, parameters, cv=10, n_jobs=-1,verbose=1)


gs_tree.fit(X_train,y_train)
```

```
Fitting 10 folds for each of 20 candidates, totalling 200 fits
```

Out[46]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(random_state=42),
             n_jobs=-1,
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]},
             verbose=1)
```

In [47]:
```
gs_tree_pred=gs_tree.predict(X_test)
```

In [48]:
```
gs_tree.best_params_
```

Out[48]:
```
{'criterion': 'entropy', 'max_depth': 10}
```

In [51]:
```
gs_tree.best_score_
```

Out[51]:
```
0.9710604929051531
```

In [52]:
```
gs_tree.score(X_test,y_test)
```

Out[52]:
```
0.9595375722543352
```

In [53]:
```
confusion_matrix(y_test,gs_tree_pred)
```

Out[53]:
```
array([[144,   9,   0,   3],
       [  0,  26,   0,   3],
       [  8,   0, 472,   0],
       [  3,   2,   0,  22]], dtype=int64)
```

In [ ]:
```

```