In [111]:

```
# seaborn contains the load_dataset method
# which contains no. of datasets availble online
# we will be using titanic dataset for this practical

import seaborn as sns

df = sns.load_dataset('titanic') # loads the dataset in dataframe df
df.head() # display only first 5 rows
```

Out[111]:

	survived	pclass	sex	age	sibsp	parch	fare	embarked	class	who	adult_male
0	0	3	male	22.0	1	0	7.2500	S	Third	man	True
1	1	1	female	38.0	1	0	71.2833	C	First	woman	False
2	1	3	female	26.0	0	0	7.9250	S	Third	woman	False
3	1	1	female	35.0	1	0	53.1000	S	First	woman	False
4	0	3	male	35.0	0	0	8.0500	S	Third	man	True

In [112]:

```
# features selection
df = df[['survived','pclass','age','fare']] # only considering this 4 columns
```

In [113]:

```
# fill missing / null values by 1 for no errors in calculations
df.fillna(1)
df.head()
```

Out[113]:

	survived	pclass	age	fare
0	0	3	22.0	7.2500
1	1	1	38.0	71.2833
2	1	3	26.0	7.9250
3	1	1	35.0	53.1000
4	0	3	35.0	8.0500

In [114]:

```
# x is features or independent variable
x = df.drop(['survived'],axis=1)
# y is label or dependent variable
y = df[['survived']]
```

In [117]:

```
type(x) # it returns the type of variable
```

Out[117]:

```
pandas.core.frame.DataFrame
```

In [118]:

```
x.describe() # it returns the describtion of the dataframe x
```

Out[118]:

	pclass	age	fare
count	891.000000	714.000000	891.000000
mean	2.308642	29.699118	32.204208
std	0.836071	14.526497	49.693429
min	1.000000	0.420000	0.000000
25%	2.000000	20.125000	7.910400
50%	3.000000	28.000000	14.454200
75%	3.000000	38.000000	31.000000
max	3.000000	80.000000	512.329200

In [94]:

```
x.shape # x.shape displays the no of rows and columns
# x contains 891 rows and 3 columns
```

Out[94]:

```
(891, 3)
```

In [95]:

```
y.shape
```

Out[95]:

```
(891, 1)
```

In [109]:

```
from sklearn.preprocessing import StandardScaler # for standardization

ss = StandardScaler()

standard_x = ss.fit_transform(x)
standard_y = ss.fit_transform(y)

print('\nBefore Standardization x : ',x)
print('\nAfter Standardization x : ',standard_x)
print('\nBefore Standardization y : ',y)
print('\nAfter Standardization y : ',standard_y)
```

```
Before Standardization x :       pclass   age     fare
0         3  22.0   7.2500
1         1  38.0  71.2833
2         3  26.0   7.9250
3         1  35.0  53.1000
4         3  35.0   8.0500
..      ...   ...      ...
886       2  27.0  13.0000
887       1  19.0  30.0000
888       3   NaN  23.4500
889       1  26.0  30.0000
890       3  32.0   7.7500

[891 rows x 3 columns]

After Standardization x :  [[ 0.82737724 -0.53037664 -0.50244517]
 [-1.56610693  0.57183099  0.78684529]
 [ 0.82737724 -0.25482473 -0.48885426]
```

In [119]:

```
from sklearn.preprocessing import MinMaxScaler # for Normalization

mms = MinMaxScaler()
#mms = MinMaxScaler(feature_range=(0,1))

normal_x = mms.fit_transform(x)
normal_y = mms.fit_transform(y)

print('\nBefore Normalization x : ',x)
print('\nAfter Normalization x : ',normal_x)
print('\nBefore Normalization y : ',y)
print('\nAfter Normalization y : ',normal_y)

```

```
Before Normalization x :       pclass   age     fare
0         3  22.0   7.2500
1         1  38.0  71.2833
2         3  26.0   7.9250
3         1  35.0  53.1000
4         3  35.0   8.0500
..      ...   ...      ...
886       2  27.0  13.0000
887       1  19.0  30.0000
888       3   NaN  23.4500
889       1  26.0  30.0000
890       3  32.0   7.7500

[891 rows x 3 columns]

After Normalization x :  [[1.         0.27117366 0.01415106]
 [0.         0.4722292  0.13913574]
 [1.         0.32143755 0.01546857]
```