In [1]:

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
%matplotlib inline
from sklearn import preprocessing
from scipy.stats import norm
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler, MinMaxScaler, normalize
import warnings
warnings.filterwarnings("ignore")
```

In [2]:

```
df = pd.read_csv("Breast Cancer Data.csv")
print(df.shape)
df.head()
```

```
(569, 33)
```

Out[2]:

	id	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_
0	842302	M	17.99	10.38	122.80	1001.0	0.
1	842517	M	20.57	17.77	132.90	1326.0	0.
2	84300903	M	19.69	21.25	130.00	1203.0	0.
3	84348301	M	11.42	20.38	77.58	386.1	0.
4	84358402	M	20.29	14.34	135.10	1297.0	0.

5 rows × 33 columns

In [3]:

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   id                       569 non-null    int64  
 1   diagnosis                569 non-null    object 
 2   radius_mean              569 non-null    float64
 3   texture_mean             569 non-null    float64
 4   perimeter_mean           569 non-null    float64
 5   area_mean                569 non-null    float64
 6   smoothness_mean          569 non-null    float64
 7   compactness_mean         569 non-null    float64
 8   concavity_mean           569 non-null    float64
 9   concave points_mean      569 non-null    float64
 10  symmetry_mean            569 non-null    float64
 11  fractal_dimension_mean   569 non-null    float64
 12  radius_se                569 non-null    float64
 13  texture_se               569 non-null    float64
 14  perimeter_se             569 non-null    float64
 15  area_se                  569 non-null    float64
 16  smoothness_se            569 non-null    float64
 17  compactness_se           569 non-null    float64
 18  concavity_se             569 non-null    float64
 19  concave points_se        569 non-null    float64
 20  symmetry_se              569 non-null    float64
 21  fractal_dimension_se     569 non-null    float64
 22  radius_worst             569 non-null    float64
 23  texture_worst            569 non-null    float64
 24  perimeter_worst          569 non-null    float64
 25  area_worst               569 non-null    float64
 26  smoothness_worst         569 non-null    float64
 27  compactness_worst        569 non-null    float64
 28  concavity_worst          569 non-null    float64
 29  concave points_worst     569 non-null    float64
 30  symmetry_worst           569 non-null    float64
 31  fractal_dimension_worst  569 non-null    float64
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

In [4]:

```
df.isnull().sum() #check Missing Value
```

Out[4]:

```
id                          0
diagnosis                   0
radius_mean                 0
texture_mean                0
perimeter_mean              0
area_mean                   0
smoothness_mean             0
compactness_mean            0
concavity_mean              0
concave points_mean         0
symmetry_mean               0
fractal_dimension_mean      0
radius_se                   0
texture_se                  0
perimeter_se                0
area_se                     0
smoothness_se               0
compactness_se              0
concavity_se                0
concave points_se           0
symmetry_se                 0
fractal_dimension_se        0
radius_worst                0
texture_worst               0
perimeter_worst             0
area_worst                  0
smoothness_worst            0
compactness_worst           0
concavity_worst             0
concave points_worst        0
symmetry_worst              0
fractal_dimension_worst     0
Unnamed: 32               569
dtype: int64
```

From the above observation Unnamed: 32 column contains all the Null values, So it would be better to drop the column

In [5]:

```
df.drop('Unnamed: 32',axis = 1,inplace = True) # Drop the Column Containing Missing Value
df.isnull().sum().sum() # Recheck the Missing Value is present or not
```

Out[5]:

```
0
```

In [6]:

```
df.describe()
```

Out[6]:

	id	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_m
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	569.000
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	0.096
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	0.014
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	0.052
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	0.086
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	0.095
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	0.105
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	0.163

8 rows × 31 columns

In [7]:

```
df.diagnosis.unique() # Check The types of values of Diagnosis Present In dataset
```

Out[7]:

```
array(['M', 'B'], dtype=object)
```

In [8]:

```
df['diagnosis'].value_counts() # Count number of Malignant (M) or Benign (B) Cells
```

Out[8]:

```
B    357
M    212
Name: diagnosis, dtype: int64
```

Feature elemination

In [9]:

```
# feature is not for our use as it consist of id of the patient
df.drop('id',axis=1,inplace= True)
df.isnull().sum().sum()
```

Out[9]:

```
0
```

Feature encoding

In [10]:

```
le = preprocessing.LabelEncoder()
df['diagnosis']=le.fit_transform(df['diagnosis'])
df['diagnosis'].unique()
```

Out[10]:

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

Spliting Dataset

In [11]:

```
data_std = df.copy()
print(data_std.shape,data_std.isnull().sum().sum())
data_std.head()
```

```
(569, 31) 0
```

Out[11]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	1	17.99	10.38	122.80	1001.0	0.11840	
1	1	20.57	17.77	132.90	1326.0	0.08474	
2	1	19.69	21.25	130.00	1203.0	0.10960	
3	1	11.42	20.38	77.58	386.1	0.14250	
4	1	20.29	14.34	135.10	1297.0	0.10030	

5 rows × 31 columns

In [12]:

```
X =data_std.drop(['diagnosis'],axis = 1)
y = data_std['diagnosis']
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.3)
X_train.shape , X_test.shape, y_train.shape, y_test.shape
```

Out[12]:

```
((398, 30), (171, 30), (398,), (171,))
```

In [13]:

```
sc = StandardScaler()
X_train_std = sc.fit_transform(X_train) #Standardize the training Dataset
X_test_std = sc.transform(X_test) # Standardize the testing Dataset
X_train_std = pd.DataFrame(X_train_std,columns=df.columns[:-1]) # COnvert to dataframe
X_train_std.head()
```

Out[13]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	-0.803669	0.142307	-0.742539	-0.732525	-1.118956	0.264933	
1	0.118791	-1.250132	0.212063	-0.037601	0.529713	1.488118	
2	1.022425	-0.744849	1.068478	0.890467	0.774494	2.067029	
3	1.673256	0.307630	1.633447	1.667615	-0.301101	0.511437	
4	1.006288	0.305302	0.974966	0.852538	0.565710	0.494630	

5 rows × 30 columns

In [14]:

```
X_train_normalize = normalize(X_train) # normalize training Dataset
X_test_normalize = normalize(X_test) # Normalize Testing Dataset
X_train_normalize = pd.DataFrame(X_train_normalize,columns=df.columns[:-1]) # Convert it in
X_train_normalize.head()
```

Out[14]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	0.018747	0.033232	0.122739	0.656151	0.000134	0.000197	
1	0.013934	0.013261	0.093166	0.622342	0.000098	0.000174	
2	0.009434	0.008441	0.062823	0.525789	0.000056	0.000112	
3	0.009707	0.009802	0.063876	0.615999	0.000043	0.000062	
4	0.010674	0.012258	0.069895	0.588409	0.000061	0.000077	

5 rows × 30 columns

In [15]:

```
minMax = MinMaxScaler()
X_train_mm = minMax.fit_transform(X_train) #Apply min-max scaler on  the training Dataset
X_test_mm = minMax.transform(X_test) #Apply min-max scaler on  the testing Dataset
X_train_mm = pd.DataFrame(X_train_mm,columns=df.columns[:-1]) # convert to dataframe
X_train_mm.head()
```

Out[15]:

	diagnosis	radius_mean	texture_mean	perimeter_mean	area_mean	smoothness_mean	com
0	0.202518	0.346635	0.206827	0.106299	0.248894	0.302804	
1	0.364854	0.144403	0.376132	0.217434	0.455629	0.503711	
2	0.523877	0.217788	0.528022	0.365854	0.486323	0.598798	
3	0.638412	0.370646	0.628222	0.490138	0.351449	0.343292	
4	0.521037	0.370308	0.511437	0.359788	0.460143	0.340531	

5 rows × 30 columns