Breast Cancer Classification.

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import sklearn.datasets
```

Data Importing and Preprocessing

In [2]:
```
columns_name = ['ID','DIAGNOSIS','RADIUS_MEAN','TEXTURE_MEAN','PERIMETER_MEAN',
               'COMPACTNESS_MEAN','CONCATIVITY_MEAN','CONCAVE_POINTS_MEAN','SYMMET
               'PERIMETER_SE', 'AREA_SE', 'SMOOTHNESS_SE', 'COMPACTNESS_SE', 'CONC
               'RADIUS_WORST', 'TEXTURE_WORST', 'PERIMETER_WORST', 'AREA_WORST', '
              ]
```

In [3]:
```
Breast_Data = pd.read_csv("wdbc.data.csv", names= columns_name)
```

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

Out[4]:

	ID	DIAGNOSIS	RADIUS_MEAN	TEXTURE_MEAN	PERIMETER_MEAN	AREA_MEAN	S
0	842302	M	17.99	10.38	122.80	1001.0	
1	842517	M	20.57	17.77	132.90	1326.0	
2	84300903	M	19.69	21.25	130.00	1203.0	
3	84348301	M	11.42	20.38	77.58	386.1	
4	84358402	M	20.29	14.34	135.10	1297.0	

5 rows × 32 columns

In [5]:
```
Breast_Data.tail()
```

Out[5]:

	ID	DIAGNOSIS	RADIUS_MEAN	TEXTURE_MEAN	PERIMETER_MEAN	AREA_MEAN	S
564	926424	M	21.56	22.39	142.00	1479.0	
565	926682	M	20.13	28.25	131.20	1261.0	
566	926954	M	16.60	28.08	108.30	858.1	
567	927241	M	20.60	29.33	140.10	1265.0	
568	92751	B	7.76	24.54	47.92	181.0	

5 rows × 32 columns

Getting Rows and Columns of Dataset

In [6]:
```
Breast_Data.shape
```

Out[6]:
```
(569, 32)
```

Getting some information about the Dataset

In [7]:
```
Breast_Data.info
```

Out[7]:
```
<bound method DataFrame.info of          ID DIAGNOSIS  RADIUS_MEAN  TEXTURE_
MEAN  PERIMETER_MEAN  AREA_MEAN  \
0      842302         M        17.99         10.38          122.80     1001.0
1      842517         M        20.57         17.77          132.90     1326.0
2    84300903         M        19.69         21.25          130.00     1203.0
3    84348301         M        11.42         20.38           77.58      386.1
4    84358402         M        20.29         14.34          135.10     1297.0
..        ...       ...          ...           ...             ...        ...
564    926424         M        21.56         22.39          142.00     1479.0
565    926682         M        20.13         28.25          131.20     1261.0
566    926954         M        16.60         28.08          108.30      858.1
567    927241         M        20.60         29.33          140.10     1265.0
568     92751         B         7.76         24.54           47.92      181.0

     SMOOTHNESS_MEAN  COMPACTNESS_MEAN  CONCATIVITY_MEAN  CONCAVE_POINTS_MEAN
\
0            0.11840           0.27760           0.30010              0.14710
1            0.08474           0.07864           0.08690              0.07017
2            0.10960           0.15990           0.19740              0.12790
3            0.14250           0.28390           0.24140              0.10520
4            0.10030           0.13280           0.19800              0.10430
..               ...               ...               ...                  ...
564          0.11100           0.11590           0.24390              0.13890
565          0.09780           0.10340           0.14400              0.09791
566          0.08455           0.10230           0.09251              0.05302
567          0.11780           0.27700           0.35140              0.15200
568          0.05263           0.04362           0.00000              0.00000

     ...  RADIUS_WORST  TEXTURE_WORST  PERIMETER_WORST  AREA_WORST  \
0    ...        25.380          17.33           184.60      2019.0
1    ...        24.990          23.41           158.80      1956.0
2    ...        23.570          25.53           152.50      1709.0
3    ...        14.910          26.50            98.87       567.7
4    ...        22.540          16.67           152.20      1575.0
..   ...           ...            ...              ...         ...
564  ...        25.450          26.40           166.10      2027.0
565  ...        23.690          38.25           155.00      1731.0
566  ...        18.980          34.12           126.70      1124.0
567  ...        25.740          39.42           184.60      1821.0
568  ...         9.456          30.37            59.16       268.6

     SMOOTHNESS_WORST  COMPACTNESS_WORST  CONCATIVITY_WORST  \
0             0.16220            0.66560             0.7119
1             0.12380            0.18660             0.2416
2             0.14440            0.42450             0.4504
3             0.20980            0.86630             0.6869
4             0.13740            0.20500             0.4000
..                ...                ...                ...
564           0.14100            0.21130             0.4107
565           0.11660            0.19220             0.3215
566           0.11390            0.30940             0.3403
567           0.16500            0.86810             0.9387
568           0.08996            0.06444             0.0000

     CONCAVE_POINTS_WORST  SYMMETRY_WORST  FRACTAL_DIMENSION_WORST
0                  0.2654          0.4601                  0.11890
1                  0.1860          0.2750                  0.08902
2                  0.2430          0.3613                  0.08758
3                  0.2575          0.6638                  0.17300
4                  0.1625          0.2364                  0.07678
```

```
..                 ...           ...                 ...
564             0.2216        0.2060             0.07115
565             0.1628        0.2572             0.06637
566             0.1418        0.2218             0.07820
567             0.2650        0.4087             0.12400
568             0.0000        0.2871             0.07039

[569 rows x 32 columns]>
```

In [8]:
```
Breast_Data.describe()
```

Out[8]:

	ID	RADIUS_MEAN	TEXTURE_MEAN	PERIMETER_MEAN	AREA_MEAN	SMOO
count	5.690000e+02	569.000000	569.000000	569.000000	569.000000	
mean	3.037183e+07	14.127292	19.289649	91.969033	654.889104	
std	1.250206e+08	3.524049	4.301036	24.298981	351.914129	
min	8.670000e+03	6.981000	9.710000	43.790000	143.500000	
25%	8.692180e+05	11.700000	16.170000	75.170000	420.300000	
50%	9.060240e+05	13.370000	18.840000	86.240000	551.100000	
75%	8.813129e+06	15.780000	21.800000	104.100000	782.700000	
max	9.113205e+08	28.110000	39.280000	188.500000	2501.000000	

8 rows × 31 columns

Getting Target points from Datasets B: Benign, M: Malignant

In [9]:
```
Breast_Data['DIAGNOSIS'].value_counts()
```

Out[9]:
```
B    357
M    212
Name: DIAGNOSIS, dtype: int64
```

Getting Mean of Malignant and Benign Data from dataset (GroupBy).

In [10]:
```
Breast_Data_Grouped = Breast_Data.groupby('DIAGNOSIS')
print(Breast_Data_Grouped)
```

```
<pandas.core.groupby.generic.DataFrameGroupBy object at 0x7fe273106430>
```

In [11]:
```
Breast_Data_Grouped.mean()
```

Out[11]:

	ID	RADIUS_MEAN	TEXTURE_MEAN	PERIMETER_MEAN	AREA_MEAN	S
DIAGNOSIS						
B	2.654382e+07	12.146524	17.914762	78.075406	462.790196	
M	3.681805e+07	17.462830	21.604906	115.365377	978.376415	

2 rows × 31 columns

In [ ]:

Converting Diagnosis column to 0,1 for training the model.

In [12]:
```
from sklearn.preprocessing import LabelEncoder
```

In [13]:
```
lb_Diagnosis = LabelEncoder()
```

In [14]:
```
Breast_Data['Label_Diagnosis'] = lb_Diagnosis.fit_transform(Breast_Data['DIAGNO
```

Created Label_Diagnosis column where M: 1 and B:0

In [16]:
```
Breast_Data['Label_Diagnosis'].info
```

Out[16]:
```
<bound method Series.info of 0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: Label_Diagnosis, Length: 569, dtype: int64>
```

Training the Dataset and Splitting into Test and Training variables.

In [20]:
```
from sklearn import preprocessing
from sklearn.model_selection import train_test_split
```

In [21]:
```
X = Breast_Data.drop(columns=['DIAGNOSIS', 'Label_Diagnosis'], axis= 1)
Y = Breast_Data['Label_Diagnosis']
```

In [23]:
```
X.shape, Y.shape
```

Out[23]:
```
((569, 31), (569,))
```

In [24]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.2, random
```

In [26]:
```
X_train.shape, X_test.shape, Y_train.shape, Y_test.shape
```

Out[26]:
```
((455, 31), (114, 31), (455,), (114,))
```

In [27]:
```
Y_train
```

Out[27]:
```
560    0
428    0
198    1
203    1
41     1
      ..
299    0
534    0
493    0
527    0
168    1
Name: Label_Diagnosis, Length: 455, dtype: int64
```

Scaling the Training and Tsting Dataframe "As we know that remaining columns is a Numerical Data so it would be beneficial for sklearn's regresstion model understand the data while training"

In [72]:
```
Scaler = preprocessing.StandardScaler()
```

In [29]:
```
X_trainS = Scaler.fit_transform(X_train)
X_testS = Scaler.fit_transform(X_test)
```

In [30]:
```
X_trainS.shape, X_testS.shape
```

Out[30]:
```
((455, 31), (114, 31))
```

Importing Tensorflow and keras to setup Neural network for classification

In [31]:
```
import tensorflow as tf
tf.random.set_seed(3)
```

```
2022-12-07 15:44:40.380632: I tensorflow/core/platform/cpu_feature_guard.cc:19
3] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical operati
ons:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate co
mpiler flags.
```

Making neural network layers using keras, Hidden Layer

In [32]:
```
from tensorflow import keras
MODEL = keras.Sequential([
                          keras.layers.Flatten(input_shape=(31,)),
                          keras.layers.Dense(20, activation='relu'),
                          keras.layers.Dense(2, activation='sigmoid')
])
```

```
2022-12-07 15:44:48.893139: I tensorflow/core/platform/cpu_feature_guard.cc:19
3] This TensorFlow binary is optimized with oneAPI Deep Neural Network Library
(oneDNN) to use the following CPU instructions in performance-critical operati
ons:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate co
mpiler flags.
```

In [33]:
```
MODEL.compile(
            optimizer='adam',
```

```
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy']
)
```

Training the Neural network

In [51]:
```
trained_model = MODEL.fit(X_trainS, Y_train, epochs=25, validation_split=0.01)
```

```
Epoch 1/25
15/15 [==============================] - 0s 7ms/step - loss: 0.0386 - accurac
y: 0.9889 - val_loss: 1.7687e-04 - val_accuracy: 1.0000
Epoch 2/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0383 - accurac
y: 0.9889 - val_loss: 1.7642e-04 - val_accuracy: 1.0000
Epoch 3/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0376 - accurac
y: 0.9889 - val_loss: 1.7509e-04 - val_accuracy: 1.0000
Epoch 4/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0370 - accurac
y: 0.9889 - val_loss: 1.5607e-04 - val_accuracy: 1.0000
Epoch 5/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0364 - accurac
y: 0.9911 - val_loss: 1.4801e-04 - val_accuracy: 1.0000
Epoch 6/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0359 - accurac
y: 0.9889 - val_loss: 1.4849e-04 - val_accuracy: 1.0000
Epoch 7/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0356 - accurac
y: 0.9889 - val_loss: 1.4262e-04 - val_accuracy: 1.0000
Epoch 8/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0352 - accurac
y: 0.9911 - val_loss: 1.3385e-04 - val_accuracy: 1.0000
Epoch 9/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0346 - accurac
y: 0.9889 - val_loss: 1.1884e-04 - val_accuracy: 1.0000
Epoch 10/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0341 - accurac
y: 0.9911 - val_loss: 1.2170e-04 - val_accuracy: 1.0000
Epoch 11/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0335 - accurac
y: 0.9911 - val_loss: 1.1555e-04 - val_accuracy: 1.0000
Epoch 12/25
15/15 [==============================] - 0s 3ms/step - loss: 0.0330 - accurac
y: 0.9911 - val_loss: 1.1140e-04 - val_accuracy: 1.0000
Epoch 13/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0327 - accurac
y: 0.9911 - val_loss: 1.1400e-04 - val_accuracy: 1.0000
Epoch 14/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0321 - accurac
y: 0.9911 - val_loss: 1.0945e-04 - val_accuracy: 1.0000
Epoch 15/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0317 - accurac
y: 0.9911 - val_loss: 1.0046e-04 - val_accuracy: 1.0000
Epoch 16/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0313 - accurac
y: 0.9933 - val_loss: 9.8982e-05 - val_accuracy: 1.0000
Epoch 17/25
15/15 [==============================] - 0s 3ms/step - loss: 0.0308 - accurac
y: 0.9933 - val_loss: 9.2450e-05 - val_accuracy: 1.0000
Epoch 18/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0303 - accurac
y: 0.9933 - val_loss: 8.5132e-05 - val_accuracy: 1.0000
Epoch 19/25
15/15 [==============================] - 0s 5ms/step - loss: 0.0299 - accurac
y: 0.9933 - val_loss: 8.3368e-05 - val_accuracy: 1.0000
Epoch 20/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0296 - accurac
y: 0.9933 - val_loss: 8.2724e-05 - val_accuracy: 1.0000
```

```
Epoch 21/25
15/15 [==============================] - 0s 5ms/step - loss: 0.0293 - accurac
y: 0.9933 - val_loss: 7.5859e-05 - val_accuracy: 1.0000
Epoch 22/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0287 - accurac
y: 0.9933 - val_loss: 7.6098e-05 - val_accuracy: 1.0000
Epoch 23/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0284 - accurac
y: 0.9933 - val_loss: 7.2927e-05 - val_accuracy: 1.0000
Epoch 24/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0281 - accurac
y: 0.9933 - val_loss: 6.8851e-05 - val_accuracy: 1.0000
Epoch 25/25
15/15 [==============================] - 0s 4ms/step - loss: 0.0278 - accurac
y: 0.9933 - val_loss: 6.6777e-05 - val_accuracy: 1.0000
```

In [61]:

```
trained_model.params
```

Out[61]:

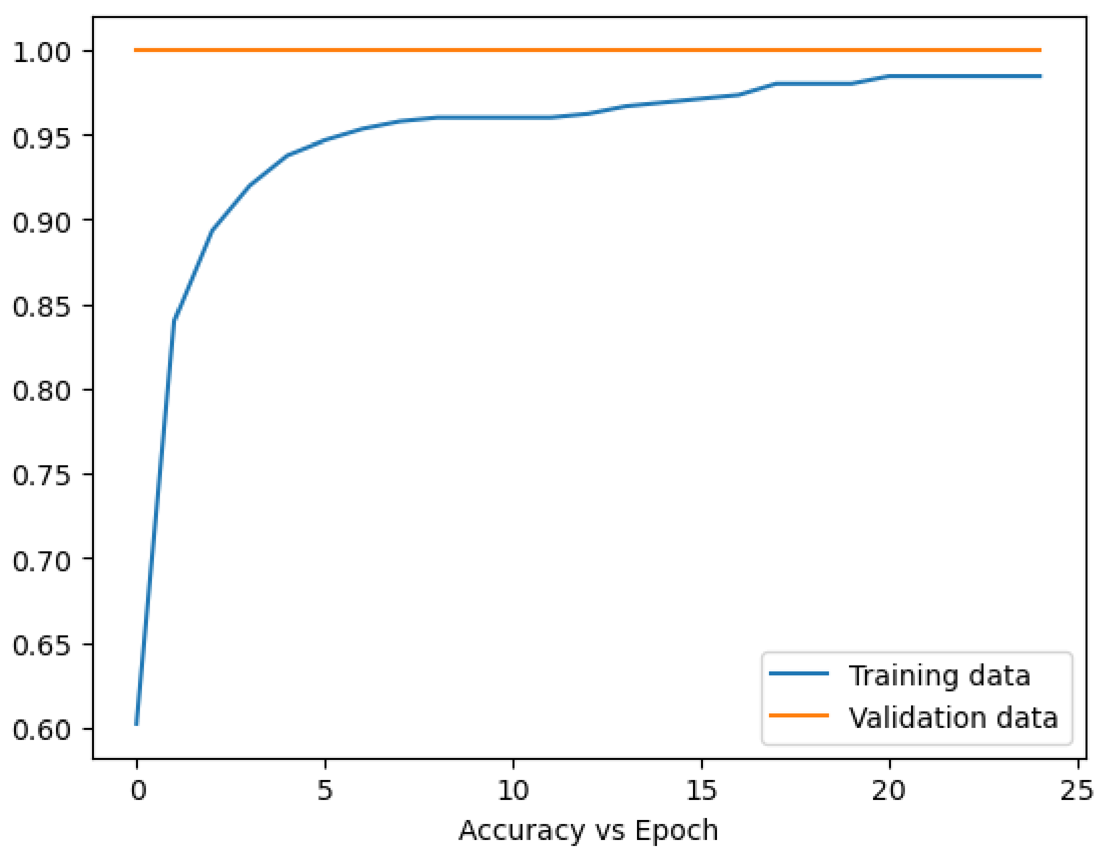
```
{'verbose': 1, 'epochs': 25, 'steps': 15}
```

In [195…

```
%matplotlib inline
plt.plot(trained_model.history['accuracy'])
plt.plot(trained_model.history['val_accuracy'])
plt.xlabel('Accuracy vs Epoch')
plt.legend(['Training data', 'Validation data'], loc = 'lower right')
```

Out[195]:

```
<matplotlib.legend.Legend at 0x7fe24c28ea60>
```

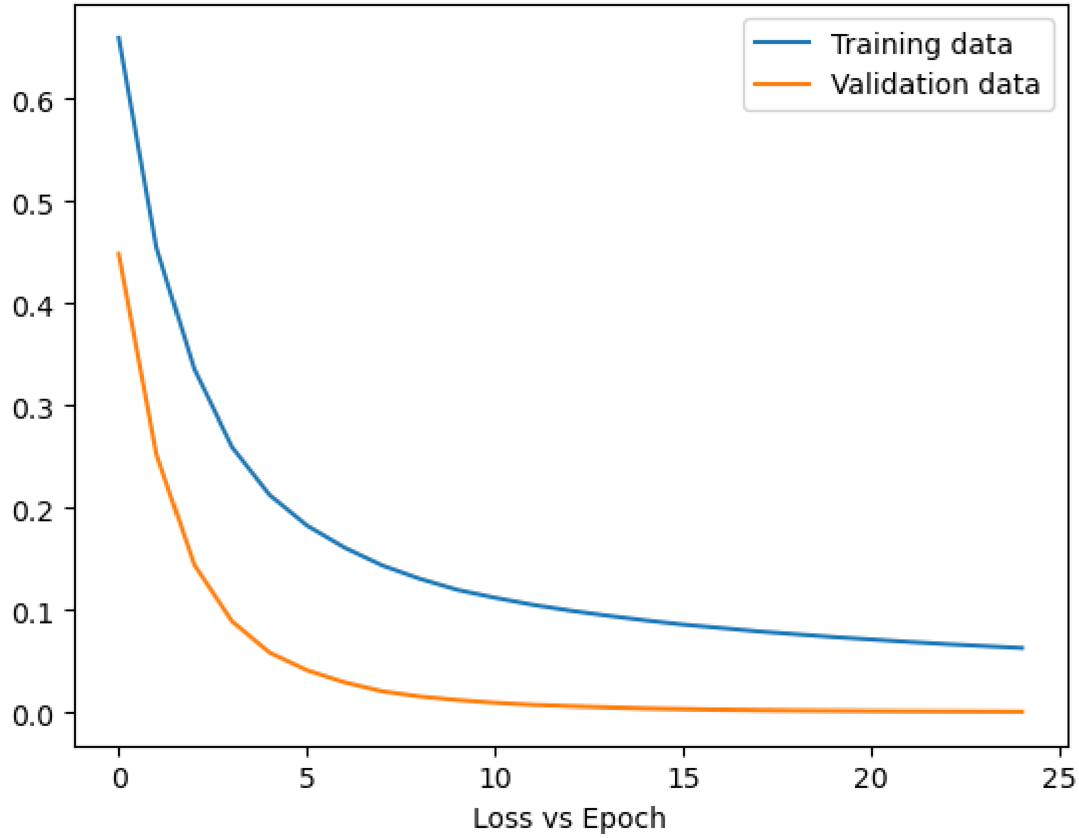


In [97]:

```
%matplotlib inline
plt.plot(trained_model.history['loss'])
plt.plot(trained_model.history['val_loss'])
```

```
plt.xlabel('Loss vs Epoch')
plt.legend(['Training data', 'Validation data'], loc = 'upper right')
```

Out[97]: <matplotlib.legend.Legend at 0x7fe24ac2e730>



In [64]:
```
trained_model.history['accuracy']
```

Out[64]:
```
[0.602222204208374,
 0.8399999737739563,
 0.8933333158493042,
 0.9200000166893005,
 0.9377777576446533,
 0.9466666579246521,
 0.95333331823349,
 0.9577777981758118,
 0.9599999785423279,
 0.9599999785423279,
 0.9599999785423279,
 0.9599999785423279,
 0.9622222185134888,
 0.9666666388511658,
 0.9688888788223267,
 0.9711111187934875,
 0.9733333587646484,
 0.9800000190734863,
 0.9800000190734863,
 0.9800000190734863,
 0.9844444394111633,
 0.9844444394111633,
 0.9844444394111633,
 0.9844444394111633,
 0.9844444394111633]
```

Finding The accuracy Of the model

In [66]:
```
Loss, Accuracy = MODEL.evaluate(X_testS, Y_test)
```

```
4/4 [==============================] - 0s 3ms/step - loss: 0.0879 - accuracy:
0.9737
```

In [67]:
```
print(Loss, Accuracy)
```

```
0.08787020295858383 0.9736841917037964
```

In [68]:
```
MODEL.predict(X_testS)
```

```
4/4 [==============================] - 0s 2ms/step
```

Out[68]:
```
array([[7.31051564e-01, 1.60803854e-01],
       [6.40998840e-01, 1.19231343e-01],
       [9.37755644e-01, 7.71941384e-04],
       [1.46365725e-10, 9.99996722e-01],
       [7.75570273e-01, 1.45885289e-01],
       [2.51738061e-06, 9.99234438e-01],
       [7.65389800e-01, 3.26402187e-02],
       [9.77818191e-01, 5.78860636e-04],
       [9.34239268e-01, 3.35949101e-03],
       [9.51056778e-01, 3.64095857e-03],
       [2.69510746e-01, 5.29248536e-01],
       [8.63291442e-01, 2.08233427e-02],
       [3.33902031e-01, 5.19329421e-02],
       [7.74546206e-01, 5.58350533e-02],
       [8.96992207e-01, 2.01532012e-03],
       [1.39508245e-03, 8.70443106e-01],
       [9.41922307e-01, 2.42094253e-03],
       [9.88540828e-01, 1.83954940e-03],
       [8.10406327e-01, 6.12178305e-03],
       [1.59763676e-05, 9.99054551e-01],
       [7.55457938e-01, 8.93481309e-04],
       [9.81762886e-01, 1.28782354e-03],
       [9.41038787e-01, 2.99099670e-03],
       [9.88616705e-01, 7.75113702e-04],
       [9.31395590e-01, 1.40412310e-02],
       [3.82299797e-04, 9.89704430e-01],
       [9.33316231e-01, 1.08463056e-02],
       [8.33260000e-01, 8.09682533e-02],
       [1.38626585e-03, 9.58235323e-01],
       [5.59933134e-04, 9.76477087e-01],
       [9.55433547e-01, 1.30898431e-02],
       [9.30512309e-01, 5.04098088e-03],
       [9.76345599e-01, 2.23975838e-03],
       [1.08914362e-08, 9.99970734e-01],
       [3.70614871e-05, 9.96860445e-01],
       [9.55898643e-01, 9.72562749e-03],
       [9.46212113e-01, 5.24495146e-04],
       [7.72765160e-01, 3.02293189e-02],
       [9.82399225e-01, 9.01362626e-04],
       [9.68568623e-01, 3.78753850e-03],
       [3.20463944e-09, 9.99991238e-01],
       [4.50073220e-02, 8.21912766e-01],
       [7.52093434e-01, 2.11587059e-03],
       [9.91524577e-01, 9.89714987e-04],
       [7.18579628e-03, 9.57393646e-01],
       [7.09191144e-01, 1.22571678e-03],
       [9.95527565e-01, 1.83623939e-04],
       [8.86224985e-01, 7.61702948e-04],
       [6.74261514e-07, 9.99415815e-01],
       [1.28702656e-03, 9.76919532e-01],
       [9.74253297e-01, 1.78660871e-03],
       [3.64623405e-02, 7.34966755e-01],
       [4.88777280e-01, 2.46961400e-01],
       [9.05061066e-01, 3.91130336e-03],
       [9.79775548e-01, 4.70038998e-04],
       [4.95718181e-01, 2.34145314e-01],
       [7.24772930e-01, 6.43579066e-02],
       [9.47778225e-01, 3.16638208e-04],
       [2.30452002e-04, 9.63313103e-01],
       [9.84664559e-01, 2.54050177e-03],
```

```
       [8.15996885e-01, 7.85517767e-02],
       [2.14043539e-03, 9.31277215e-01],
       [9.86494899e-01, 1.08903891e-03],
       [8.36508843e-05, 9.95497286e-01],
       [4.16360563e-03, 8.83585155e-01],
       [9.31065381e-01, 2.64307903e-03],
       [1.48323625e-06, 9.99362111e-01],
       [2.74783233e-03, 9.63605404e-01],
       [4.38670486e-01, 1.14846162e-01],
       [2.60057777e-01, 6.23337217e-02],
       [6.14540651e-02, 6.10680342e-01],
       [2.29546975e-04, 9.92862225e-01],
       [9.46358562e-01, 2.77371169e-03],
       [1.52478134e-02, 9.22355473e-01],
       [9.86157715e-01, 3.17322672e-04],
       [1.06969764e-02, 9.36964154e-01],
       [9.35326517e-01, 2.55872938e-03],
       [9.90373254e-01, 9.76463896e-04],
       [6.53627098e-01, 8.44691247e-02],
       [2.68205479e-02, 9.03193891e-01],
       [1.52896289e-04, 9.90437508e-01],
       [7.53849233e-03, 9.32610810e-01],
       [4.61395830e-05, 9.97916758e-01],
       [9.51771140e-01, 9.77563206e-03],
       [9.29120004e-01, 5.08035999e-03],
       [2.57168531e-01, 2.49635205e-01],
       [9.96572077e-01, 2.61942710e-04],
       [9.87862170e-01, 7.43782613e-04],
       [9.73691404e-01, 8.20897426e-03],
       [3.91975254e-06, 9.98498261e-01],
       [9.80087698e-01, 2.14376533e-03],
       [9.09832656e-01, 2.88372356e-02],
       [9.97190773e-01, 5.36317064e-04],
       [4.92907013e-04, 9.52523053e-01],
       [1.38734635e-02, 9.19794023e-01],
       [9.77464557e-01, 2.48204288e-03],
       [1.70654857e-05, 9.98443663e-01],
       [1.42859906e-04, 9.85853493e-01],
       [7.73116529e-01, 2.43944768e-02],
       [9.49195504e-01, 6.48476183e-04],
       [9.80180800e-01, 3.17703205e-04],
       [4.84219491e-02, 8.77735734e-01],
       [6.30656132e-08, 9.99676943e-01],
       [2.06084081e-07, 9.99889791e-01],
       [8.47639740e-01, 8.97210930e-03],
       [9.94968712e-01, 1.84790217e-04],
       [9.98517811e-01, 9.22357067e-05],
       [9.90152776e-01, 4.10761219e-04],
       [9.17411208e-01, 1.63754885e-05],
       [5.05371332e-01, 5.50724491e-02],
       [8.09481753e-06, 9.96149719e-01],
       [5.73819443e-06, 9.99302149e-01],
       [3.40470187e-02, 4.52843338e-01],
       [6.62702776e-04, 9.85725522e-01]], dtype=float32)
```

In [ ]:

Prediction Using random Data

In [74]:
```
from sklearn.preprocessing import StandardScaler
```

In [75]:
```
SCALER = StandardScaler()
```

In [82]:
```
Random_data = (0,21.6,74.72,427.9,0.08637,0.04966,0.01657,0.01115,0.1495,0.0588

Random_data_N = np.asarray(Random_data)

Random_data_N = Random_data_N.reshape(1,-1)

Random_data_NS = SCALER.fit_transform(Random_data_N)
```

In [83]:
```
MODEL.predict(Random_data_NS)
```

```
1/1 [==============================] - 0s 23ms/step
```

Out[83]:
```
array([[0.5062147 , 0.42554605]], dtype=float32)
```

In [86]:
```
Label = [np.argmax(MODEL.predict(Random_data_NS))]
Label
```

```
1/1 [==============================] - 0s 22ms/step
```

Out[86]:
```
[0]
```

In [87]:
```
if(Label[0] == 0):
  print('The tumor is Malignant')

else:
  print('The tumor is Benign')
```

```
The tumor is Malignant
```

Primary Tumor Dataset

In [113…
```
Col_namesT = ["Class", "Age", "Sex", "Histological_type", "Degree_of_diff", "Bo
              "Peritoneium", "Liver", "Brain", "Skin", "Neck", "Supraclavicular
             ]
```

In [114…
```
TumorData = pd.read_csv("primary-tumorD.csv", names= Col_namesT)
```

In [ ]:

In [116…
```
TumorData["Age"].unique()
```

Out[116]:
```
array([1, 2, 3])
```

In [120…
```
TumorData.head()
```

Out[120]:

	Class	Age	Sex	Histological_type	Degree_of_diff	Bone	Bone-marow	lung	Pluera	Peritoneiu
0	1	1	1	?	3	2	2	1	2	
1	1	1	1	?	3	2	2	2	2	
2	1	1	2	2	3	1	2	2	2	
3	1	1	2	?	3	1	2	1	1	
4	1	1	2	?	3	1	2	1	1	

In [121…

```
TumorData.tail()
```

Out[121]:

	Class	Age	Sex	Histological_type	Degree_of_diff	Bone	Bone-marow	lung	Pluera	Periton
334	22	2	2	2	?	2	2	2	2	
335	22	2	2	2	?	2	2	2	2	
336	22	2	2	?	?	1	2	2	2	
337	22	3	2	2	2	2	2	2	2	
338	22	3	2	2	2	2	2	2	2	

In [122…

```
TumorData.shape
```

Out[122]:

```
(339, 18)
```

In [124…

```
TumorData.info
```

Out[124]:
```
<bound method DataFrame.info of      Class  Age Sex Histological_type Degree_
of_diff  Bone  Bone-marow  lung  \
0         1    1   1                 ?            3      2            2     1
1         1    1   1                 ?            3      2            2     2
2         1    1   2                 2            3      1            2     2
3         1    1   2                 ?            3      1            2     1
4         1    1   2                 ?            3      1            2     1
..      ...  ...  ..               ...          ...    ...          ...   ...
334      22    2   2                 2            ?      2            2     2
335      22    2   2                 2            ?      2            2     2
336      22    2   2                 ?            ?      1            2     2
337      22    3   2                 2            2      2            2     2
338      22    3   2                 2            2      2            2     2

     Pluera  Peritoneium  Liver  Brain Skin  Neck  Supraclavicular axillar  \
0         2            2      2      2    2     2                2       2
1         2            2      1      2    2     2                1       2
2         2            2      2      2    2     2                2       2
3         1            2      2      2    2     2                2       2
4         1            2      2      2    2     2                2       2
..      ...          ...    ...    ...  ...   ...              ...     ...
334       2            2      2      2    2     2                2       1
335       2            2      2      2    2     2                2       1
336       2            2      2      2    2     2                1       1
337       2            2      2      2    2     1                1       1
338       2            2      2      2    2     2                1       1

     Mediastinum  Abdominal
0              2          2
1              1          2
2              1          2
3              1          2
4              1          2
..           ...        ...
334            2          2
335            2          2
336            2          2
337            2          2
338            2          2

[339 rows x 18 columns]>
```

In [126…
```python
TumorData.describe()
```

Out[126]:

	Class	Age	Bone	Bone-marow	lung	Pluera	Peritoneiu
count	339.000000	339.000000	339.000000	339.000000	339.000000	339.000000	339.00000
mean	8.678466	2.247788	1.722714	1.979351	1.778761	1.778761	1.71976
std	7.052624	0.568362	0.448321	0.142416	0.415695	0.415695	0.4497
min	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.00000
25%	2.000000	2.000000	1.000000	2.000000	2.000000	2.000000	1.00000
50%	7.000000	2.000000	2.000000	2.000000	2.000000	2.000000	2.00000
75%	14.000000	3.000000	2.000000	2.000000	2.000000	2.000000	2.00000
max	22.000000	3.000000	2.000000	2.000000	2.000000	2.000000	2.00000

Checking for Null values, according to dataset description here null value indicated by "?"

In [137…

```
TumorData.isnull().sum()
```

Out[137]:

```
Class                 0
Age                   0
Sex                   0
Histological_type     0
Degree_of_diff        0
Bone                  0
Bone-marow            0
lung                  0
Pluera                0
Peritoneium           0
Liver                 0
Brain                 0
Skin                  0
Neck                  0
Supraclavicular       0
axillar               0
Mediastinum           0
Abdominal             0
dtype: int64
```

In [140…

```
TumorDataR = TumorData.replace("?", "0")
TumorDataR.head()
```

Out[140]:

	Class	Age	Sex	Histological_type	Degree_of_diff	Bone	Bone-marow	lung	Pluera	Peritoneiu
0	1	1	1	0	3	2	2	1	2	
1	1	1	1	0	3	2	2	2	2	
2	1	1	2	2	3	1	2	2	2	
3	1	1	2	0	3	1	2	1	1	
4	1	1	2	0	3	1	2	1	1	

In [141…

```
TumorDataR["Class"].value_counts()
```

Out[141]:

```
1     84
5     39
18    29
11    28
14    24
22    24
2     20
12    16
7     14
4     14
17    10
3      9
13     7
8      6
19     6
10     2
15     2
20     2
6      1
16     1
21     1
Name: Class, dtype: int64
```

In [ ]: