Decision Trees and Cross Validation

-Aaryansh Sahay J052

Importing Libraries

In [112]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
%matplotlib inline
```

In [87]:
```
from sklearn.model_selection import train_test_split, GridSearchCV, cross_val_score
from sklearn.preprocessing import OrdinalEncoder
from sklearn.tree import DecisionTreeClassifier, plot_tree
from sklearn.metrics import accuracy_score, confusion_matrix, classification_report
```

In [88]:
```
import warnings
warnings.filterwarnings('ignore')
```

Data Processing and Preperation

In [89]:
```
data=pd.read_csv('car_evaluation.csv')
data.head()
```

Out[89]:

	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 [90]:
```
col_names=['buying','maint','doors','persons','lug_boot','safety','class']
data.columns=col_names
data.head()
```

Out[90]:

	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 [91]:
```
data.describe().T
```

Out[91]:

	count	unique	top	freq
buying	1727	4	med	432
maint	1727	4	med	432
doors	1727	4	4	432
persons	1727	3	4	576
lug_boot	1727	3	med	576
safety	1727	3	med	576
class	1727	4	unacc	1209

In [92]:
```
for i in col_names:
    print(data[i].value_counts())
    print('')
```

```
med      432
high     432
low      432
vhigh    431
Name: buying, dtype: int64

med      432
high     432
low      432
vhigh    431
Name: maint, dtype: int64

4        432
5more    432
3        432
2        431
Name: doors, dtype: int64
```

```
4       576
more    576
2       575
Name: persons, dtype: int64

med      576
big      576
small    575
Name: lug_boot, dtype: int64

med     576
high    576
low     575
Name: safety, dtype: int64

unacc    1209
acc       384
good       69
vgood      65
Name: class, dtype: int64
```

In [93]:
```
data.shape
```

Out[93]:
```
(1727, 7)
```

In [94]:
```
X=data.drop('class',axis=1)
y=data['class']
```

In [95]:
```
enc=OrdinalEncoder()
X=enc.fit_transform(X)
```

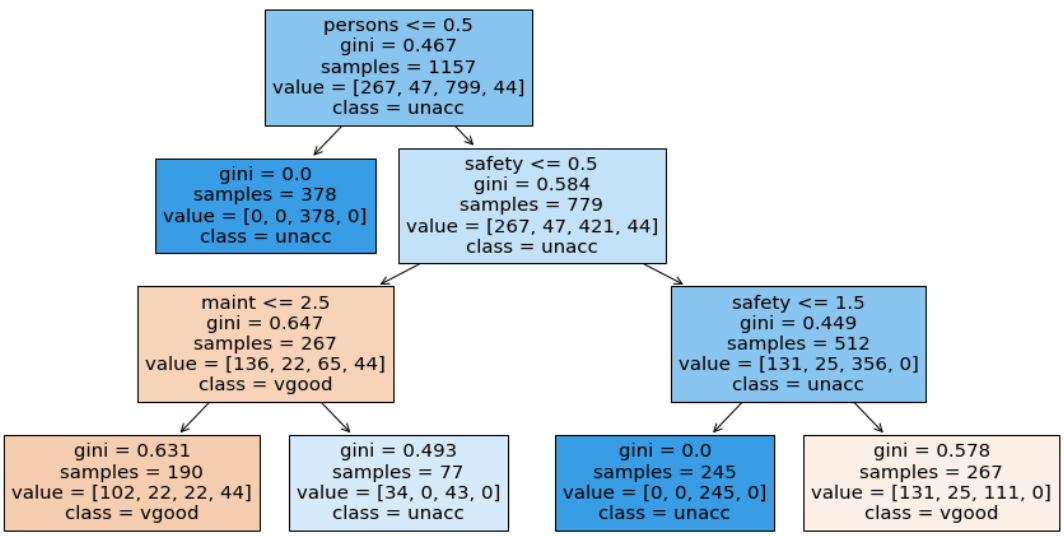
In [96]:
```
X_train,X_test,y_train,y_test=train_test_split(X,y,test_size=0.33,random_state=101)
```

Decision Tree model with gini criterion

In [97]:
```
clf=DecisionTreeClassifier(criterion='gini',max_depth=3,random_state=101)
clf.fit(X_train,y_train)
y_preds=clf.predict(X_test)
train_score=clf.score(X_train,y_train)
test_score=accuracy_score(y_test,y_preds)
print('Train Accuracy : {}\nTest Accuracy : {}'.format(train_score,test_score))
```

```
Train Accuracy : 0.777009507346586
Test Accuracy : 0.7859649122807018
```

In [98]:
```
plt.figure(figsize=(15,8))
plot_tree(clf,
          feature_names=col_names[:-1],
          class_names=list(set(y_train)),
          filled=True)
plt.show()
```



In [99]:
```
report1= confusion_matrix(y_test,y_preds)
report2=classification_report(y_test,y_preds)
print('Confusion Matrix is :\n\n{}\n\nClassification Report :\n\n{}'.format(report1,report2))
```

```
Confusion Matrix is :

[[105   0  12   0]
 [ 22   0   0   0]
 [ 67   0 343   0]
 [ 21   0   0   0]]
```

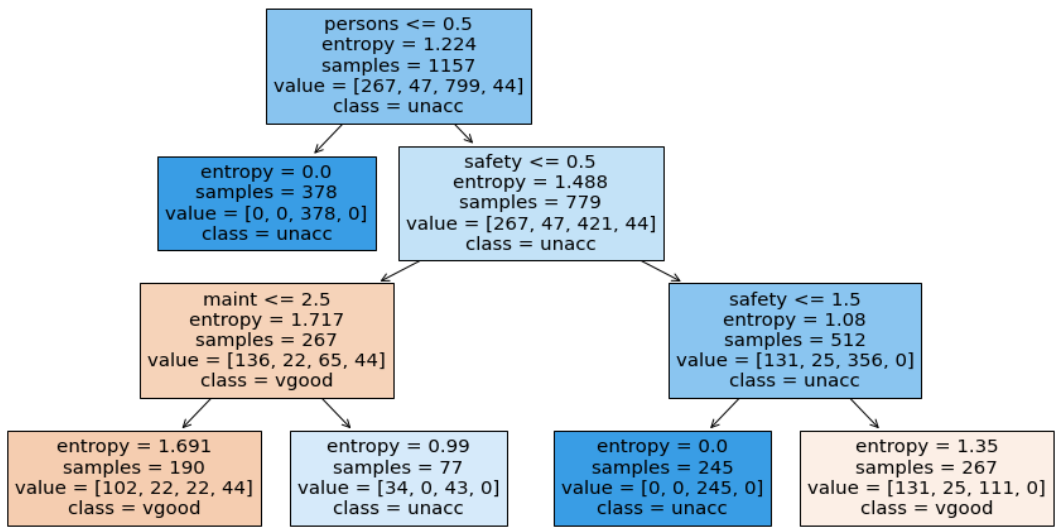
```
Classification Report :

              precision    recall  f1-score   support

         acc       0.49      0.90      0.63       117
        good       0.00      0.00      0.00        22
       unacc       0.97      0.84      0.90       410
       vgood       0.00      0.00      0.00        21

    accuracy                           0.79       570
   macro avg       0.36      0.43      0.38       570
weighted avg       0.80      0.79      0.77       570
```

Decision tree model with entropy criterion

In [100]:
```
clf=DecisionTreeClassifier(criterion='entropy',max_depth=3,random_state=101)
clf.fit(X_train,y_train)
y_preds=clf.predict(X_test)
train_score=clf.score(X_train,y_train)
test_score=accuracy_score(y_test,y_preds)
print('Train Accuracy : {}\nTest Accuracy : {}'.format(train_score,test_score))
```

```
Train Accuracy : 0.777009507346586
Test Accuracy : 0.7859649122807018
```

In [101]:
```
plt.figure(figsize=(15,8))
plot_tree(clf,
          feature_names=col_names[:-1],
          class_names=list(set(y_train)),
          filled=True)
plt.show()
```



In [102]:
```
report1= confusion_matrix(y_test,y_preds)
report2=classification_report(y_test,y_preds)
print('Confusion Matrix is :\n\n{}\n\nClassification Report :\n\n{}'.format(report1,report2))
```

```
Confusion Matrix is :

[[105   0  12   0]
 [ 22   0   0   0]
 [ 67   0 343   0]
 [ 21   0   0   0]]

Classification Report :

              precision    recall  f1-score   support

         acc       0.49      0.90      0.63       117
        good       0.00      0.00      0.00        22
       unacc       0.97      0.84      0.90       410
       vgood       0.00      0.00      0.00        21

    accuracy                           0.79       570
   macro avg       0.36      0.43      0.38       570
weighted avg       0.80      0.79      0.77       570
```

Hyperparamter optimization

In [103]:
```
#Grid Search
params={'criterion':['gini','entropy'],'max_features':[1,2,3,4,5,6,None],'max_depth':[1,2,3,4,5,6,7],'min_samples_split':[2,3,4,5
grid=GridSearchCV(DecisionTreeClassifier(),params,cv=10,scoring=['accuracy'],refit='accuracy')
grid.fit(X,y)
```

Out[103]:
```
GridSearchCV(cv=10, estimator=DecisionTreeClassifier(),
             param_grid={'criterion': ['gini', 'entropy'],
                         'max_depth': [1, 2, 3, 4, 5, 6, 7],
                         'max_features': [1, 2, 3, 4, 5, 6, None],
                         'min_samples_split': [2, 3, 4, 5]},
             refit='accuracy', scoring=['accuracy'])
```

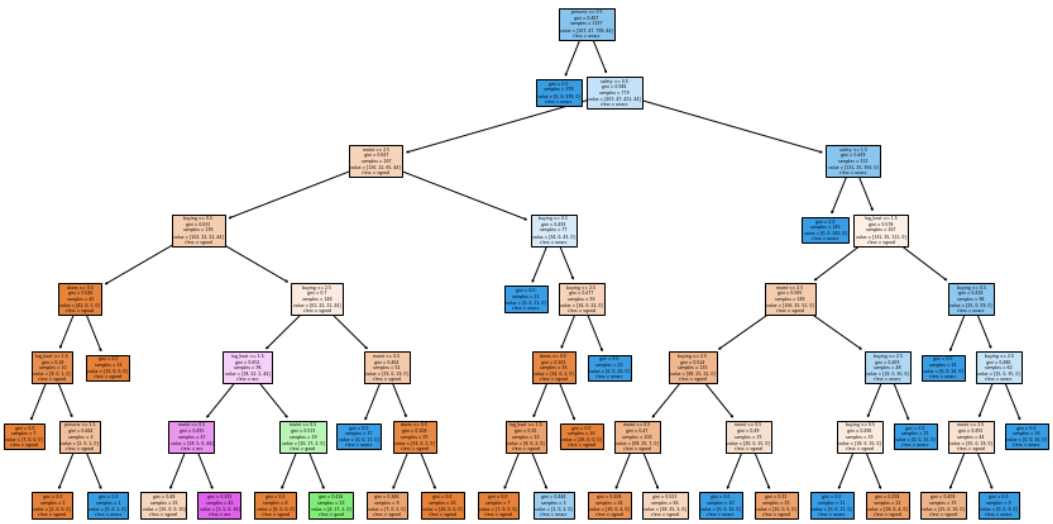
In [104]:
```
print('Best Parameters : \n{}\n\n Score for these parameters : {}'.format(grid.best_params_,grid.best_score_))
```

```
Best Parameters :
{'criterion': 'gini', 'max_depth': 7, 'max_features': 6, 'min_samples_split': 2}

 Score for these parameters : 0.8513106600349509
```

In [105]:
```
clf=DecisionTreeClassifier(criterion='gini',max_depth=7,max_features=6,min_samples_split=2,random_state=101)
clf.fit(X_train,y_train)
y_preds=clf.predict(X_test)
train_score=clf.score(X_train,y_train)
test_score=accuracy_score(y_test,y_preds)
print('Train Accuracy : {}\nTest Accuracy : {}'.format(train_score,test_score))
```

```
Train Accuracy : 0.9325842696629213
Test Accuracy : 0.9298245614035088
```

In [106]:
```
plt.figure(figsize=(15,8))
plot_tree(clf,
          feature_names=col_names[:-1],
          class_names=list(set(y_train)),
          filled=True)
plt.show()
```



In [107]:
```
report1= confusion_matrix(y_test,y_preds)
report2=classification_report(y_test,y_preds)
print('Confusion Matrix is :\n\n{}\n\nClassification Report :\n\n{}'.format(report1,report2))
```

```
Confusion Matrix is :

[[113   3   1   0]
 [ 14   4   0   4]
 [ 13   2 395   0]
 [  3   0   0  18]]

Classification Report :

              precision    recall  f1-score   support

         acc       0.79      0.97      0.87       117
        good       0.44      0.18      0.26        22
       unacc       1.00      0.96      0.98       410
       vgood       0.82      0.86      0.84        21

    accuracy                           0.93       570
   macro avg       0.76      0.74      0.74       570
weighted avg       0.93      0.93      0.92       570
```

In [111]:
```
train_score=cross_val_score(clf,X_train,y_train,cv=10,scoring='accuracy')
test_score=cross_val_score(clf,X_test,y_test,cv=10,scoring='accuracy')
train_score,test_score=train_score.mean(),test_score.mean()
print('Score on Training Data : {}\nScore on Testing Data : {}'.format(train_score,test_score))
```

```
Score on Training Data : 0.9179160419790104
Score on Testing Data : 0.9017543859649123
```