```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

```
df=pd.read_csv('/content/car[1].csv')
df
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety	Car_Acceptability
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
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

```
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_Price       1728 non-null   object
 1   Maintenance_Price  1728 non-null   object
 2   No_of_Doors        1728 non-null   object
 3   Person_Capacity    1728 non-null   object
 4   Size_of_Luggage    1728 non-null   object
 5   Safety             1728 non-null   object
 6   Car_Acceptability  1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

```
df.shape
```

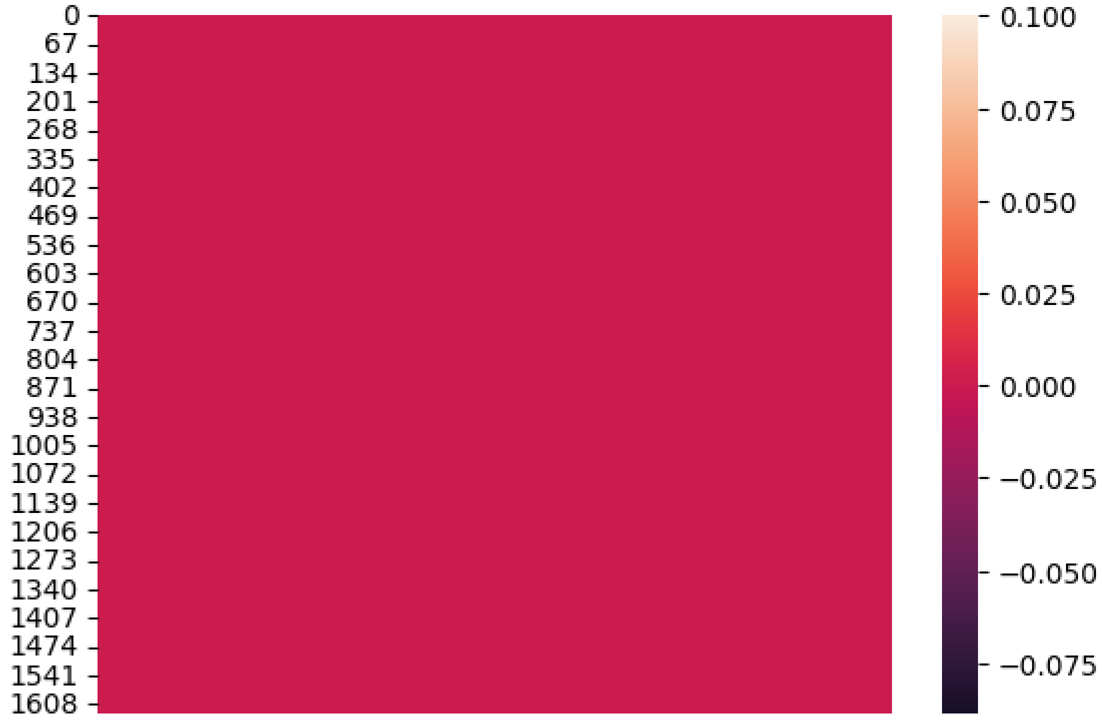
```
(1728, 7)
```

```
df.isna().sum()
```

```
Buying_Price         0
Maintenance_Price    0
No_of_Doors          0
Person_Capacity      0
Size_of_Luggage      0
Safety               0
Car_Acceptability    0
dtype: int64
```

```
sns.heatmap(df.isna())
```

```
<Axes: >
```


0
67
134
201
268
335
402
469
536
603
670
737
804
871
938
1005
1072
1139
1206
1273
1340
1407
1474
1541
1608
0.100
0.075
0.050
0.025
0.000
−0.025
−0.050
−0.075

```
df.head(10)
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety	Car_Acceptability
0	vhigh	vhigh	2	2	small	low	unacc
5	vhigh	vhigh	2	2	med	high	unacc

```
df['Buying_Price'].value_counts()
```

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

```
df['Person_Capacity'].value_counts()
```

```
2       576
4       576
more    576
Name: Person_Capacity, dtype: int64
```

```
df['Safety'].value_counts()
```

```
low     576
med     576
high    576
Name: Safety, dtype: int64
```

```
df['No_of_Doors'].value_counts()
```

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

```
df['Maintenance_Price'].value_counts()
```

```
vhigh    432
high     432
```

```
med      432
low      432
Name: Maintenance_Price, dtype: int64
```

```
column=df.columns
column
```

```
Index(['Buying_Price', 'Maintenance_Price', 'No_of_Doors', 'Person_Capacity',
       'Size_of_Luggage', 'Safety', 'Car_Acceptability'],
      dtype='object')
```

```
for column in (df.columns):
    print ("Unique columns of", column, "\n", df[column].unique())
    print ("--------------")
```

```
Unique columns of Buying_Price
 ['vhigh' 'high' 'med' 'low']
--------------
Unique columns of Maintenance_Price
 ['vhigh' 'high' 'med' 'low']
--------------
Unique columns of No_of_Doors
 ['2' '3' '4' '5more']
--------------
Unique columns of Person_Capacity
 ['2' '4' 'more']
--------------
Unique columns of Size_of_Luggage
 ['small' 'med' 'big']
--------------
Unique columns of Safety
 ['low' 'med' 'high']
--------------
Unique columns of Car_Acceptability
 ['unacc' 'acc' 'vgood' 'good']
--------------
```

```
df['Buying_Price'].replace({'low': 0, 'med': 1, 'high': 2, 'vhigh': 3}, inplace = True)
df['Maintenance_Price'].replace({'low': 0, 'med': 1, 'high': 2, 'vhigh': 3}, inplace = True)
```

```
df['No_of_Doors'].replace({'5more': 5}, inplace = True)
df['Person_Capacity'].replace({'more': 5}, inplace = True)
df['Size_of_Luggage'].replace({'small': 0, 'med': 1, 'big': 2}, inplace = True)
df['Safety'].replace({'low': 0, 'med': 1, 'high': 2}, inplace = True)
df['Car_Acceptability'].replace({'unacc': 0, 'acc': 1, 'good': 2, 'vgood': 3}, inplace = True)
```

```
df['Buying_Price']=df['Buying_Price'].astype(int)
df['No_of_Doors'] = df['No_of_Doors'].astype(int)
df['Size_of_Luggage']= df['Size_of_Luggage'].astype(int)
df['Safety']=df['Safety'].astype(int)
df['Person_Capacity']=df['Person_Capacity'].astype(int)
df['Car_Acceptability'] = df['Car_Acceptability'].astype(int)
df['Maintenance_Price']= df['Maintenance_Price'].astype(int)
```

```
df.dtypes
```

```
Buying_Price         int64
Maintenance_Price    int64
No_of_Doors          int64
Person_Capacity      int64
Size_of_Luggage      int64
Safety               int64
Car_Acceptability    int64
dtype: object
```

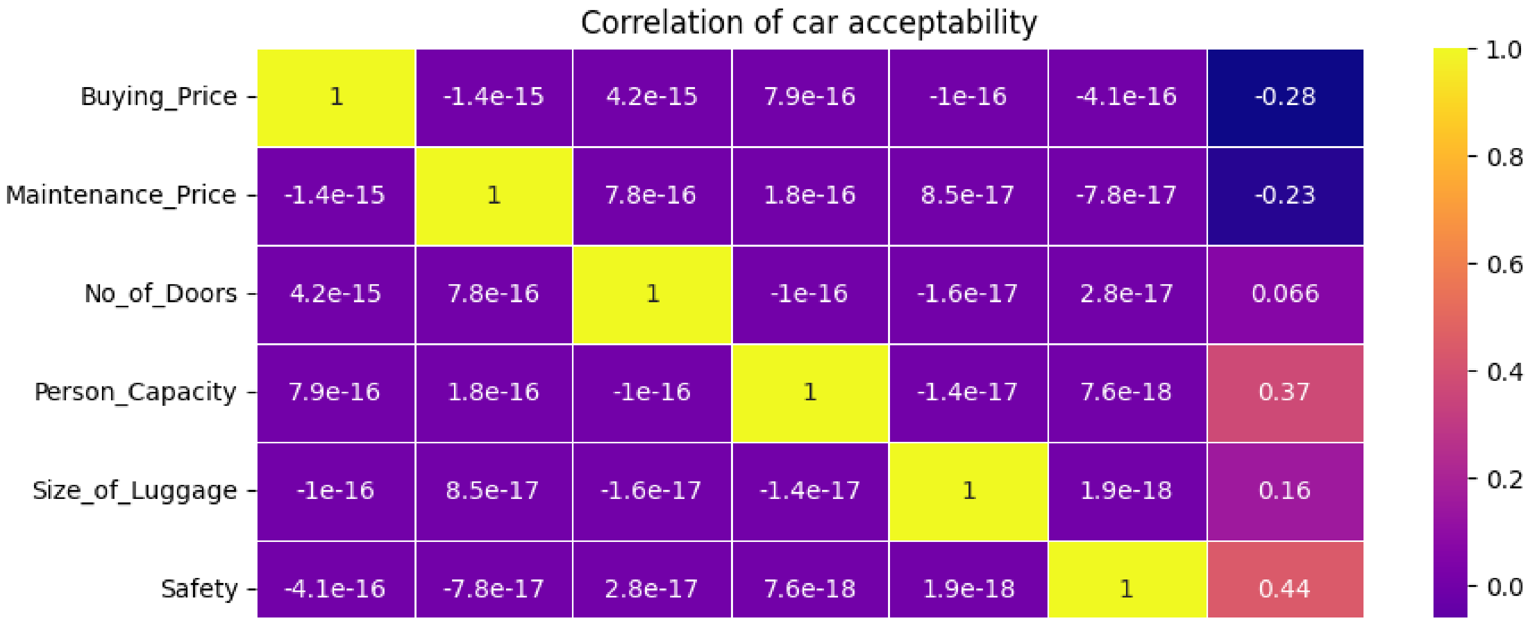
```
df.head(5)
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety	Car_Acceptat
0	3	3	2	2	0	0	
1	3	3	2	2	0	1	
2	3	3	2	2	0	2	
3	3	3	2	2	1	0	
4	3	3	2	2	1	1	

```
corr=df.corr()
corr
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	
Buying_Price	1.000000e+00	-1.356939e-15	4.191709e-15	7.886258e-16	-1.045866e-16	-4
Maintenance_Price	-1.356939e-15	1.000000e+00	7.812681e-16	1.822741e-16	8.544286e-17	-7
No_of_Doors	4.191709e-15	7.812681e-16	1.000000e+00	-9.989138e-17	-1.632846e-17	2.8
Person_Capacity	7.886258e-16	1.822741e-16	-9.989138e-17	1.000000e+00	-1.438481e-17	7.5
Size_of_Luggage	-1.045866e-16	8.544286e-17	-1.632846e-17	-1.438481e-17	1.000000e+00	1.9

```
plt.figure(figsize=(10,5))
sns.heatmap(corr, cmap='plasma', annot=True, linewidth=0.5)
plt.title('Correlation of car acceptability')
plt.show()
```

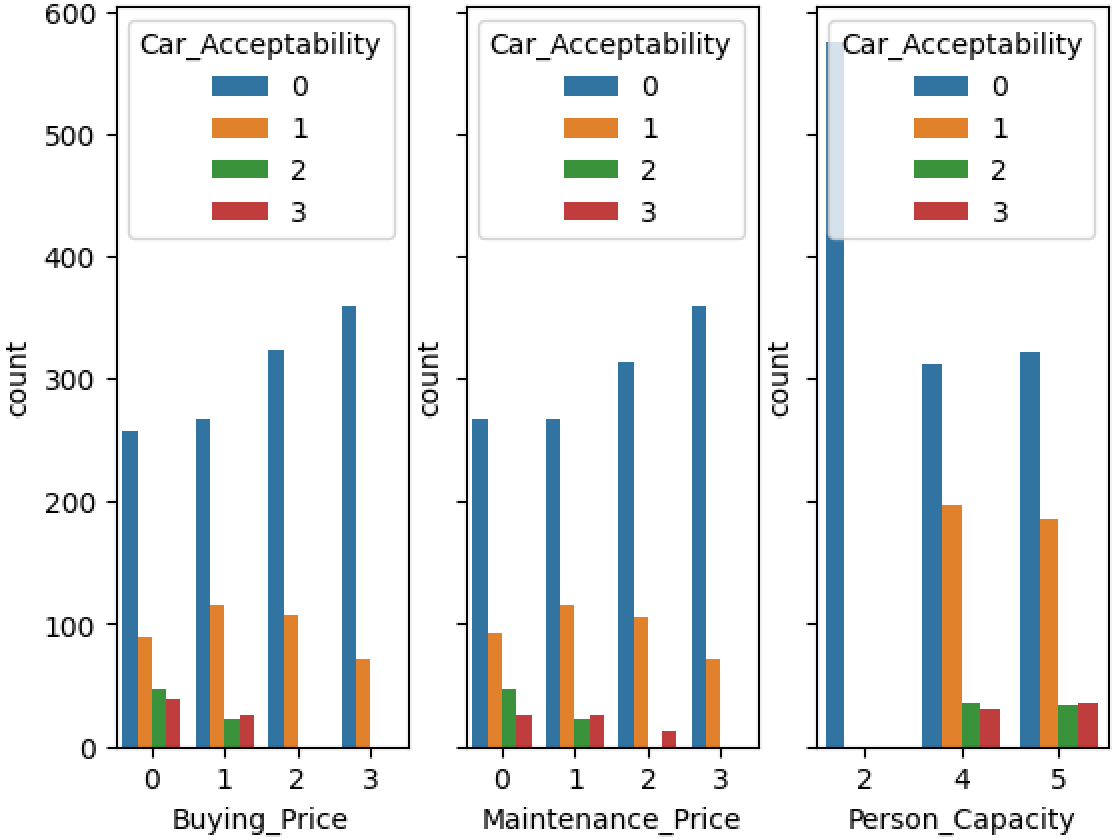


```
sns.countplot(x='Buying_Price',data=df)
```

```
<Axes: xlabel='Buying_Price', ylabel='count'>
```

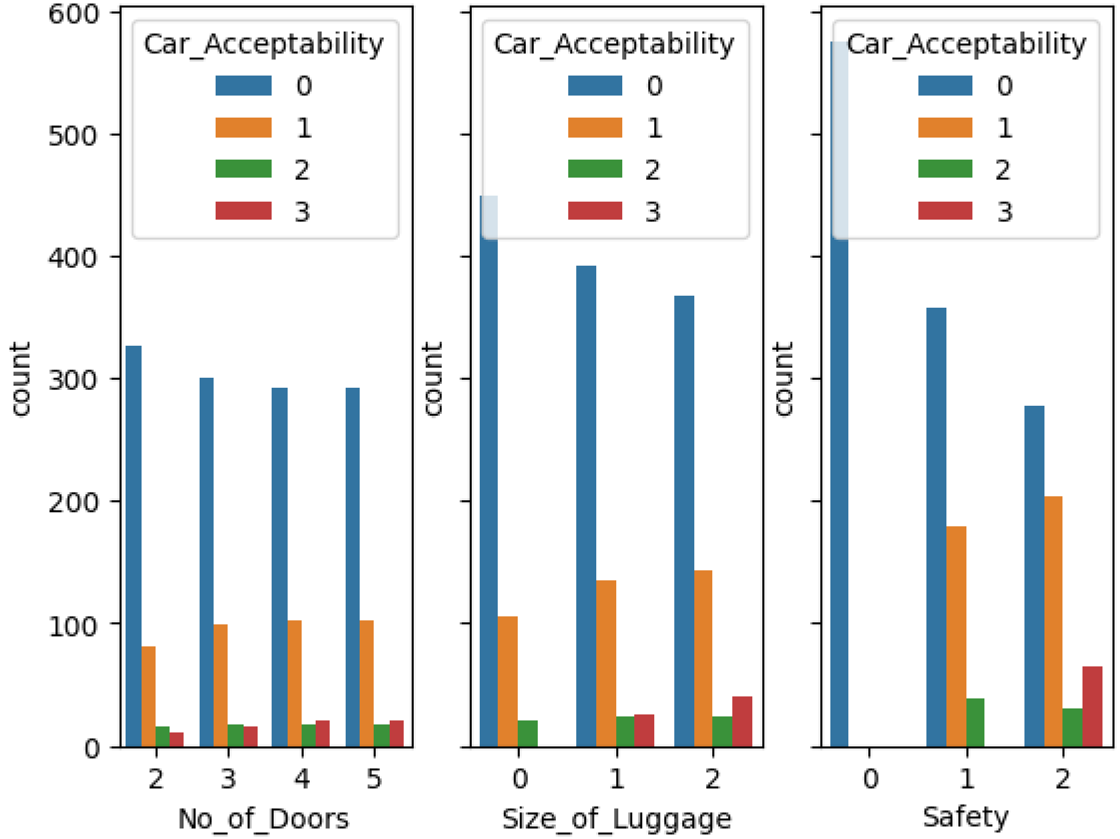
```
fig,ax=plt.subplots(1,3,sharey=True)
sns.countplot(x='Buying_Price',hue='Car_Acceptability',data=df,ax=ax[0])
sns.countplot(x='Maintenance_Price',hue='Car_Acceptability',data=df,ax=ax[1])
sns.countplot(x='Person_Capacity',hue='Car_Acceptability',data=df,ax=ax[2])
```

```
<Axes: xlabel='Person_Capacity', ylabel='count'>
```



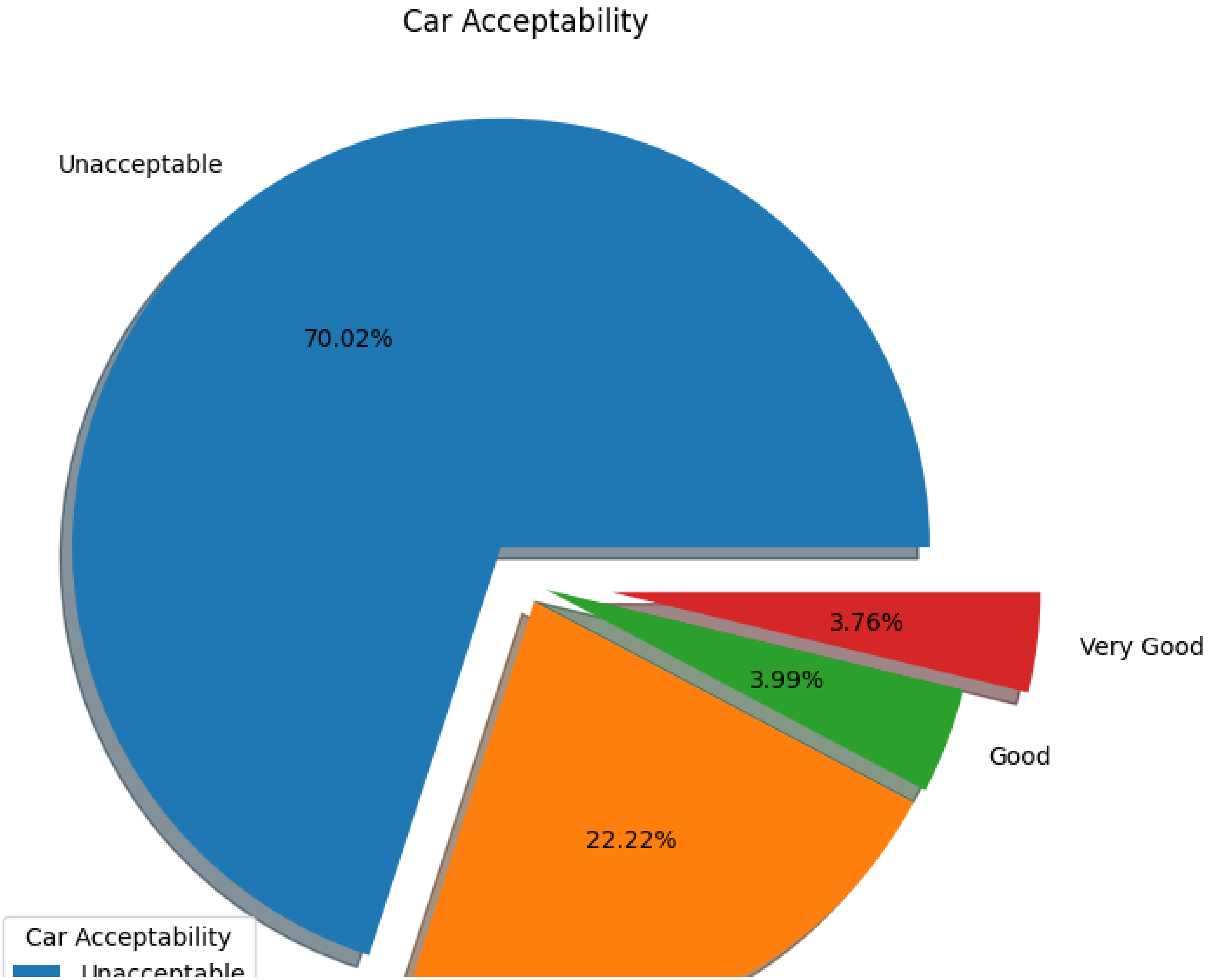
```
fig,ax=plt.subplots(1,3,sharey=True)
sns.countplot(x='No_of_Doors',hue='Car_Acceptability',data=df,ax=ax[0])
sns.countplot(x='Size_of_Luggage',hue='Car_Acceptability',data=df,ax=ax[1])
sns.countplot(x='Safety',hue='Car_Acceptability',data=df,ax=ax[2])
```

```
<Axes: xlabel='Safety', ylabel='count'>
```

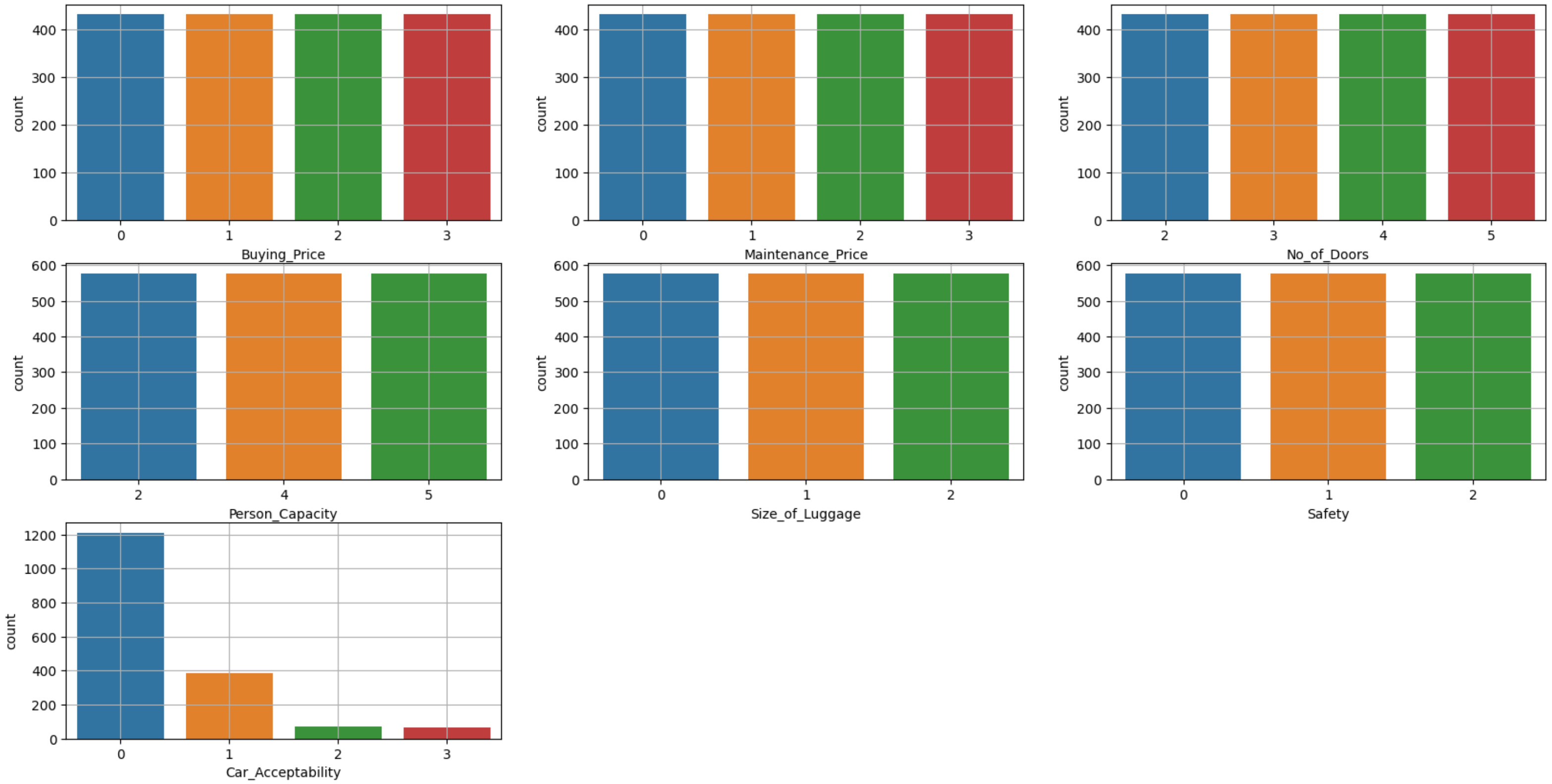


```
plt.figure(figsize=(11, 8))
plt.title('Car Acceptability')
plt.pie(df['Car_Acceptability'].value_counts(), explode = (0.1, 0.05, 0.05, 0.2), labels=['Unacceptable', 'Acceptable', 'Good', 'Very Good'], shadow=True, autopc
plt.legend(title='Car Acceptability', loc='lower left')
```

```
<matplotlib.legend.Legend at 0x7b8da96ea4a0>
```



```
plt.figure(figsize = (20, 10))

for i in range(7):
    plt.subplot(3 , 3, i+1)
    sns.countplot(data=df, x=df.iloc[:, i])
    plt.grid()
```


count
0
100
200
300
400
0
1
2
3
Buying_Price
Maintenance_Price
2
3
4
5
No_of_Doors
500
600
4
5
Person_Capacity
Size_of_Luggage
Safety
800
1000
1200
Car_Acceptability

```
df
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety	Car_Accep
0	3	3	2	2	0	0	
1	3	3	2	2	0	1	
2	3	3	2	2	0	2	
3	3	3	2	2	1	0	
4	3	3	2	2	1	1	
...	...	...	...	...	...	...	
1723	0	0	5	5	1	1	
1724	0	0	5	5	1	2	
1725	0	0	5	5	2	0	

```
x=df.iloc[:,:6]
x
```

	Buying_Price	Maintenance_Price	No_of_Doors	Person_Capacity	Size_of_Luggage	Safety
0	3	3	2	2	0	0

```
y=df['Car_Acceptability']
y
```

```
0       0
1       0
2       0
3       0
4       0
       ..
1723    2
1724    3
1725    0
1726    2
1727    3
Name: Car_Acceptability, Length: 1728, dtype: int64
```

```
1727         0         0         5         5         2         2
```

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.2,random_state=42)
```

```
x_train.shape
```

```
(1382, 6)
```

```
x_test.shape
```

```
(346, 6)
```

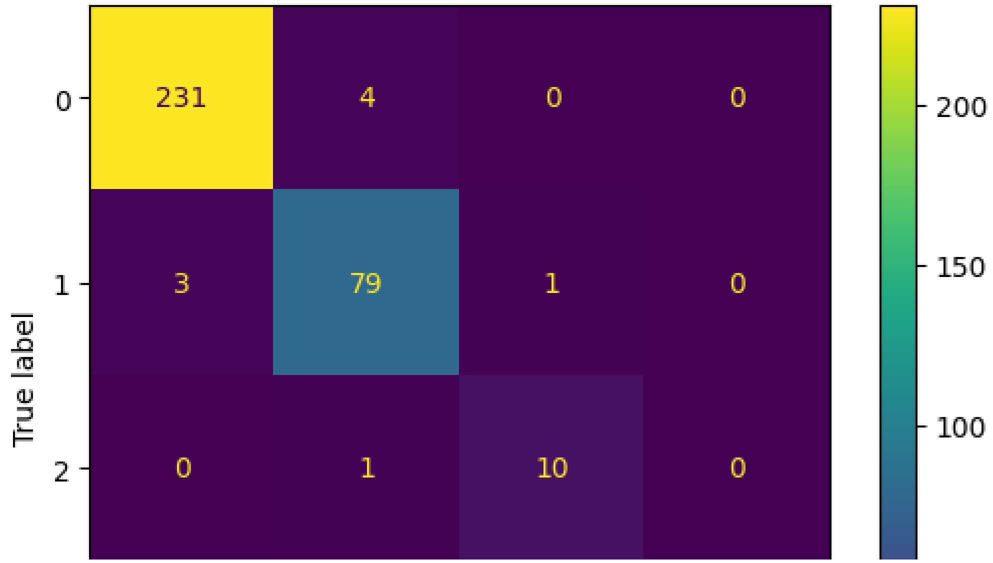
```
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.fit_transform(x_test)
```

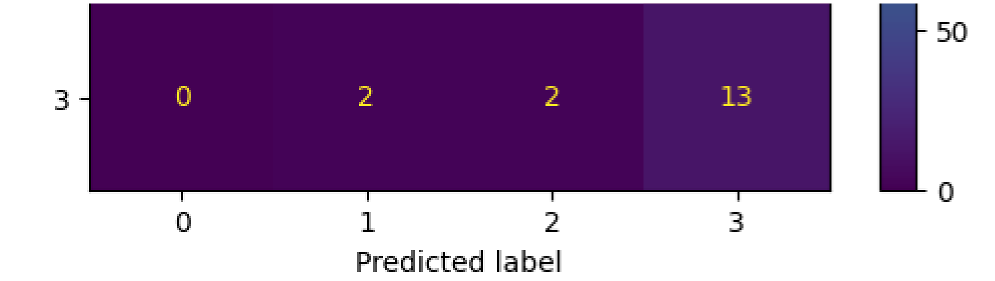
```
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.svm import SVC
from xgboost import XGBClassifier
from sklearn.ensemble import RandomForestClassifier, GradientBoostingClassifier, AdaBoostClassifier
from sklearn.naive_bayes import GaussianNB
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score, classification_report, ConfusionMatrixDisplay


classifiers = [

    ('KNeighborsClassifier', KNeighborsClassifier()),
    ('DecisionTreeClassifier', DecisionTreeClassifier()),
    ('RandomForestClassifier', RandomForestClassifier()),
    ('GradientBoostingClassifier', GradientBoostingClassifier()),
    ('AdaBoostClassifier', AdaBoostClassifier()),
    ('SVC', SVC()),
    ('GaussianNB', GaussianNB())
]


result=pd.DataFrame(columns=['Classifier', 'Accuracy'])


for clf_name,clf in classifiers:
    clf.fit(x_train, y_train)
    y_pred = clf.predict(x_test)
    accuracy=accuracy_score(y_test, y_pred)
    report=classification_report(y_test,y_pred)
    cmatrix=ConfusionMatrixDisplay.from_predictions(y_test,y_pred)
    print("Accuracy is:",accuracy)
    print("Classification report is:",report)
    print("Confusion matrix is:",cmatrix)
    result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
```

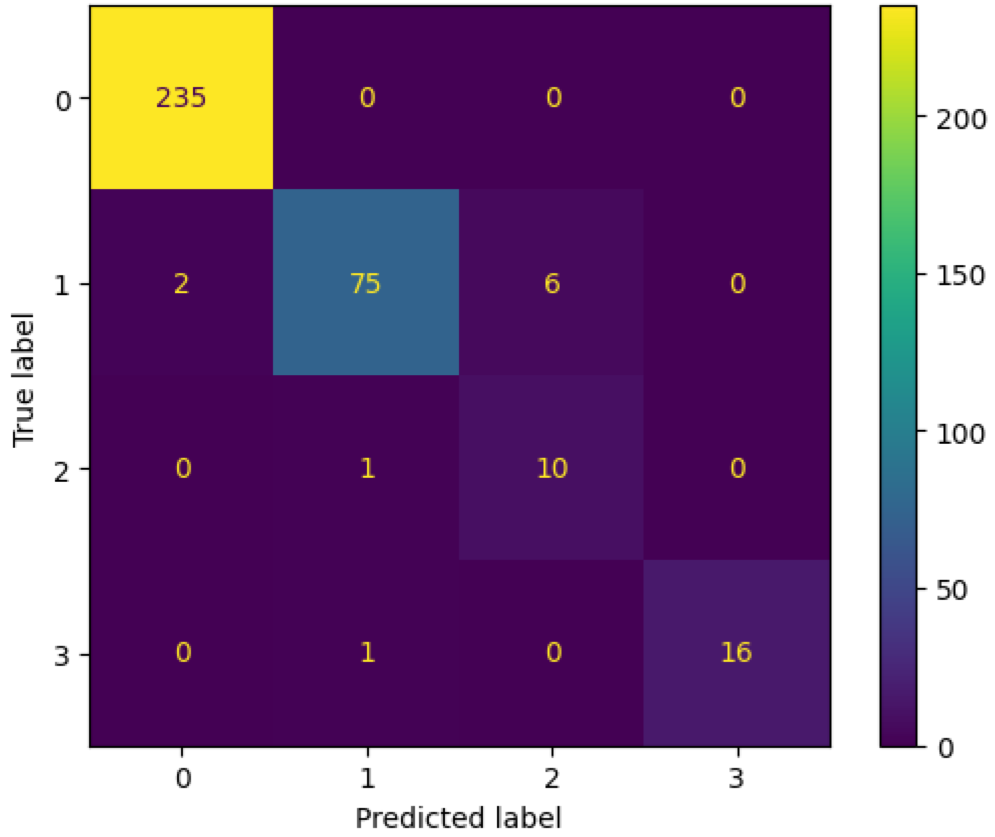
```
Accuracy is: 0.9624277456647399
Classification report is:                precision    recall  f1-score   support

           0       0.99      0.98      0.99       235
           1       0.92      0.95      0.93        83
           2       0.77      0.91      0.83        11
           3       1.00      0.76      0.87        17

    accuracy                           0.96       346
   macro avg       0.92      0.90      0.90       346
weighted avg       0.96      0.96      0.96       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8da6
Accuracy is: 0.9710982658959537
Classification report is:                precision    recall  f1-score   support

           0       0.99      1.00      1.00       235
           1       0.97      0.90      0.94        83
           2       0.62      0.91      0.74        11
           3       1.00      0.94      0.97        17

    accuracy                           0.97       346
   macro avg       0.90      0.94      0.91       346
weighted avg       0.98      0.97      0.97       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8da6
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be r
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9682080924855492
Classification report is:                precision    recall  f1-score   support

           0       1.00      1.00      1.00       235
           1       0.99      0.89      0.94        83
           2       0.59      0.91      0.71        11
           3       0.89      0.94      0.91        17

    accuracy                           0.97       346
   macro avg       0.86      0.94      0.89       346
weighted avg       0.98      0.97      0.97       346
```

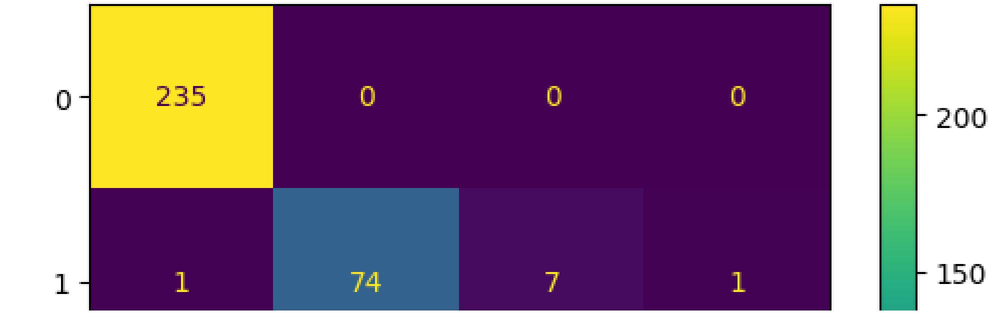
```
Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8da6
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be re
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9682080924855492
Classification report is:               precision    recall  f1-score   support

           0       1.00      1.00      1.00       235
           1       0.99      0.90      0.94        83
           2       0.55      1.00      0.71        11
           3       0.93      0.82      0.87        17

    accuracy                           0.97       346
   macro avg       0.87      0.93      0.88       346
weighted avg       0.98      0.97      0.97       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8d95
Accuracy is: 0.8670520231213873
Classification report is:               precision    recall  f1-score   support

           0       0.90      1.00      0.95       235
           1       0.90      0.52      0.66        83
           2       0.39      0.64      0.48        11
           3       0.79      0.88      0.83        17

    accuracy                           0.87       346
   macro avg       0.74      0.76      0.73       346
weighted avg       0.88      0.87      0.86       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8d95
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be re
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be re
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
Accuracy is: 0.9595375722543352
Classification report is:               precision    recall  f1-score   support

           0       0.97      1.00      0.98       235
           1       0.97      0.87      0.92        83
           2       0.73      1.00      0.85        11
           3       1.00      0.82      0.90        17

    accuracy                           0.96       346
   macro avg       0.92      0.92      0.91       346
```

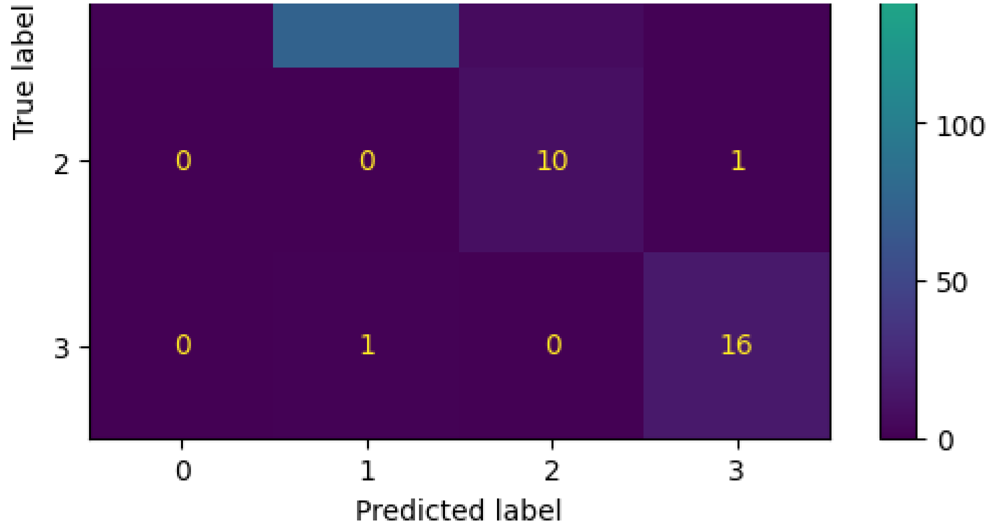
```
weighted avg       0.96      0.96      0.96       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8d95ε
Accuracy is: 0.7832369942196532
Classification report is:               precision    recall  f1-score   support

           0       0.88      0.93      0.90       235
           1       0.55      0.58      0.56        83
           2       0.42      0.45      0.43        11
           3       0.00      0.00      0.00        17

    accuracy                           0.78       346
   macro avg       0.46      0.49      0.48       346
weighted avg       0.75      0.78      0.76       346

Confusion matrix is: <sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay object at 0x7b8d95ε
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be re
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning
  _warn_prf(average, modifier, msg_start, len(result))
/usr/local/lib/python3.10/dist-packages/sklearn/metrics/_classification.py:1344: UndefinedMetricWarning
  _warn_prf(average, modifier, msg_start, len(result))
<ipython-input-38-db3d13d0e22f>:10: FutureWarning: The frame.append method is deprecated and will be re
  result=result.append({'Classifier': clf_name, 'Accuracy': accuracy}, ignore_index=True)
```

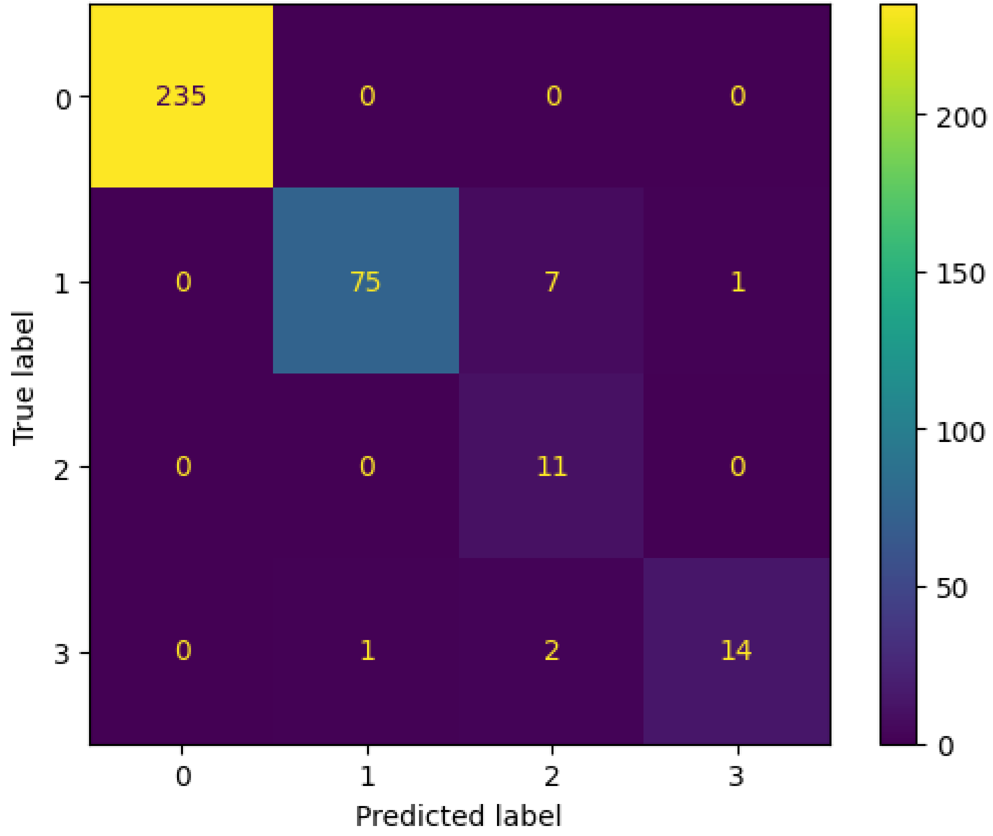


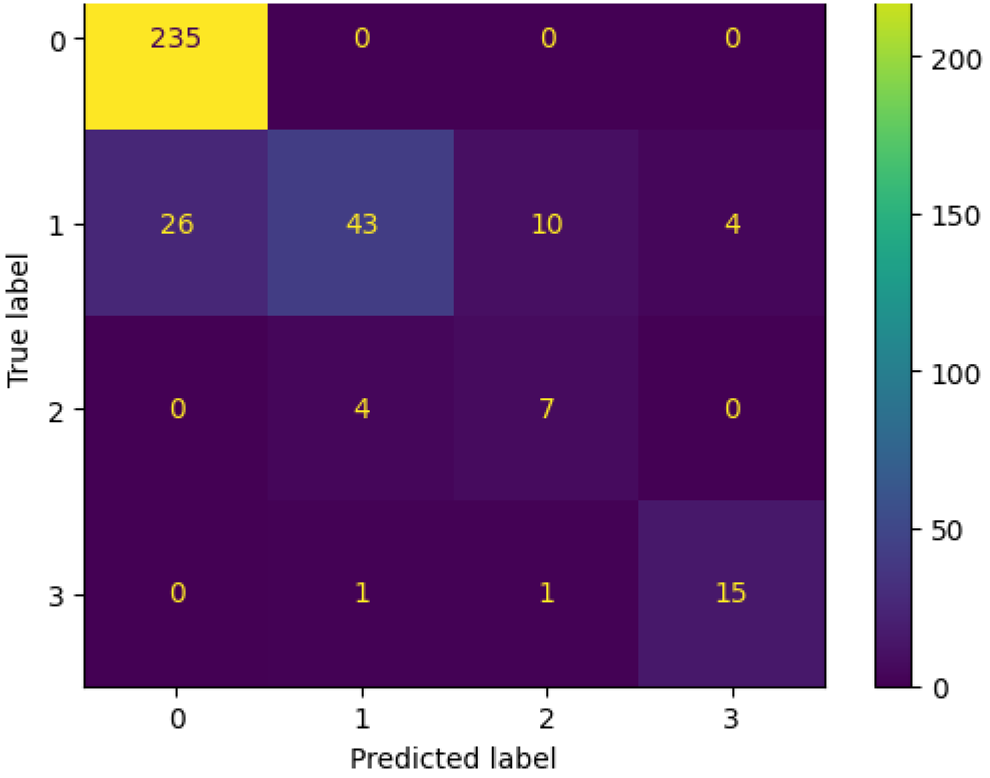

3
0
2
2
13
0
1
2
3
Predicted label
50
0

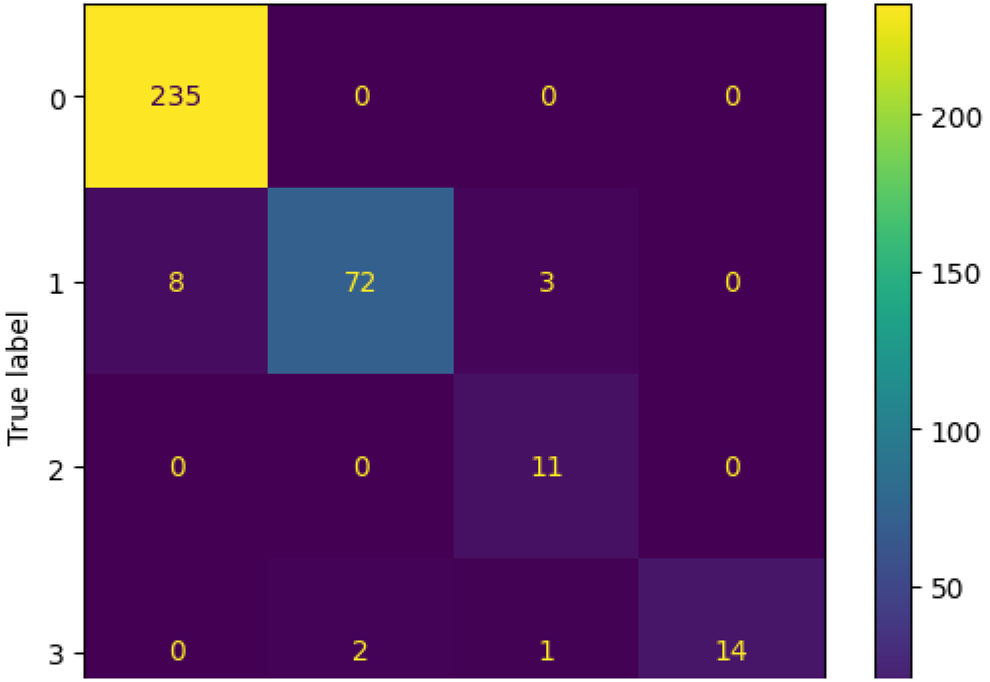

True label
0
1
2
3
235
0
0
0
2
75
6
0
0
1
10
0
0
1
0
16
0
1
2
3
Predicted label
200
150
100
50
0

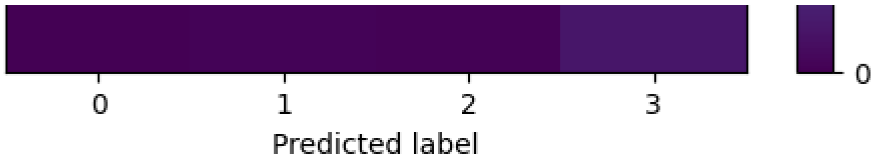

0
1
235
0
0
0
1
74
7
1
200
150

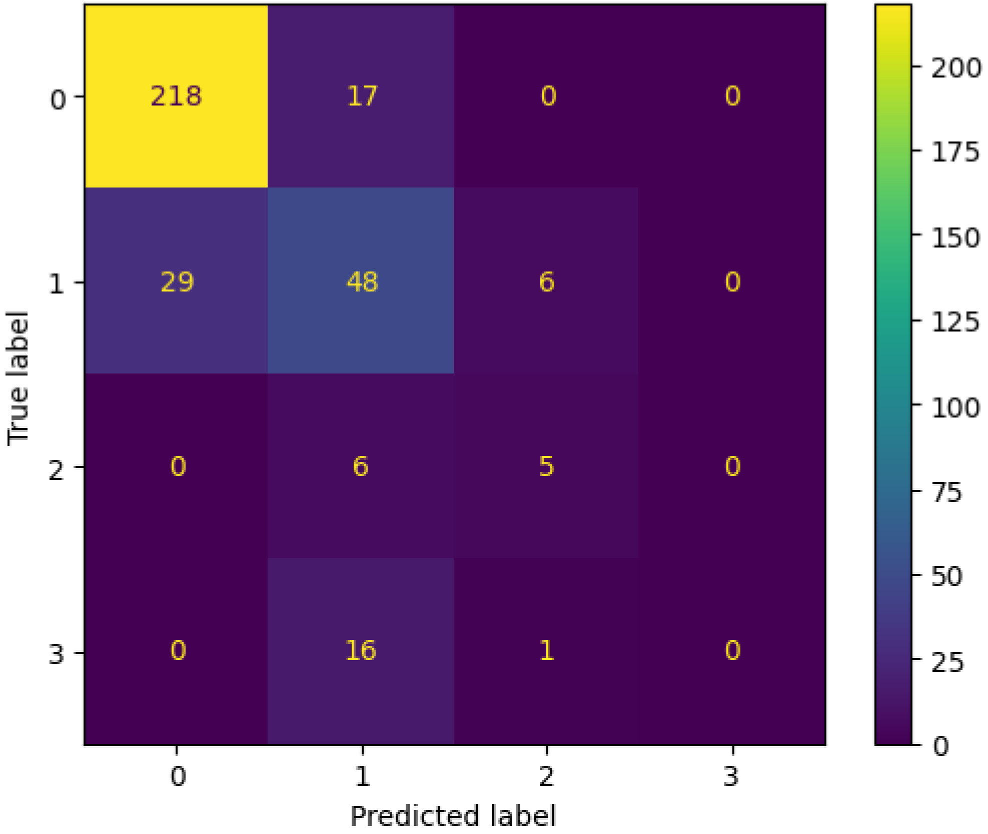

True label
2
0
0
10
1
3
0
1
0
16
0
1
2
3
Predicted label
100
50
0


0
235
0
0
0
1
0
75
7
1
2
0
0
11
0
3
0
1
2
14
True label
0
1
2
3
Predicted label
200
150
100
50
0


True label
Predicted label
0
1
2
3
235 0 0 0
26 43 10 4
0 4 7 0
0 1 1 15
0
1
2
3
0
50
100
150
200


True label
0
1
2
3
235 0 0 0
8 72 3 0
0 0 11 0
0 2 1 14
50
100
150
200





```
print(result)
```

```
                     Classifier  Accuracy
0          KNeighborsClassifier  0.962428
1        DecisionTreeClassifier  0.971098
2        RandomForestClassifier  0.968208
3    GradientBoostingClassifier  0.968208
4            AdaBoostClassifier  0.867052
5                           SVC  0.959538
6                    GaussianNB  0.783237
```

Hence, we obtained high accuracy while using Decision Tree Classifier: 0.968208

```
model_dt=DecisionTreeClassifier().fit(x_train,y_train)
y_pred_dt=model_dt.predict(x_test)
```

```
y_pred_dt
```

```
array([0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 3, 1, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1,
       1, 0, 1, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 1, 2, 0, 0, 0, 0, 0, 1, 1,
       2, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 2, 0, 1, 0, 0, 1, 3,
       0, 1, 1, 0, 0, 0, 0, 0, 2, 3, 0, 0, 0, 0, 3, 0, 0, 1, 3, 1, 0, 1,
       3, 1, 0, 2, 0, 0, 0, 2, 0, 0, 0, 1, 0, 0, 0, 0, 2, 2, 1, 0, 0, 1,
       0, 0, 0, 0, 0, 1, 2, 0, 0, 0, 0, 1, 0, 1, 1, 0, 2, 0, 0, 0, 0, 0,
       1, 0, 0, 1, 0, 3, 0, 0, 0, 1, 0, 1, 0, 1, 0, 0, 0, 0, 1, 3, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 1, 0, 1, 3, 1, 0, 1, 0, 0, 1,
       0, 0, 0, 0, 1, 0, 0, 0, 2, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0, 0, 0, 1,
       0, 0, 0, 0, 0, 2, 0, 0, 0, 2, 0, 0, 0, 0, 1, 0, 0, 1, 0, 0, 0, 2,
       3, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 1, 0, 0, 0,
       0, 0, 1, 3, 0, 0, 0, 1, 0, 0, 3, 0, 0, 1, 2, 0, 1, 0, 1, 1, 0, 0,
       1, 2, 3, 1, 1, 0, 0, 0, 0, 3, 1, 1, 0, 0, 0, 0, 3, 0, 0, 0, 0, 1,
       0, 0, 3, 1, 1, 0, 0, 1, 1, 1, 0, 1, 1, 0, 0, 0, 0, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0])
```

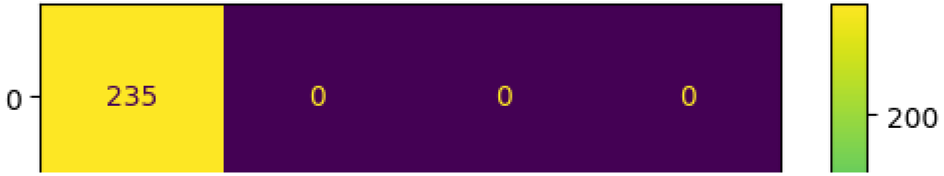
```
test_check=pd.DataFrame()
test_check['Actual']=y_test
test_check['Predicted']=y_pred_dt
test_check.sort_index()
```

	Actual	Predicted
15	0	0
23	0	0
29	0	0
30	0	0
32	0	0
...	...	...
1694	2	2

High accuracy is while using Decision Tree Classifier ie, Accuracy=0.968208

```
cmatrix_dt=ConfusionMatrixDisplay.from_predictions(y_test,y_pred_dt)
cmatrix_dt
```

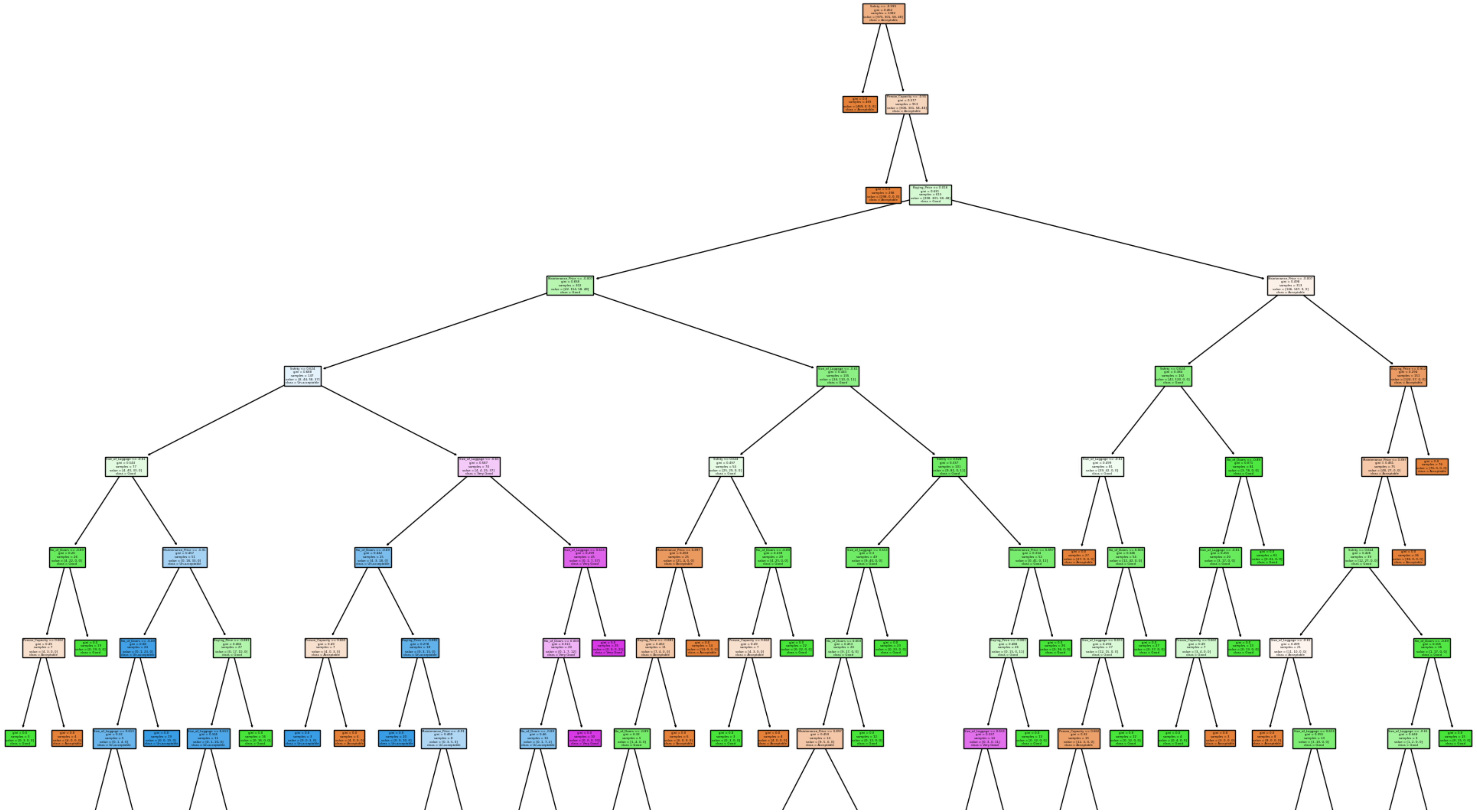
```
<sklearn.metrics._plot.confusion_matrix.ConfusionMatrixDisplay at 0x7b8da95dfb20>
```

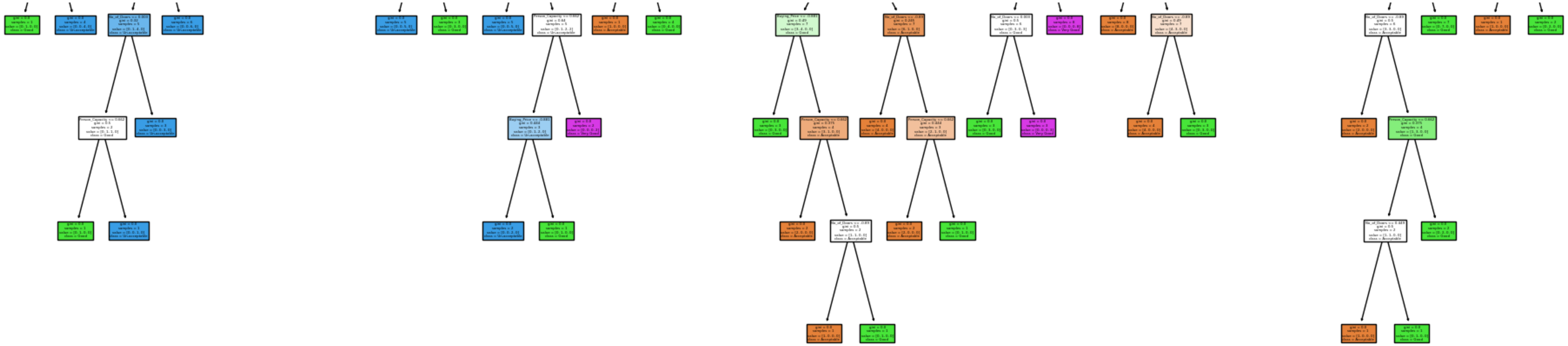


```
feature_names=df.columns[0: 6]
target_names=df['Car_Acceptability'].unique().tolist()
```

```
from sklearn.tree import plot_tree # tree diagram

plt.figure(figsize=(25, 20))
plot_tree(model_dt, feature_names = feature_names, class_names = ['Acceptable', 'Good', 'Un-acceptable', 'Very Good'], filled = True, rounded = False)
```

```
[Text(0.5982142857142857, 0.9615384615384616, 'Safety <= -0.599\ngini = 0.452\nsamples = 1382\nvalue = [975, 301, 58, 48]\nclass = Acceptable'),
 Text(0.5823412698412699, 0.8846153846153846, 'gini = 0.0\nsamples = 469\nvalue = [469, 0, 0, 0]\nclass = Acceptable'),
 Text(0.6140873015873016, 0.8846153846153846, 'Person_Capacity <= -0.55\ngini = 0.577\nsamples = 913\nvalue = [506, 301, 58, 48]\nclass = Acceptable'),
 Text(0.5982142857142857, 0.8076923076923077, 'gini = 0.0\nsamples = 298\nvalue = [298, 0, 0, 0]\nclass = Acceptable'),
 Text(0.6299603174603174, 0.8076923076923077, 'Buying_Price <= 0.016\ngini = 0.631\nsamples = 615\nvalue = [208, 301, 58, 48]\nclass = Good'),
 Text(0.3869047619047619, 0.7307692307692307, 'Maintenance_Price <= -0.007\ngini = 0.658\nsamples = 302\nvalue = [42, 154, 58, 48]\nclass = Good'),
 Text(0.20634920634920634, 0.6538461538461539, 'Safety <= 0.624\ngini = 0.688\nsamples = 147\nvalue = [8, 44, 58, 37]\nclass = Un-acceptable'),
 Text(0.0873015873015873, 0.5769230769230769, 'Size_of_Luggage <= -0.61\ngini = 0.544\nsamples = 77\nvalue = [4, 40, 33, 0]\nclass = Good'),
 Text(0.047619047619047616, 0.5, 'No_of_Doors <= -0.89\ngini = 0.26\nsamples = 26\nvalue = [4, 22, 0, 0]\nclass = Good'),
 Text(0.031746031746031744, 0.4230769230769231, 'Person_Capacity <= 0.662\ngini = 0.49\nsamples = 7\nvalue = [4, 3, 0, 0]\nclass = Acceptable'),
 Text(0.015873015873015872, 0.34615384615384615, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.047619047619047616, 0.34615384615384615, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0]\nclass = Acceptable'),
 Text(0.06349206349206349, 0.4230769230769231, 'gini = 0.0\nsamples = 19\nvalue = [0, 19, 0, 0]\nclass = Good'),
 Text(0.12698412698412698, 0.5, 'Maintenance_Price <= -0.91\ngini = 0.457\nsamples = 51\nvalue = [0, 18, 33, 0]\nclass = Un-acceptable'),
 Text(0.09523809523809523, 0.4230769230769231, 'No_of_Doors <= -0.89\ngini = 0.08\nsamples = 24\nvalue = [0, 1, 23, 0]\nclass = Un-acceptable'),
 Text(0.07936507936507936, 0.34615384615384615, 'Size_of_Luggage <= 0.613\ngini = 0.32\nsamples = 5\nvalue = [0, 1, 4, 0]\nclass = Un-acceptable'),
 Text(0.06349206349206349, 0.2692307692307692, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.09523809523809523, 0.2692307692307692, 'gini = 0.0\nsamples = 4\nvalue = [0, 0, 4, 0]\nclass = Un-acceptable'),
 Text(0.1111111111111111, 0.34615384615384615, 'gini = 0.0\nsamples = 19\nvalue = [0, 0, 19, 0]\nclass = Un-acceptable'),
 Text(0.15873015873015872, 0.4230769230769231, 'Buying_Price <= -0.881\ngini = 0.466\nsamples = 27\nvalue = [0, 17, 10, 0]\nclass = Good'),
 Text(0.14285714285714285, 0.34615384615384615, 'Size_of_Luggage <= 0.613\ngini = 0.165\nsamples = 11\nvalue = [0, 1, 10, 0]\nclass = Un-acceptable'),
 Text(0.12698412698412698, 0.2692307692307692, 'No_of_Doors <= 0.003\ngini = 0.32\nsamples = 5\nvalue = [0, 1, 4, 0]\nclass = Un-acceptable'),
 Text(0.1111111111111111, 0.19230769230769232, 'Person_Capacity <= 0.662\ngini = 0.5\nsamples = 2\nvalue = [0, 1, 1, 0]\nclass = Good'),
 Text(0.09523809523809523, 0.11538461538461539, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.12698412698412698, 0.11538461538461539, 'gini = 0.0\nsamples = 1\nvalue = [0, 0, 1, 0]\nclass = Un-acceptable'),
 Text(0.14285714285714285, 0.19230769230769232, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0]\nclass = Un-acceptable'),
 Text(0.15873015873015872, 0.2692307692307692, 'gini = 0.0\nsamples = 6\nvalue = [0, 0, 6, 0]\nclass = Un-acceptable'),
 Text(0.1746031746031746, 0.34615384615384615, 'gini = 0.0\nsamples = 16\nvalue = [0, 16, 0, 0]\nclass = Good'),
 Text(0.3253968253968254, 0.5769230769230769, 'Size_of_Luggage <= -0.61\ngini = 0.587\nsamples = 70\nvalue = [4, 4, 25, 37]\nclass = Very Good'),
 Text(0.25396825396825395, 0.5, 'No_of_Doors <= -0.89\ngini = 0.442\nsamples = 25\nvalue = [4, 3, 18, 0]\nclass = Un-acceptable'),
 Text(0.2222222222222222, 0.4230769230769231, 'Person_Capacity <= 0.662\ngini = 0.49\nsamples = 7\nvalue = [4, 0, 3, 0]\nclass = Acceptable'),
 Text(0.20634920634920634, 0.34615384615384615, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 3, 0]\nclass = Un-acceptable'),
 Text(0.23809523809523808, 0.34615384615384615, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0]\nclass = Acceptable'),
 Text(0.2857142857142857, 0.4230769230769231, 'Buying_Price <= -0.881\ngini = 0.278\nsamples = 18\nvalue = [0, 3, 15, 0]\nclass = Un-acceptable'),
 Text(0.2698412698412698, 0.34615384615384615, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 10, 0]\nclass = Un-acceptable'),
 Text(0.30158730158730157, 0.34615384615384615, 'Maintenance_Price <= -0.91\ngini = 0.469\nsamples = 8\nvalue = [0, 3, 5, 0]\nclass = Un-acceptable'),
 Text(0.2857142857142857, 0.2692307692307692, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 5, 0]\nclass = Un-acceptable'),
 Text(0.31746031746031744, 0.2692307692307692, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.3968253968253968, 0.5, 'Size_of_Luggage <= 0.613\ngini = 0.299\nsamples = 45\nvalue = [0, 1, 7, 37]\nclass = Very Good'),
 Text(0.38095238095238093, 0.4230769230769231, 'No_of_Doors <= 0.003\ngini = 0.515\nsamples = 20\nvalue = [0, 1, 7, 12]\nclass = Very Good'),
 Text(0.36507936507936506, 0.34615384615384615, 'No_of_Doors <= -0.89\ngini = 0.46\nsamples = 10\nvalue = [0, 1, 7, 2]\nclass = Un-acceptable'),
 Text(0.3492063492063492, 0.2692307692307692, 'gini = 0.0\nsamples = 5\nvalue = [0, 0, 5, 0]\nclass = Un-acceptable'),
 Text(0.38095238095238093, 0.2692307692307692, 'Person_Capacity <= 0.662\ngini = 0.64\nsamples = 5\nvalue = [0, 1, 2, 2]\nclass = Un-acceptable'),
 Text(0.36507936507936506, 0.19230769230769232, 'Buying_Price <= -0.881\ngini = 0.444\nsamples = 3\nvalue = [0, 1, 2, 0]\nclass = Un-acceptable'),
 Text(0.3492063492063492, 0.11538461538461539, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 2, 0]\nclass = Un-acceptable'),
 Text(0.38095238095238093, 0.11538461538461539, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.3968253968253968, 0.19230769230769232, 'gini = 0.0\nsamples = 2\nvalue = [0, 0, 0, 2]\nclass = Very Good'),
 Text(0.3968253968253968, 0.34615384615384615, 'gini = 0.0\nsamples = 10\nvalue = [0, 0, 0, 10]\nclass = Very Good'),
 Text(0.4126984126984127, 0.4230769230769231, 'gini = 0.0\nsamples = 25\nvalue = [0, 0, 0, 25]\nclass = Very Good'),
 Text(0.5674603174603174, 0.6538461538461539, 'Size_of_Luggage <= -0.61\ngini = 0.443\nsamples = 155\nvalue = [34, 110, 0, 11]\nclass = Good'),
 Text(0.49206349206349204, 0.5769230769230769, 'Safety <= 0.624\ngini = 0.497\nsamples = 54\nvalue = [25, 29, 0, 0]\nclass = Good'),
 Text(0.4603174603174603, 0.5, 'Maintenance_Price <= 0.897\ngini = 0.269\nsamples = 25\nvalue = [21, 4, 0, 0]\nclass = Acceptable'),
 Text(0.4444444444444444, 0.4230769230769231, 'Buying_Price <= -0.881\ngini = 0.463\nsamples = 11\nvalue = [7, 4, 0, 0]\nclass = Acceptable'),
 Text(0.42857142857142855, 0.34615384615384615, 'No_of_Doors <= -0.89\ngini = 0.32\nsamples = 5\nvalue = [1, 4, 0, 0]\nclass = Good'),
 Text(0.4126984126984127, 0.2692307692307692, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0]\nclass = Acceptable'),
 Text(0.4444444444444444, 0.2692307692307692, 'gini = 0.0\nsamples = 4\nvalue = [0, 4, 0, 0]\nclass = Good'),
 Text(0.4603174603174603, 0.34615384615384615, 'gini = 0.0\nsamples = 6\nvalue = [6, 0, 0, 0]\nclass = Acceptable'),
 Text(0.47619047619047616, 0.4230769230769231, 'gini = 0.0\nsamples = 14\nvalue = [14, 0, 0, 0]\nclass = Acceptable'),
 Text(0.5238095238095238, 0.5, 'No_of_Doors <= -0.89\ngini = 0.238\nsamples = 29\nvalue = [4, 25, 0, 0]\nclass = Good'),
 Text(0.5079365079365079, 0.4230769230769231, 'Person_Capacity <= 0.662\ngini = 0.49\nsamples = 7\nvalue = [4, 3, 0, 0]\nclass = Acceptable'),
 Text(0.49206349206349204, 0.34615384615384615, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.5238095238095238, 0.34615384615384615, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0]\nclass = Acceptable'),
 Text(0.5396825396825397, 0.4230769230769231, 'gini = 0.0\nsamples = 22\nvalue = [0, 22, 0, 0]\nclass = Good'),
 Text(0.6428571428571429, 0.5769230769230769, 'Safety <= 0.624\ngini = 0.337\nsamples = 101\nvalue = [9, 81, 0, 11]\nclass = Good'),
 Text(0.5873015873015873, 0.5, 'Size_of_Luggage <= 0.613\ngini = 0.3\nsamples = 49\nvalue = [9, 40, 0, 0]\nclass = Good'),
 Text(0.5714285714285714, 0.4230769230769231, 'No_of_Doors <= 0.003\ngini = 0.453\nsamples = 26\nvalue = [9, 17, 0, 0]\nclass = Good'),
 Text(0.5555555555555556, 0.34615384615384615, 'Maintenance_Price <= 0.897\ngini = 0.459\nsamples = 14\nvalue = [9, 5, 0, 0]\nclass = Acceptable'),
 Text(0.5238095238095238, 0.2692307692307692, 'Buying_Price <= -0.881\ngini = 0.49\nsamples = 7\nvalue = [3, 4, 0, 0]\nclass = Good'),
 Text(0.5079365079365079, 0.19230769230769232, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.5396825396825397, 0.19230769230769232, 'Person_Capacity <= 0.662\ngini = 0.375\nsamples = 4\nvalue = [3, 1, 0, 0]\nclass = Acceptable'),
 Text(0.5238095238095238, 0.11538461538461539, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0, 0]\nclass = Acceptable'),
 Text(0.5555555555555556, 0.11538461538461539, 'No_of_Doors <= -0.89\ngini = 0.5\nsamples = 2\nvalue = [1, 1, 0, 0]\nclass = Acceptable'),
 Text(0.5396825396825397, 0.038461538461538464, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0]\nclass = Acceptable'),
 Text(0.5714285714285714, 0.038461538461538464, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.5873015873015873, 0.2692307692307692, 'No_of_Doors <= -0.89\ngini = 0.245\nsamples = 7\nvalue = [6, 1, 0, 0]\nclass = Acceptable'),
 Text(0.5714285714285714, 0.19230769230769232, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0]\nclass = Acceptable'),
 Text(0.6031746031746031, 0.19230769230769232, 'Person_Capacity <= 0.662\ngini = 0.444\nsamples = 3\nvalue = [2, 1, 0, 0]\nclass = Acceptable'),
 Text(0.5873015873015873, 0.11538461538461539, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0, 0]\nclass = Acceptable'),
 Text(0.6190476190476191, 0.11538461538461539, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.5873015873015873, 0.34615384615384615, 'gini = 0.0\nsamples = 12\nvalue = [0, 12, 0, 0]\nclass = Good'),
 Text(0.6031746031746031, 0.4230769230769231, 'gini = 0.0\nsamples = 23\nvalue = [0, 23, 0, 0]\nclass = Good'),
 Text(0.6984126984126984, 0.5, 'Maintenance_Price <= 0.897\ngini = 0.334\nsamples = 52\nvalue = [0, 41, 0, 11]\nclass = Good'),
 Text(0.6825396825396826, 0.4230769230769231, 'Buying_Price <= -0.881\ngini = 0.488\nsamples = 26\nvalue = [0, 15, 0, 11]\nclass = Good'),
 Text(0.6666666666666666, 0.34615384615384615, 'Size_of_Luggage <= 0.613\ngini = 0.337\nsamples = 14\nvalue = [0, 3, 0, 11]\nclass = Very Good'),
 Text(0.6507936507936508, 0.2692307692307692, 'No_of_Doors <= 0.003\ngini = 0.5\nsamples = 6\nvalue = [0, 3, 0, 3]\nclass = Good'),
 Text(0.6349206349206349, 0.19230769230769232, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.6666666666666666, 0.19230769230769232, 'gini = 0.0\nsamples = 3\nvalue = [0, 0, 0, 3]\nclass = Very Good'),
 Text(0.6825396825396826, 0.2692307692307692, 'gini = 0.0\nsamples = 8\nvalue = [0, 0, 0, 8]\nclass = Very Good'),
 Text(0.6984126984126984, 0.34615384615384615, 'gini = 0.0\nsamples = 12\nvalue = [0, 12, 0, 0]\nclass = Good'),
 Text(0.7142857142857143, 0.4230769230769231, 'gini = 0.0\nsamples = 26\nvalue = [0, 26, 0, 0]\nclass = Good'),
 Text(0.873015873015873, 0.7307692307692307, 'Maintenance_Price <= -0.007\ngini = 0.498\nsamples = 313\nvalue = [166, 147, 0, 0]\nclass = Acceptable'),
 Text(0.7936507936507936, 0.6538461538461539, 'Safety <= 0.624\ngini = 0.384\nsamples = 162\nvalue = [42, 120, 0, 0]\nclass = Good'),
 Text(0.746031746031746, 0.5769230769230769, 'Size_of_Luggage <= -0.61\ngini = 0.499\nsamples = 81\nvalue = [39, 42, 0, 0]\nclass = Good'),
 Text(0.7301587301587301, 0.5, 'gini = 0.0\nsamples = 27\nvalue = [27, 0, 0, 0]\nclass = Acceptable'),
 Text(0.7619047619047619, 0.5, 'No_of_Doors <= 0.003\ngini = 0.346\nsamples = 54\nvalue = [12, 42, 0, 0]\nclass = Good'),
 Text(0.746031746031746, 0.4230769230769231, 'Size_of_Luggage <= 0.613\ngini = 0.494\nsamples = 27\nvalue = [12, 15, 0, 0]\nclass = Good'),
 Text(0.7301587301587301, 0.34615384615384615, 'Person_Capacity <= 0.662\ngini = 0.32\nsamples = 15\nvalue = [12, 3, 0, 0]\nclass = Acceptable'),
 Text(0.7142857142857143, 0.2692307692307692, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0]\nclass = Acceptable'),
 Text(0.746031746031746, 0.2692307692307692, 'No_of_Doors <= -0.89\ngini = 0.49\nsamples = 7\nvalue = [4, 3, 0, 0]\nclass = Acceptable'),
 Text(0.7301587301587301, 0.19230769230769232, 'gini = 0.0\nsamples = 4\nvalue = [4, 0, 0, 0]\nclass = Acceptable'),
 Text(0.7619047619047619, 0.19230769230769232, 'gini = 0.0\nsamples = 3\nvalue = [0, 3, 0, 0]\nclass = Good'),
 Text(0.7619047619047619, 0.34615384615384615, 'gini = 0.0\nsamples = 12\nvalue = [0, 12, 0, 0]\nclass = Good'),
 Text(0.7777777777777778, 0.4230769230769231, 'gini = 0.0\nsamples = 27\nvalue = [0, 27, 0, 0]\nclass = Good'),
 Text(0.8412698412698413, 0.5769230769230769, 'No_of_Doors <= -0.89\ngini = 0.071\nsamples = 81\nvalue = [3, 78, 0, 0]\nclass = Good'),
 Text(0.8253968253968254, 0.5, 'Size_of_Luggage <= -0.61\ngini = 0.255\nsamples = 20\nvalue = [3, 17, 0, 0]\nclass = Good'),
 Text(0.8095238095238095, 0.4230769230769231, 'Person_Capacity <= 0.662\ngini = 0.49\nsamples = 7\nvalue = [3, 4, 0, 0]\nclass = Good'),
 Text(0.7936507936507936, 0.34615384615384615, 'gini = 0.0\nsamples = 4\nvalue = [0, 4, 0, 0]\nclass = Good'),
 Text(0.8253968253968254, 0.34615384615384615, 'gini = 0.0\nsamples = 3\nvalue = [3, 0, 0, 0]\nclass = Acceptable'),
 Text(0.8412698412698413, 0.4230769230769231, 'gini = 0.0\nsamples = 13\nvalue = [0, 13, 0, 0]\nclass = Good'),
 Text(0.8571428571428571, 0.5, 'gini = 0.0\nsamples = 61\nvalue = [0, 61, 0, 0]\nclass = Good'),
 Text(0.9523809523809523, 0.6538461538461539, 'Buying_Price <= 0.912\ngini = 0.294\nsamples = 151\nvalue = [124, 27, 0, 0]\nclass = Acceptable'),
 Text(0.9365079365079365, 0.5769230769230769, 'Maintenance_Price <= 0.897\ngini = 0.461\nsamples = 75\nvalue = [48, 27, 0, 0]\nclass = Acceptable'),
 Text(0.9206349206349206, 0.5, 'Safety <= 0.624\ngini = 0.426\nsamples = 39\nvalue = [12, 27, 0, 0]\nclass = Good'),
 Text(0.873015873015873, 0.4230769230769231, 'Size_of_Luggage <= -0.61\ngini = 0.499\nsamples = 21\nvalue = [11, 10, 0, 0]\nclass = Acceptable'),
 Text(0.8571428571428571, 0.34615384615384615, 'gini = 0.0\nsamples = 8\nvalue = [8, 0, 0, 0]\nclass = Acceptable'),
 Text(0.8888888888888888, 0.34615384615384615, 'Size_of_Luggage <= 0.613\ngini = 0.355\nsamples = 13\nvalue = [3, 10, 0, 0]\nclass = Good'),
 Text(0.873015873015873, 0.2692307692307692, 'No_of_Doors <= -0.89\ngini = 0.5\nsamples = 6\nvalue = [3, 3, 0, 0]\nclass = Acceptable'),
 Text(0.8571428571428571, 0.19230769230769232, 'gini = 0.0\nsamples = 2\nvalue = [2, 0, 0, 0]\nclass = Acceptable'),
 Text(0.8888888888888888, 0.19230769230769232, 'Person_Capacity <= 0.662\ngini = 0.375\nsamples = 4\nvalue = [1, 3, 0, 0]\nclass = Good'),
 Text(0.873015873015873, 0.11538461538461539, 'No_of_Doors <= 0.449\ngini = 0.5\nsamples = 2\nvalue = [1, 1, 0, 0]\nclass = Acceptable'),
 Text(0.8571428571428571, 0.038461538461538464, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0]\nclass = Acceptable'),
 Text(0.8888888888888888, 0.038461538461538464, 'gini = 0.0\nsamples = 1\nvalue = [0, 1, 0, 0]\nclass = Good'),
 Text(0.9047619047619048, 0.11538461538461539, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0]\nclass = Good'),
 Text(0.9047619047619048, 0.2692307692307692, 'gini = 0.0\nsamples = 7\nvalue = [0, 7, 0, 0]\nclass = Good'),
 Text(0.9682539682539683, 0.4230769230769231, 'No_of_Doors <= -0.89\ngini = 0.105\nsamples = 18\nvalue = [1, 17, 0, 0]\nclass = Good'),
 Text(0.9523809523809523, 0.34615384615384615, 'Size_of_Luggage <= -0.61\ngini = 0.444\nsamples = 3\nvalue = [1, 2, 0, 0]\nclass = Good'),
 Text(0.9365079365079365, 0.2692307692307692, 'gini = 0.0\nsamples = 1\nvalue = [1, 0, 0, 0]\nclass = Acceptable'),
 Text(0.9682539682539683, 0.2692307692307692, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0]\nclass = Good'),
 Text(0.9682539682539683, 0.2692307692307692, 'gini = 0.0\nsamples = 2\nvalue = [0, 2, 0, 0]\nclass = Good'),
 Text(0.9841269841269841, 0.34615384615384615, 'gini = 0.0\nsamples = 15\nvalue = [0, 15, 0, 0]\nclass = Good'),
 Text(0.9523809523809523, 0.5, 'gini = 0.0\nsamples = 36\nvalue = [36, 0, 0, 0]\nclass = Acceptable'),
 Text(0.9682539682539683, 0.5769230769230769, 'gini = 0.0\nsamples = 76\nvalue = [76, 0, 0, 0]\nclass = Acceptable')]
```





✓ 6s completed at 4:40 PM