In [139]:
```
import numpy as np
import pandas as pd
import sklearn
from sklearn.utils import shuffle
from sklearn.neighbors import KNeighborsClassifier
from sklearn import linear_model
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

In [140]:
```
data = pd.read_csv("car.data")
data
```

Out[140]:

	buying	maint	door	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
...	...	...	...	...	...	...	...
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

In [141]:
```
len(data)
```

Out[141]:
```
1728
```

In [142]:
```
data.dtypes
```

Out[142]:
```
buying      object
maint       object
door        object
persons     object
lug_boot    object
safety      object
class       object
dtype: object
```

In [143]:
```
data.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   door      1728 non-null   object
 3   persons   1728 non-null   object
 4   lug_boot  1728 non-null   object
 5   safety    1728 non-null   object
 6   class     1728 non-null   object
dtypes: object(7)
memory usage: 94.6+ KB
```

In [144]:
```
data.describe()
```

Out[144]:

	buying	maint	door	persons	lug_boot	safety	class
count	1728	1728	1728	1728	1728	1728	1728
unique	4	4	4	3	3	3	4
top	vhigh	vhigh	2	2	small	low	unacc
freq	432	432	432	576	576	576	1210

In [145]:
```
data.corr()
```

Out[145]:

—

In [146]:
```
data.head()
```

Out[146]:

	buying	maint	door	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 [147]:
```
data.tail()
```

Out[147]:

	buying	maint	door	persons	lug_boot	safety	class
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

In [148]:
```
data.index
```

Out[148]:
```
RangeIndex(start=0, stop=1728, step=1)
```

In [149]:
```
data.isna()
```

Out[149]:

	buying	maint	door	persons	lug_boot	safety	class
0	False	False	False	False	False	False	False
1	False	False	False	False	False	False	False
2	False	False	False	False	False	False	False
3	False	False	False	False	False	False	False
4	False	False	False	False	False	False	False
...	...	...	...	...	...	...	...
1723	False	False	False	False	False	False	False
1724	False	False	False	False	False	False	False
1725	False	False	False	False	False	False	False
1726	False	False	False	False	False	False	False
1727	False	False	False	False	False	False	False

1728 rows × 7 columns

In [150]:
```
data.isna().sum()
```

Out[150]:
```
buying      0
maint       0
door        0
persons     0
lug_boot    0
safety      0
class       0
dtype: int64
```

```
LabelEncoder
```

In [151]:
```
le = preprocessing.LabelEncoder()
```

In [152]:
```
buying = le.fit_transform(list(data['buying']))
maint = le.fit_transform(list(data['maint']))
door = le.fit_transform(list(data['door']))
persons = le.fit_transform(list(data['persons']))
lug_boot = le.fit_transform(list(data['lug_boot']))
safety = le.fit_transform(list(data['safety']))
cls = le.fit_transform(list(data['class']))
```

In [153]:
```
print(buying)
```

```
[3 3 3 ... 1 1 1]
```

In [154]:
```
predict = 'class'
```

In [155]:
```
x=list(zip(buying, maint,door,persons,lug_boot,safety))
y=list(cls)
```

In [156]:
```
x_train, x_test, y_train, y_test = train_test_split(x,y,test_size=0.1)
```

In [157]:
```
model = KNeighborsClassifier(n_neighbors=9)
```

In [158]:
```
model.fit(x_train,y_train)
```

Out[158]:
```
KNeighborsClassifier(n_neighbors=9)
```

In [159]:
```
acc = model.score(x_test,y_test)
print(acc)
```

```
0.953757225433526

C:\Users\Noah\anaconda3\lib\site-packages\sklearn\neighbors\_classification.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 taken 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 [160]:
```
predicted = model.predict(x_test)
```

```
C:\Users\Noah\anaconda3\lib\site-packages\sklearn\neighbors\_classification.p
y:228: FutureWarning: Unlike other reduction functions (e.g. `skew`, `kurtosi
s`), the default behavior of `mode` typically preserves the axis it acts alon
g. In SciPy 1.11.0, this behavior will change: the default value of `keepdims
` will become False, the `axis` over which the statistic is taken will be eli
minated, and the value None will no longer be accepted. Set `keepdims` to Tru
e or False to avoid this warning.
  mode, _ = stats.mode(_y[neigh_ind, k], axis=1)
```

In [161]:
```
names = ["unacc","acc","good","vgood"]
```

In [165]:
```
for x in range(len(predicted)):
    print("Predicted: ", names[predicted[x]], "Data: ", x_test[x],"Actual: ",
    n = model.kneighbors(([x_test[x]]),9,True)
    print(n)
```

```
(array([[1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.41421356, 1.41421356]]), array([[ 501,
326,  710, 1261,  978,  863,  418,  299, 1364]],
      dtype=int64))
Predicted:  good Data:  (2, 3, 1, 1, 1, 1) Actual:  good
(array([[1., 1., 1., 1., 1., 1., 1., 1., 1.]]), array([[ 378, 1513, 1484,
1142,  765, 1304, 1526,  815,   82]],
      dtype=int64))
Predicted:  good Data:  (2, 0, 0, 2, 1, 1) Actual:  good
(array([[1., 1., 1., 1., 1., 1., 1., 1., 1.]]), array([[ 986,  706, 1365,
153, 1028, 1349, 1489,   99,  868]],
      dtype=int64))
Predicted:  good Data:  (0, 0, 1, 0, 1, 1) Actual:  good
(array([[1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 1.41421356]]), array([[ 519,
302, 1067,  899, 1481,  310,  628,   79, 1259]],
      dtype=int64))
Predicted:  unacc Data:  (0, 1, 3, 1, 1, 0) Actual:  unacc
(array([[1.        , 1.        , 1.        , 1.        , 1.        ,
        1.        , 1.        , 1.        , 1.41421356]]), array([[ 182, 1
```