In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
from sklearn.datasets import load_breast_cancer
cancer = load_breast_cancer()
```

In [2]:
```
cancer.keys()
```

Out[2]:
```
dict_keys(['data', 'target', 'target_names', 'DESCR', 'feature_names'])
```

In [3]:
```
cancer["data"]
```

Out[3]:
```
array([[1.799e+01, 1.038e+01, 1.228e+02, ..., 2.654e-01, 4.601e-01,
         1.189e-01],
        [2.057e+01, 1.777e+01, 1.329e+02, ..., 1.860e-01, 2.750e-01,
         8.902e-02],
        [1.969e+01, 2.125e+01, 1.300e+02, ..., 2.430e-01, 3.613e-01,
         8.758e-02],
        ...,
        [1.660e+01, 2.808e+01, 1.083e+02, ..., 1.418e-01, 2.218e-01,
         7.820e-02],
        [2.060e+01, 2.933e+01, 1.401e+02, ..., 2.650e-01, 4.087e-01,
         1.240e-01],
        [7.760e+00, 2.454e+01, 4.792e+01, ..., 0.000e+00, 2.871e-01,
         7.039e-02]])
```

In [4]:
```
cancer["target"]
```

Out[4]:
```
array([0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1,
       0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 1, 1, 1, 1, 0, 1, 0, 0,
       1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 0, 0,
       1, 1, 1, 0, 1, 1, 0, 0, 1, 1, 1, 0, 0, 1, 1, 1, 1, 0, 1, 1, 0, 1,
       1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 1, 0, 0, 1, 1, 1, 0, 0, 1, 0, 1, 0,
       0, 1, 0, 0, 1, 1, 0, 1, 1, 0, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 0, 0, 1, 0, 1, 1, 0, 0, 1, 1, 0, 0, 1, 1, 1,
       1, 0, 1, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 0, 0,
       0, 0, 1, 0, 0, 0, 1, 0, 1, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 1, 0, 0,
       1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 1, 0, 1, 1, 0, 0, 1, 0, 1, 1,
       1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 0, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 0, 1, 1, 1, 1, 0, 0,
       0, 1, 1, 1, 1, 0, 1, 0, 1, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0,
       0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 1, 0, 0, 0, 1, 0, 0,
       1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 1, 0, 1, 1, 0, 0, 1, 1,
       1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 0, 1, 1, 0,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 0, 1, 1, 1, 1,
       1, 0, 1, 1, 0, 1, 0, 1, 1, 0, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0,
       1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 1, 1, 1, 1,
       1, 1, 1, 0, 1, 0, 1, 1, 0, 1, 1, 1, 1, 1, 0, 0, 1, 0, 1, 0, 1, 1,
       1, 1, 1, 0, 1, 1, 0, 1, 0, 1, 0, 0, 1, 1, 1, 0, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 0, 1, 0, 0, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1,
       1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 0, 0, 0, 0, 0, 0, 1])
```

In [5]:
```
print(cancer['DESCR'])  #dataset description
```

```
Breast Cancer Wisconsin (Diagnostic) Database
=============================================

Notes
-----
Data Set Characteristics:
    :Number of Instances: 569

    :Number of Attributes: 30 numeric, predictive attributes and the class

    :Attribute Information:
        - radius (mean of distances from center to points on the perimeter)
        - texture (standard deviation of gray-scale values)
        - perimeter
        - area
        - smoothness (local variation in radius lengths)
        - compactness (perimeter^2 / area - 1.0)
        - concavity (severity of concave portions of the contour)
        - concave points (number of concave portions of the contour)
```

In [6]:
```
x = cancer['data']
y = cancer['target']
```

In [7]:
```
x_df = pd.DataFrame(x)
x_df.head()
```

Out[7]:

	0	1	2	3	4	5	6	7	8	9	...	20	2
0	17.99	10.38	122.80	1001.0	0.11840	0.27760	0.3001	0.14710	0.2419	0.07871	...	25.38	17.3
1	20.57	17.77	132.90	1326.0	0.08474	0.07864	0.0869	0.07017	0.1812	0.05667	...	24.99	23.4
2	19.69	21.25	130.00	1203.0	0.10960	0.15990	0.1974	0.12790	0.2069	0.05999	...	23.57	25.5
3	11.42	20.38	77.58	386.1	0.14250	0.28390	0.2414	0.10520	0.2597	0.09744	...	14.91	26.5
4	20.29	14.34	135.10	1297.0	0.10030	0.13280	0.1980	0.10430	0.1809	0.05883	...	22.54	16.6

5 rows × 30 columns

In [8]:

```
y_df = pd.DataFrame(y)
y_df.tail(10)
```

Out[8]:

	0
559	1
560	1
561	1
562	0
563	0
564	0
565	0
566	0
567	0
568	1

In [9]:

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y)
```

In [10]:

```
from sklearn.preprocessing import StandardScaler
scalar = StandardScaler()
scalar.fit(x_train)
```

Out[10]:

```
StandardScaler(copy=True, with_mean=True, with_std=True)
```

In [11]:

```
x_train=scalar.transform(x_train)
x_test=scalar.transform(x_test)
```

In [12]:

```
x_train_df = pd.DataFrame(x_train)
x_train_df.head()
```

Out[12]:

	0	1	2	3	4	5	6	7	8
0	-0.269022	-1.056637	-0.323814	-0.317785	-1.628520	-1.021009	-0.713341	-0.598087	-1.596108
1	-0.641466	-1.009633	-0.631010	-0.672462	1.341373	0.268735	-0.977790	-0.818031	2.414279
2	1.796348	0.466295	1.846605	1.862197	0.828232	1.075170	1.642538	1.589084	0.130639
3	1.460584	0.997441	1.490731	1.394025	0.420531	0.916829	1.466660	0.995416	0.616520
4	2.197007	0.621408	2.231111	2.321858	0.673587	1.645934	1.971518	2.559798	0.040938

5 rows × 30 columns

In [13]:

```
from sklearn.neural_network import MLPClassifier #multilevel perceptron
```

In [14]:

```
mlp = MLPClassifier(hidden_layer_sizes=(30,30,30,30,30,30,30,30))
```

In [15]:

```
mlp.fit(x_train,y_train) #training of neural network model
```

Out[15]:

```
MLPClassifier(activation='relu', alpha=0.0001, batch_size='auto', beta_1=0.9,
              beta_2=0.999, early_stopping=False, epsilon=1e-08,
              hidden_layer_sizes=(30, 30, 30, 30, 30, 30, 30, 30),
              learning_rate='constant', learning_rate_init=0.001, max_iter=200,
              momentum=0.9, nesterovs_momentum=True, power_t=0.5,
              random_state=None, shuffle=True, solver='adam', tol=0.0001,
              validation_fraction=0.1, verbose=False, warm_start=False)
```

In [16]:

```
predictions = mlp.predict(x_test)
```

In [17]:

```
from sklearn.metrics import confusion_matrix
print(confusion_matrix(y_test,predictions))
```

```
[[48  2]
 [ 1 92]]
```

In [18]:

```
from sklearn.metrics import accuracy_score
accuracy_nn=round(accuracy_score(predictions,y_test)*100,2)
print("Accuracy of this model is ",accuracy_nn,"%")
```

```
Accuracy of this model is  97.9 %
```

In [19]:

```
from sklearn.ensemble import RandomForestClassifier

model_rand=RandomForestClassifier(n_estimators=100)
model_rand.fit(x_train,y_train)

predicted_rand=model_rand.predict(x_test)

from sklearn.metrics import confusion_matrix
print(confusion_matrix(predicted_rand,y_test))

from sklearn.metrics import accuracy_score

accuracy=round(accuracy_score(predicted_rand,y_test)*100,2)
print("Accuracy of this model is ",accuracy,"%")
```

```
[[45  1]
 [ 5 92]]
Accuracy of this model is  95.8 %
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: