Decision Tree Grid Search CV:

KABIR CHATURVEDI, J011.

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

In [2]:
```
df = pd.read_csv('car_evaluation.csv')
```

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

Out[3]:

	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 [4]:
```
col_names = ['buying','maint','doors','people','lugg_boot','safety','class']
```

In [5]:
```
df.columns = col_names
df.head() #we rename the columns for simplicity.
```

Out[5]:

	buying	maint	doors	people	lugg_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 [6]:
```
df.describe()
```

Out[6]:

	buying	maint	doors	people	lugg_boot	safety	class
count	1727	1727	1727	1727	1727	1727	1727
unique	4	4	4	3	3	3	4
top	high	high	3	more	med	high	unacc
freq	432	432	432	576	576	576	1209

In [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   people      1727 non-null   object
 4   lugg_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 [8]:
```
df.isna().sum() #we check whether there are any NA values or not.
```

Out[8]:

```
buying      0
maint       0
doors       0
people      0
lugg_boot   0
safety      0
class       0
dtype: int64
```

In [9]:
```
df.shape
```

Out[9]:

```
(1727, 7)
```

In [10]:
```
X = df.drop('class',axis=1)
y = df['class']
```

In [11]:
```
from sklearn.model_selection import train_test_split
```

In [12]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_state=101)
```

In [13]:
```
from sklearn.preprocessing import OrdinalEncoder
encoder = OrdinalEncoder()
```

In [14]:
```
X_train = encoder.fit_transform(X_train)
X_test = encoder.transform(X_test)
```

Building the model:

In [15]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [16]:
```
model = DecisionTreeClassifier(max_depth=5,random_state=101)
model.fit(X_train,y_train)
```

Out[16]:

```
DecisionTreeClassifier(max_depth=5, random_state=101)
```

In [17]:
```
y_pred = model.predict(X_test)
```

Now we use GridSearchCV :

In [18]:
```
from sklearn.model_selection import GridSearchCV
```

In [19]:
```
# we make a dictionary to input in the gridsearchcv function:
criterion = ["gini", "entropy"]

param_grid = {'criterion':criterion, 'max_depth':[3,4,5,6,7], 'max_features':[2,3,4,5,6],
              'min_samples_leaf':[1,2], 'min_samples_split':[2,3,4,5]}

model2 = GridSearchCV(model,param_grid,cv=4,scoring='accuracy')
```

In [37]:
```
model2.fit(X_train,y_train)
```

Out[37]:

```
GridSearchCV(cv=4,
             estimator=DecisionTreeClassifier(max_depth=5, random_state=101),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [3, 4, 5, 6, 7],
                         'max_features': [2, 3, 4, 5, 6],
                         'min_samples_leaf': [1, 2],
                         'min_samples_split': [2, 3, 4, 5]},
             scoring='accuracy')
```

In [38]:
```
model2.best_params_
```

Out[38]:

```
{'criterion': 'entropy',
 'max_depth': 7,
 'max_features': 6,
 'min_samples_leaf': 1,
 'min_samples_split': 2}
```

In [22]:
```
model2.best_score_
```

Out[22]:

```
0.9248031261186016
```

In [23]:
```
model.score(X_train,y_train)
```

Out[23]:

```
0.8582541054451167
```

In [24]:
```
model.score(X_test,y_test)
```

Out[24]:

```
0.8596491228070176
```

Final model:

In [25]:
```
final_model = DecisionTreeClassifier(criterion='entropy',max_depth=7, max_features=6, min_samples_leaf=1, min_samples_split=2)
```

In [26]:
```
final_model.fit(X_train,y_train)
```

Out[26]:

```
DecisionTreeClassifier(criterion='entropy', max_depth=7, max_features=6)
```

In [27]:
```
y_preds = final_model.predict(X_test)
```

In [28]:
```
final_model.score(X_train,y_train)
```

Out[28]:

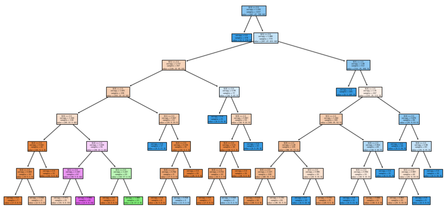
```
0.9325842696629213
```

In [29]:
```
final_model.score(X_test,y_test)
```

Out[29]:

```
0.9245614035087719
```

In [31]:
```
from sklearn import tree
plt.figure(figsize=(16,8))
tree.plot_tree(final_model,filled=True)
plt.show()
```



Cross Validation:

In [32]:
```
from sklearn.model_selection import cross_val_score
```

In [33]:
```
score = cross_val_score(final_model,X_train,y_train,cv=10,scoring='accuracy')
score.mean()
```

Out[33]:

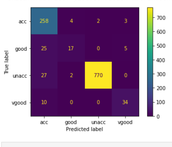
```
0.9256559670164917
```

In [34]:
```
from sklearn.metrics import plot_confusion_matrix, classification_report
```

In [35]:
```
plot_confusion_matrix(final_model,X_train,y_train)
```

Out[35]:

```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x154d1d549a0>
```



In [36]:
```
print(classification_report(y_test,y_preds))
```

```
              precision    recall  f1-score   support

         acc       0.78      0.96      0.86       117
        good       0.44      0.18      0.26        22
       unacc       0.99      0.96      0.98       410
       vgood       0.82      0.86      0.84        21

    accuracy                           0.92       570
   macro avg       0.76      0.74      0.73       570
weighted avg       0.92      0.92      0.92       570
```