In [1]:

```
import numpy as np
from sklearn.ensemble import RandomForestClassifier
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.metrics import classification_report,confusion_matrix,accuracy_score
```

In [2]:

```
df=pd.read_csv("car_evaluation.csv")
df
```

Out[2]:

	buying	maint	doors	persons	lug_boot	safety	outcome
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	5	5	med	med	good
1724	low	low	5	5	med	high	vgood
1725	low	low	5	5	big	low	unacc
1726	low	low	5	5	big	med	good
1727	low	low	5	5	big	high	vgood

1728 rows × 7 columns

In [3]:

```
df.shape
```

Out[3]:

```
(1728, 7)
```

In [4]:

```
df.size
```

Out[4]:

```
12096
```

In [5]:

```
df.describe()
```

Out[5]:

	doors	persons
count	1728.000000	1728.000000
mean	3.500000	3.666667
std	1.118358	1.247580
min	2.000000	2.000000
25%	2.750000	2.000000

50%	3.500000	4.000000
75%	4.250000	5.000000
max	5.000000	5.000000

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   int64 
 3   persons   1728 non-null   int64 
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   outcome   1728 non-null   object
dtypes: int64(2), object(5)
memory usage: 94.6+ KB
```

In [7]:

```
df.isnull().sum()
```

Out[7]:

```
buying      0
maint       0
doors       0
persons     0
lug_boot    0
safety      0
outcome     0
dtype: int64
```

In [8]:

```
X=df.iloc[:,:-1]
y=df[["outcome"]]
```

In [9]:

```
X
```

Out[9]:

	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	5	5	med	med
1724	low	low	5	5	med	high
1725	low	low	5	5	big	low
1726	low	low	5	5	big	med
1727	low	low	5	5	big	high

1728 rows × 6 columns

In [10]:

```
y
```

Out[10]:

	outcome
0	unacc
1	unacc
2	unacc
3	unacc
4	unacc
...	...
1723	good
1724	vgood
1725	unacc
1726	good
1727	vgood

1728 rows × 1 columns

In [11]:

```
from sklearn.preprocessing import LabelEncoder
encoder=LabelEncoder()
X.buying=encoder.fit_transform(X.buying)
X.maint=encoder.fit_transform(X.maint)
X.lug_boot=encoder.fit_transform(X.lug_boot)
X.safety=encoder.fit_transform(X.safety)
```

In [12]:

```
X.buying.head()
```

Out[12]:

```
0    3
1    3
2    3
3    3
4    3
Name: buying, dtype: int64
```

In [13]:

```
X.head()
```

Out[13]:

	buying	maint	doors	persons	lug_boot	safety
0	3	3	2	2	2	1
1	3	3	2	2	2	2
2	3	3	2	2	2	0
3	3	3	2	2	1	1
4	3	3	2	2	1	2

In [14]:

```
from sklearn.ensemble import RandomForestClassifier
```

```
from sklearn.model_selection import train_test_split
```

In [15]:

```
X_train,X_test,y_train,y_test=train_test_split(X,y)
```

In [16]:

```
model=RandomForestClassifier()
model.fit(X_train,y_train)
model
```

```
<ipython-input-16-54c02575a96d>:2: DataConversionWarning: A column-vector y was passed wh
en a 1d array was expected. Please change the shape of y to (n_samples,), for example usi
ng ravel().
  model.fit(X_train,y_train)
```

Out[16]:

▾ RandomForestClassifier

RandomForestClassifier()

In [17]:

```
predict=model.predict(X_test)
predict
```

Out[17]:

```
array(['unacc', 'unacc', 'good', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'good',
       'unacc', 'acc', 'unacc', 'vgood', 'unacc', 'acc', 'unacc', 'acc',
       'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'good',
       'vgood', 'vgood', 'vgood', 'acc', 'acc', 'acc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc',
       'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc',
       'unacc', 'good', 'unacc', 'unacc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'good', 'unacc', 'unacc', 'acc', 'acc', 'unacc',
       'vgood', 'unacc', 'good', 'unacc', 'acc', 'unacc', 'acc', 'vgood',
       'unacc', 'acc', 'unacc', 'acc', 'unacc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'vgood', 'unacc', 'acc', 'unacc', 'unacc',
       'unacc', 'unacc', 'vgood', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'acc', 'acc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc',
       'acc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc',
       'unacc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc',
       'acc', 'acc', 'acc', 'unacc', 'acc', 'acc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'good', 'unacc',
       'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'acc', 'acc',
       'acc', 'unacc', 'acc', 'unacc', 'unacc', 'vgood', 'unacc', 'unacc',
       'unacc', 'acc', 'unacc', 'acc', 'unacc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'unacc', 'vgood', 'unacc', 'unacc', 'unacc',
       'acc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'acc',
       'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'acc',
       'vgood', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc',
       'unacc', 'unacc', 'acc', 'acc', 'acc', 'unacc', 'unacc', 'unacc',
       'unacc', 'acc', 'acc', 'acc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'vgood', 'acc', 'unacc', 'acc', 'vgood', 'unacc',
       'unacc', 'unacc', 'good', 'acc', 'unacc', 'acc', 'unacc', 'unacc',
       'unacc', 'acc', 'good', 'acc', 'unacc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'acc', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc',
```

```
       'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'good', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'vgood', 'unacc', 'unacc', 'unacc', 'unacc', 'acc',
       'unacc', 'unacc', 'good', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'acc', 'acc', 'unacc', 'unacc', 'acc', 'unacc', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'good', 'acc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'unacc',
       'vgood', 'unacc', 'unacc', 'unacc', 'acc', 'unacc', 'unacc', 'acc',
       'unacc', 'acc', 'unacc', 'good', 'unacc', 'unacc', 'acc', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc', 'unacc', 'acc', 'unacc',
       'unacc', 'unacc', 'unacc', 'acc', 'acc', 'unacc', 'unacc', 'acc',
       'unacc', 'unacc', 'unacc', 'acc', 'acc', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'good', 'unacc', 'unacc', 'unacc',
       'unacc', 'unacc', 'unacc', 'acc', 'acc', 'unacc', 'vgood', 'acc',
       'unacc', 'unacc', 'unacc', 'unacc'], dtype=object)
```

In [18]:

```
accuracy_score(y_test,predict)
```

Out[18]:

```
0.9675925925925926
```

In [19]:

```
print(classification_report(y_test,predict))
```

```
              precision    recall  f1-score   support

         acc       0.93      0.93      0.93        95
        good       0.71      0.71      0.71        14
       unacc       0.99      0.99      0.99       307
       vgood       1.00      1.00      1.00        16

    accuracy                           0.97       432
   macro avg       0.91      0.91      0.91       432
weighted avg       0.97      0.97      0.97       432
```

In [20]:

```
confusion_matrix(y_test,predict)
```

Out[20]:

```
array([[ 88,   4,   3,   0],
       [  4,  10,   0,   0],
       [  3,   0, 304,   0],
       [  0,   0,   0,  16]])
```