In [1]:
```
##importing required library
import numpy as np
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.neighbors import KNeighborsClassifier
from sklearn import metrics
from sklearn.preprocessing import LabelEncoder,OneHotEncoder
```

In [2]:
```
df=pd.read_csv('C://Users//rishu//Desktop//DATA SET/car.data')
```

In [3]:
```
pd.set_option("display.max_rows",None)
pd.set_option("display.max_columns",None)
```

In [4]:
```
df.head()
```

Out[4]:

	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 [5]:
```
##Converting the data
le=LabelEncoder()
```

In [6]:
```
df['class']=le.fit_transform(df['class'])
# df.head()
```

In [7]:
```
df['buying']=le.fit_transform(df['buying'])
# df.head()
```

In [8]:
```
df['maint']=le.fit_transform(df['maint'])
# df.head()
```

In [9]:
```
df['lug_boot']=le.fit_transform(df['lug_boot'])
# df.head()
```

In [10]:
```
df['safety']=le.fit_transform(df['safety'])
# df
```

In [11]:
```
df['persons'] = df['persons'].replace(['more'],method='pad')
# df['persons']
```

In [12]:
```
df['doors'] = df['doors'].replace(['5more'],method='pad')
#df['doors']
```

In [13]:
```
##Making a X and y variables
X=df.drop(columns="class")
y=df['class']
```

In [14]:
```
print("Shape of X :- ",X.shape)
print("Shape of y :- ",y.shape)
```

```
Shape of X :-  (1728, 6)
Shape of y :-  (1728,)
```

In [15]:
```
##spliting the data into two parts training and testing
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.30, r
```

In [16]:
```
print("Shape of X_train ",X_train.shape)
print("Shape of y_train ",y_train.shape)
print("Shape of X_test ",X_test.shape)
print("Shape of y_test ",y_test.shape)
```

```
Shape of X_train  (1209, 6)
Shape of y_train  (1209,)
Shape of X_test  (519, 6)
Shape of y_test  (519,)
```

In [17]:
```
##Let's create a model
knn=KNeighborsClassifier(n_neighbors=25,weights="uniform")
```

In [18]:
```
knn.fit(X_train,y_train)
```

Out[18]:
```
KNeighborsClassifier(n_neighbors=25)
```

In [19]:
```
y_pred=knn.predict(X_test)
```

```
C:\Users\rishu\anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `
kurtosis`), the default behavior of `mode` typically preserves the axis it
acts along. In SciPy 1.11.0, this behavior will change: the default value
of `keepdims` will become False, the `axis` over which the statistic is ta
ken will be eliminated, and the value None will no longer be accepted. Set
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [20]:
```
##accuracy
from sklearn.metrics import accuracy_score,confusion_matrix,classification_
```

In [21]:
```
print("Accuracy of Knn Algo :- \n",accuracy_score(y_test,y_pred))
```

```
Accuracy of Knn Algo :- 
 0.8901734104046243
```

In [22]:
```
print("Classifier reports of KNN:- \n",classification_report(y_test,y_pred)
```

```
Classifier reports of KNN:-
               precision    recall  f1-score   support

           0       0.83      0.69      0.76       118
           1       0.88      0.37      0.52        19
           2       0.91      0.99      0.95       358
           3       0.86      0.79      0.83        24

    accuracy                           0.89       519
   macro avg       0.87      0.71      0.76       519
weighted avg       0.89      0.89      0.88       519
```

In [23]:
```
print("Confusion metrix of knn \n",confusion_matrix(y_test,y_pred))
```

```
Confusion metrix of knn
 [[ 82   1  33   2]
 [ 10   7   1   1]
 [  4   0 354   0]
 [  3   0   2  19]]
```

In [24]:
```
print("Actual value :- ",y[100])
print("Predicted value :- ",knn.predict(X_test)[100])
```

```
Actual value :-  2
Predicted value :-  2

C:\Users\rishu\anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `
kurtosis`), the default behavior of `mode` typically preserves the axis it
acts along. In SciPy 1.11.0, this behavior will change: the default value
of `keepdims` will become False, the `axis` over which the statistic is ta
ken will be eliminated, and the value None will no longer be accepted. Set
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [25]:
```
print("Actual value :- ",y[11])
print("Predicted value :- ",knn.predict(X_test)[11])
```

```
Actual value :-  2
Predicted value :-  0

C:\Users\rishu\anaconda3\lib\site-packages\sklearn\neighbors\_classificati
on.py:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `
kurtosis`), the default behavior of `mode` typically preserves the axis it
acts along. In SciPy 1.11.0, this behavior will change: the default value
of `keepdims` will become False, the `axis` over which the statistic is ta
ken will be eliminated, and the value None will no longer be accepted. Set
`keepdims` to True or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```