Name: Manasi Amdekar

Roll No.: J003

Exp 9 Decision Tree with Cross Validation and GridSearchCV

In [2]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import os
print(os.getcwd())
%matplotlib inline
```

```
C:\Users\mvamd\ML
```

In [3]:
```
df = pd.read_csv("D:\\Sem 5\\ML\\car_evaluation.csv", header = None)
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]:
```
for i in col_names:
    print(df[i].value_counts())
```

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

In [7]:
```
df.shape
```

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

In [8]:
```
X = df.drop(['class'],axis = 1)
y = df['class']
```

In [9]:
```
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,ra
ndom_state=42)
```

In [10]:
```
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 [11]:
```
from sklearn.tree import DecisionTreeClassifier
```

In [12]:
```
clf_gini = DecisionTreeClassifier(criterion='gini', max_depth=3, random_
state=42)
clf_gini.fit(X_train, y_train)
```

Out[12]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='gin
i',
                       max_depth=3, max_features=None, max_leaf_nodes=No
ne,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=42, splitter='best')
```

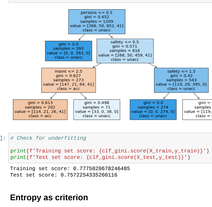
In [13]:
```
y_pred = clf_gini.predict(X_test)
```

In [14]:
```
from sklearn.metrics import accuracy_score

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

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

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



In [16]:
```
# 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 [17]:
```
clf_entropy = DecisionTreeClassifier(criterion='entropy', max_depth=3, r
andom_state=41)
clf_entropy.fit(X_train, y_train)
```

Out[17]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entr
opy',
                       max_depth=3, max_features=None, max_leaf_nodes=No
ne,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=1, min_samples_split=2,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=42, splitter='best')
```

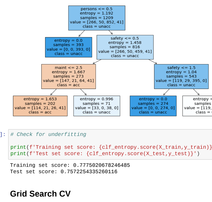
In [18]:
```
y_pred = clf_entropy.predict(X_test)
```

In [19]:
```
from sklearn.metrics import accuracy_score

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

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

In [20]:
```
plt.figure(figsize=(15,8))
tree.plot_tree(clf_entropy,
               feature_names=['buying', 'maint', 'doors', 'persons', 'lu
g_boot', 'safety'],
               class_names= list(set(y_train)),
               filled = True)
plt.show()
```



In [21]:
```
# 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
```

Grid Search CV

In [22]:
```
from sklearn.model_selection import GridSearchCV
```

In [23]:
```
params = {'criterion': ['gini','entropy'],'max_depth': list(range(3,7)),
'min_samples_split': list(range(2,7)),'min_samples_leaf': list(range(1,7
)),'max_leaf_nodes': list(range(3,12)) }
```

In [24]:
```
decision_tree = DecisionTreeClassifier()
dt = GridSearchCV(decision_tree, params, cv=10, scoring = 'accuracy')
dt.fit(X_train,y_train)
dt.best_score_
```

Out[24]:
```
0.8461988154269971
```

In [25]:
```
dt.best_params_
```

Out[25]:
```
{'criterion': 'entropy',
 'max_depth': 6,
 'max_leaf_nodes': 11,
 'min_samples_leaf': 3,
 'min_samples_split': 3}
```

In [26]:
```
y_pred  = dt.predict(X_test)
```

In [27]:
```
print(f'Model with Decision Tree gives an accuracy of: {accuracy_score(y
_test, y_pred)}')
```

```
Model with Decision Tree gives an accuracy of: 0.8612716763005780
```

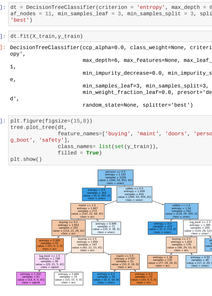
After GridSearchCV

In [30]:
```
dt = DecisionTreeClassifier(criterion = 'entropy', max_depth = 6, max_le
af_nodes = 11, min_samples_leaf = 3, min_samples_split = 3, splitter =
'best')
```

In [31]:
```
dt.fit(X_train,y_train)
```

Out[31]:
```
DecisionTreeClassifier(ccp_alpha=0.0, class_weight=None, criterion='entr
opy',
                       max_depth=6, max_features=None, max_leaf_nodes=1
1,
                       min_impurity_decrease=0.0, min_impurity_split=Non
e,
                       min_samples_leaf=3, min_samples_split=3,
                       min_weight_fraction_leaf=0.0, presort='deprecate
d',
                       random_state=None, splitter='best')
```

In [32]:
```
plt.figure(figsize=(15,8))
tree.plot_tree(dt,
               feature_names=['buying', 'maint', 'doors', 'persons', 'lu
g_boot', 'safety'],
               class_names= list(set(y_train)),
               filled = True)
plt.show()
```



Cross Validation

In [33]:
```
from sklearn.model_selection import cross_val_score
```

In [34]:
```
score = cross_val_score(dt,X_train, y_train, cv=10, scoring = 'accuracy'
)
score.mean()
```

Out[34]:
```
0.8461988154269971
```

In [35]:
```
# Check for underfitting

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

```
Training set score: 0.8585607940446665
Test set score: 0.8612716763005780
```

In [ ]:

In [36]:
```
from sklearn.metrics import confusion_matrix, classification_report
cm = confusion_matrix(y_test, y_pred)
```

In [37]:
```
print(cm)
```

```
[[ 78   0  32   8]
 [ 16   0   0   3]
 [ 13   0 345   0]
 [  0   0   0  24]]
```

In [38]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         acc       0.73      0.66      0.69       118
        good       0.00      0.00      0.00        19
       unacc       0.92      0.96      0.94       358
       vgood       0.69      1.00      0.81        24

    accuracy                           0.86       519
   macro avg       0.58      0.66      0.61       519
weighted avg       0.83      0.86      0.84       519
```

```
C:\Users\mvamd\anaconda3\lib\site-packages\sklearn\metrics\_classificati
on.py:1272: UndefinedMetricWarning: Precision and F-score are ill-define
d and being set to 0.0 in labels with no predicted samples. Use `zero_di
vision` parameter to control this behavior.
  _warn_prf(average, modifier, msg_start, len(result))
```