```python
#import pandas
import pandas as pd
#import numpy
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sb
import tensorflow as tf
import keras

df = pd.read_csv("/content/breast-cancer.csv")
print(df)
```

```
           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  concavity_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
..               ...               ...             ...
```

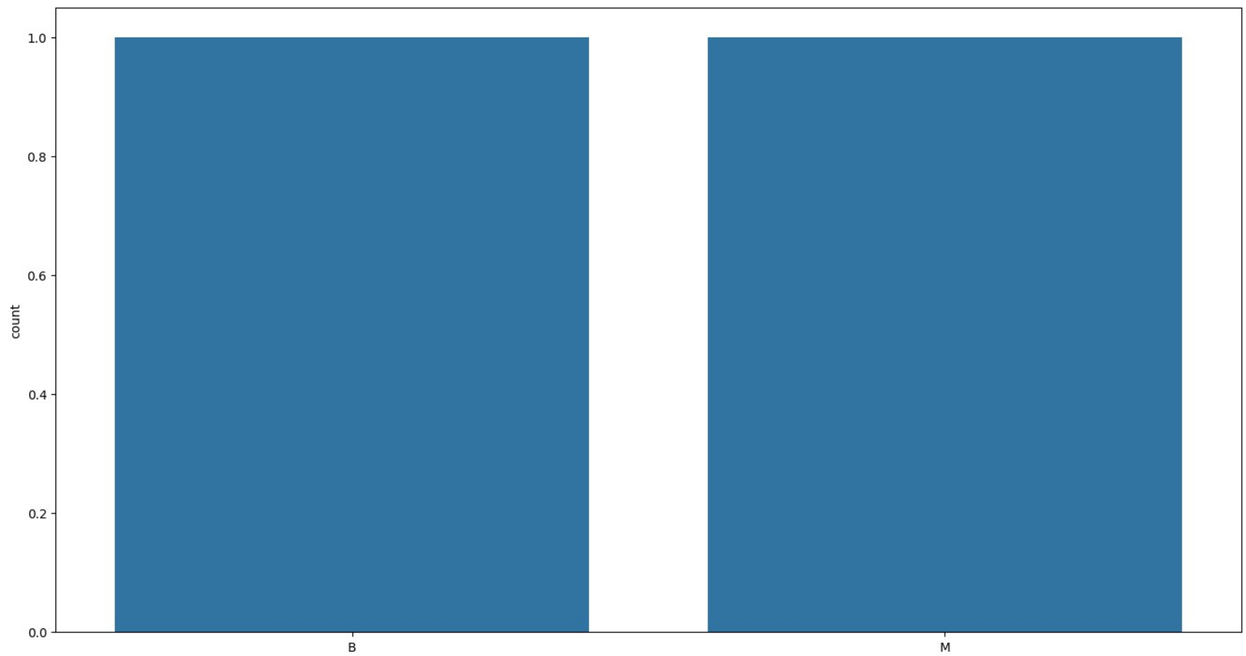
```
...
564          0.11100           0.11590         0.24390
0.13890
565          0.09780           0.10340         0.14400
0.09791
566          0.08455           0.10230         0.09251
0.05302
567          0.11780           0.27700         0.35140
0.15200
568          0.05263           0.04362         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  concavity_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]
```

```
# counting values of variables in 'diagnosis'
df['diagnosis'].value_counts()
```

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

```
plt.figure(figsize=[17,9])
sb.countplot(df['diagnosis'].value_counts())
plt.show()
```




```
df.isnull().sum()
```

```
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
dtype: int64
```

```
# independent variables
x = df.drop('diagnosis',axis=1)
#dependent variables
y = df.diagnosis
```

```
from sklearn.preprocessing import LabelEncoder
#creating the object
lb = LabelEncoder()
y = lb.fit_transform(y)
```

```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest =
train_test_split(x,y,test_size=0.3,random_state=40)
```

```
#importing StandardScaler
from sklearn.preprocessing import StandardScaler
#creating object
sc = StandardScaler()
xtrain = sc.fit_transform(xtrain)
xtest = sc.transform(xtest)
```

```
#importing keras
import keras
#importing sequential module
from keras.models import Sequential
# import dense module for hidden layers
from keras.layers import Dense
#importing activation functions
from keras.layers import LeakyReLU,PReLU,ELU
from keras.layers import Dropout
```

```
#creating model
classifier = Sequential()

#first hidden layer
classifier.add(Dense(units=9,kernel_initializer='he_uniform',activatio
n='relu',input_dim=31))
#second hidden layer
classifier.add(Dense(units=9,kernel_initializer='he_uniform',activatio
n='relu'))
# last layer or output layer
classifier.add(Dense(units=1,kernel_initializer='glorot_uniform',activ
ation='sigmoid'))

#taking summary of layers
classifier.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 9)                 288

 dense_13 (Dense)            (None, 9)                 90

 dense_14 (Dense)            (None, 1)                 10

=================================================================
Total params: 388 (1.52 KB)
Trainable params: 388 (1.52 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```


```
#taking summary of layers
classifier.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_12 (Dense)            (None, 9)                 288

 dense_13 (Dense)            (None, 9)                 90

 dense_14 (Dense)            (None, 1)                 10

=================================================================
Total params: 388 (1.52 KB)
Trainable params: 388 (1.52 KB)
Non-trainable params: 0 (0.00 Byte)
_________________________________________________________________
```

```
#compiling the ANN
classifier.compile(optimizer='adam',loss='binary_crossentropy',metrics
=['accuracy'])

 #fitting the ANN to the training set
model = classifier.fit(xtrain,ytrain,batch_size=100,epochs=100)
```

```
Epoch 1/100
4/4 [==============================] - 1s 5ms/step - loss: 1.2490 -
accuracy: 0.2362
Epoch 2/100
4/4 [==============================] - 0s 5ms/step - loss: 1.1396 -
accuracy: 0.2764
Epoch 3/100
4/4 [==============================] - 0s 5ms/step - loss: 1.0440 -
accuracy: 0.3367
Epoch 4/100
4/4 [==============================] - 0s 5ms/step - loss: 0.9555 -
accuracy: 0.3769
Epoch 5/100
4/4 [==============================] - 0s 5ms/step - loss: 0.8801 -
accuracy: 0.4397
Epoch 6/100
4/4 [==============================] - 0s 5ms/step - loss: 0.8140 -
accuracy: 0.4925
Epoch 7/100
4/4 [==============================] - 0s 7ms/step - loss: 0.7551 -
accuracy: 0.5503
Epoch 8/100
4/4 [==============================] - 0s 6ms/step - loss: 0.7056 -
accuracy: 0.6080
Epoch 9/100
4/4 [==============================] - 0s 5ms/step - loss: 0.6591 -
accuracy: 0.6482
Epoch 10/100
4/4 [==============================] - 0s 6ms/step - loss: 0.6199 -
accuracy: 0.6884
Epoch 11/100
4/4 [==============================] - 0s 6ms/step - loss: 0.5860 -
accuracy: 0.7111
Epoch 12/100
4/4 [==============================] - 0s 5ms/step - loss: 0.5553 -
accuracy: 0.7387
Epoch 13/100
4/4 [==============================] - 0s 6ms/step - loss: 0.5283 -
accuracy: 0.7688
Epoch 14/100
4/4 [==============================] - 0s 5ms/step - loss: 0.5045 -
accuracy: 0.7764
Epoch 15/100
```

```
4/4 [==============================] - 0s 5ms/step - loss: 0.4843 -
accuracy: 0.7889
Epoch 16/100
4/4 [==============================] - 0s 6ms/step - loss: 0.4658 -
accuracy: 0.8040
Epoch 17/100
4/4 [==============================] - 0s 5ms/step - loss: 0.4494 -
accuracy: 0.8090
Epoch 18/100
4/4 [==============================] - 0s 6ms/step - loss: 0.4343 -
accuracy: 0.8241
Epoch 19/100
4/4 [==============================] - 0s 5ms/step - loss: 0.4200 -
accuracy: 0.8317
Epoch 20/100
4/4 [==============================] - 0s 6ms/step - loss: 0.4068 -
accuracy: 0.8417
Epoch 21/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3953 -
accuracy: 0.8518
Epoch 22/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3839 -
accuracy: 0.8568
Epoch 23/100
4/4 [==============================] - 0s 6ms/step - loss: 0.3737 -
accuracy: 0.8618
Epoch 24/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3633 -
accuracy: 0.8668
Epoch 25/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3545 -
accuracy: 0.8744
Epoch 26/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3452 -
accuracy: 0.8794
Epoch 27/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3371 -
accuracy: 0.8869
Epoch 28/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3289 -
accuracy: 0.8869
Epoch 29/100
4/4 [==============================] - 0s 4ms/step - loss: 0.3211 -
accuracy: 0.8894
Epoch 30/100
4/4 [==============================] - 0s 5ms/step - loss: 0.3136 -
accuracy: 0.8920
Epoch 31/100
4/4 [==============================] - 0s 6ms/step - loss: 0.3063 -
```

```
accuracy: 0.8945
Epoch 32/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2994 -
accuracy: 0.8995
Epoch 33/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2924 -
accuracy: 0.8995
Epoch 34/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2858 -
accuracy: 0.9020
Epoch 35/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2796 -
accuracy: 0.9020
Epoch 36/100
4/4 [==============================] - 0s 6ms/step - loss: 0.2736 -
accuracy: 0.9070
Epoch 37/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2677 -
accuracy: 0.9070
Epoch 38/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2618 -
accuracy: 0.9095
Epoch 39/100
4/4 [==============================] - 0s 6ms/step - loss: 0.2566 -
accuracy: 0.9146
Epoch 40/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2507 -
accuracy: 0.9171
Epoch 41/100
4/4 [==============================] - 0s 6ms/step - loss: 0.2454 -
accuracy: 0.9171
Epoch 42/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2405 -
accuracy: 0.9171
Epoch 43/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2354 -
accuracy: 0.9171
Epoch 44/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2308 -
accuracy: 0.9171
Epoch 45/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2262 -
accuracy: 0.9171
Epoch 46/100
4/4 [==============================] - 0s 6ms/step - loss: 0.2219 -
accuracy: 0.9196
Epoch 47/100
4/4 [==============================] - 0s 6ms/step - loss: 0.2177 -
accuracy: 0.9221
```

```
Epoch 48/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2136 -
accuracy: 0.9246
Epoch 49/100
4/4 [==============================] - 0s 4ms/step - loss: 0.2096 -
accuracy: 0.9246
Epoch 50/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2057 -
accuracy: 0.9271
Epoch 51/100
4/4 [==============================] - 0s 5ms/step - loss: 0.2019 -
accuracy: 0.9296
Epoch 52/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1981 -
accuracy: 0.9296
Epoch 53/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1945 -
accuracy: 0.9296
Epoch 54/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1912 -
accuracy: 0.9296
Epoch 55/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1876 -
accuracy: 0.9347
Epoch 56/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1844 -
accuracy: 0.9372
Epoch 57/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1811 -
accuracy: 0.9397
Epoch 58/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1780 -
accuracy: 0.9397
Epoch 59/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1751 -
accuracy: 0.9397
Epoch 60/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1721 -
accuracy: 0.9397
Epoch 61/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1692 -
accuracy: 0.9397
Epoch 62/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1665 -
accuracy: 0.9397
Epoch 63/100
4/4 [==============================] - 0s 10ms/step - loss: 0.1640 -
accuracy: 0.9397
Epoch 64/100
```

```
4/4 [==============================] - 0s 9ms/step - loss: 0.1612 -
accuracy: 0.9422
Epoch 65/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1588 -
accuracy: 0.9447
Epoch 66/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1564 -
accuracy: 0.9497
Epoch 67/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1540 -
accuracy: 0.9497
Epoch 68/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1517 -
accuracy: 0.9548
Epoch 69/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1496 -
accuracy: 0.9548
Epoch 70/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1474 -
accuracy: 0.9548
Epoch 71/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1452 -
accuracy: 0.9548
Epoch 72/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1432 -
accuracy: 0.9548
Epoch 73/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1412 -
accuracy: 0.9548
Epoch 74/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1393 -
accuracy: 0.9573
Epoch 75/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1374 -
accuracy: 0.9573
Epoch 76/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1355 -
accuracy: 0.9573
Epoch 77/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1338 -
accuracy: 0.9573
Epoch 78/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1319 -
accuracy: 0.9598
Epoch 79/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1302 -
accuracy: 0.9623
Epoch 80/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1286 -
```

```
accuracy: 0.9673
Epoch 81/100
4/4 [==============================] - 0s 8ms/step - loss: 0.1269 -
accuracy: 0.9673
Epoch 82/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1254 -
accuracy: 0.9698
Epoch 83/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1238 -
accuracy: 0.9673
Epoch 84/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1224 -
accuracy: 0.9673
Epoch 85/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1208 -
accuracy: 0.9673
Epoch 86/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1194 -
accuracy: 0.9673
Epoch 87/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1180 -
accuracy: 0.9673
Epoch 88/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1165 -
accuracy: 0.9673
Epoch 89/100
4/4 [==============================] - 0s 8ms/step - loss: 0.1152 -
accuracy: 0.9673
Epoch 90/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1139 -
accuracy: 0.9673
Epoch 91/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1127 -
accuracy: 0.9673
Epoch 92/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1115 -
accuracy: 0.9673
Epoch 93/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1102 -
accuracy: 0.9673
Epoch 94/100
4/4 [==============================] - 0s 7ms/step - loss: 0.1089 -
accuracy: 0.9673
Epoch 95/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1076 -
accuracy: 0.9724
Epoch 96/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1065 -
accuracy: 0.9724
```

```
Epoch 97/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1051 -
accuracy: 0.9724
Epoch 98/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1039 -
accuracy: 0.9724
Epoch 99/100
4/4 [==============================] - 0s 5ms/step - loss: 0.1028 -
accuracy: 0.9724
Epoch 100/100
4/4 [==============================] - 0s 6ms/step - loss: 0.1016 -
accuracy: 0.9724
```

```
#now testing for Test data
y_pred = classifier.predict(xtest)
```

```
6/6 [==============================] - 0s 4ms/step
```

```
#converting values
y_pred = (y_pred>0.5)
print(y_pred)
```

```
[[False]
 [ True]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [ True]
 [ True]
 [ True]
 [ True]
 [ True]
```

```
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [False]
 [ True]
 [ True]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [ True]
 [False]
 [False]
 [ True]
 [ True]
 [ True]
 [False]
 [ True]
 [ True]
 [ True]
```

```
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [ True]
 [False]
 [ True]
 [ True]
 [False]
 [ True]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
```

```
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [ True]
 [False]
 [False]
 [ True]
 [False]
 [ True]
 [False]
 [ True]
 [False]
 [ True]
 [ True]
 [False]
 [ True]
 [False]
 [ True]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [False]
 [ True]
 [ True]
 [ True]]
```

```
from sklearn.metrics import confusion_matrix
from sklearn.metrics import accuracy_score
cm = confusion_matrix(ytest,y_pred)
score = accuracy_score(ytest,y_pred)
```

```python
print(cm)
print('score is:',score)
```

```
[[109   6]
 [  1  55]]
score is: 0.9590643274853801
```