```
# data analysis
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
import warnings
warnings.filterwarnings("ignore")
```

```
#read data
df=pd.read_csv("breast-cancer.csv")
df
```

| | id | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | sm |
|---|---|---|---|---|---|---|---|
| **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 | |

569 rows × 32 columns

```
# data information
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 32 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
dtypes: float64(30), int64(1), object(1)
memory usage: 142.4+ KB
```

```
# data contain 569 entries float64(30), int64(1), object(1)
```

```
# checking null values
# checking null values
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
```

```
df.drop('id',1,inplace=True)
```

```
df['diagnosis'].value_counts()
```

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

```
#handling object columns
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
df['diagnosis']=le.fit_transform(df['diagnosis'])
```

```
df.head()
```

| | diagnosis | radius_mean | texture_mean | perimeter_mean | area_mean | smoothness_mea |
|---|---|---|---|---|---|---|
| **0** | 1 | 17.99 | 10.38 | 122.80 | 1001.0 | 0.1184 |
| **1** | 1 | 20.57 | 17.77 | 132.90 | 1326.0 | 0.0847 |
| **2** | 1 | 19.69 | 21.25 | 130.00 | 1203.0 | 0.1096 |
| **3** | 1 | 11.42 | 20.38 | 77.58 | 386.1 | 0.1425 |
| **4** | 1 | 20.29 | 14.34 | 135.10 | 1297.0 | 0.1003 |

```
# spliting data into train_test_split
x=df.drop('diagnosis',1).values
x
```

```
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]])
```

```
y=df['diagnosis'].values
y
```

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

```
from sklearn.model_selection import train_test_split
x_train,x_test,y_train,y_test=train_test_split(x,y,test_size=0.3,random_state=15)
```

```
# scaling
from sklearn.preprocessing import StandardScaler
sc=StandardScaler()
x_train=sc.fit_transform(x_train)
x_test=sc.transform(x_test)
```

```
# creating a model
import tensorflow as tf
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report
```

```
# initialize the model
ann=Sequential()
```

```
# add layers to the data
ann.add(Dense(units=6,activation='relu'))
# output layer
ann.add(Dense(units=1,activation="sigmoid"))
#establish a connection between layers
ann.compile(optimizer="adam",loss="binary_crossentropy",metrics=["accuracy"])
```

```
# train the model
ann.fit(x_train,y_train,batch_size=30,epochs=100)
ypred=ann.predict(x_test)
```

```
Epoch 1/100
14/14 [==============================] - 1s 3ms/step - loss: 1.1710 - accuracy: 0.1834
Epoch 2/100
14/14 [==============================] - 0s 3ms/step - loss: 0.9887 - accuracy: 0.2462
Epoch 3/100
14/14 [==============================] - 0s 2ms/step - loss: 0.8604 - accuracy: 0.3618
Epoch 4/100
14/14 [==============================] - 0s 2ms/step - loss: 0.7631 - accuracy: 0.4322
Epoch 5/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6940 - accuracy: 0.5025
Epoch 6/100
14/14 [==============================] - 0s 2ms/step - loss: 0.6397 - accuracy: 0.5779
Epoch 7/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5955 - accuracy: 0.6683
Epoch 8/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5587 - accuracy: 0.7437
Epoch 9/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5282 - accuracy: 0.8141
Epoch 10/100
14/14 [==============================] - 0s 2ms/step - loss: 0.5011 - accuracy: 0.8618
Epoch 11/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4746 - accuracy: 0.8794
```

```
Epoch 12/100
14/14 [==============================] - 0s 2ms/step - loss: 0.4486 - accuracy: 0.8894
Epoch 13/100
14/14 [==============================] - 0s 3ms/step - loss: 0.4238 - accuracy: 0.9070
Epoch 14/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3979 - accuracy: 0.9171
Epoch 15/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3723 - accuracy: 0.9246
Epoch 16/100
14/14 [==============================] - 0s 3ms/step - loss: 0.3472 - accuracy: 0.9347
Epoch 17/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3228 - accuracy: 0.9422
Epoch 18/100
14/14 [==============================] - 0s 2ms/step - loss: 0.3002 - accuracy: 0.9472
Epoch 19/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2794 - accuracy: 0.9497
Epoch 20/100
14/14 [==============================] - 0s 3ms/step - loss: 0.2612 - accuracy: 0.9497
Epoch 21/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2441 - accuracy: 0.9523
Epoch 22/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2292 - accuracy: 0.9523
Epoch 23/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2158 - accuracy: 0.9523
Epoch 24/100
14/14 [==============================] - 0s 2ms/step - loss: 0.2038 - accuracy: 0.9523
Epoch 25/100
14/14 [==============================] - 0s 3ms/step - loss: 0.1921 - accuracy: 0.9548
Epoch 26/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1821 - accuracy: 0.9548
Epoch 27/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1727 - accuracy: 0.9548
Epoch 28/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1639 - accuracy: 0.9573
Epoch 29/100
14/14 [==============================] - 0s 2ms/step - loss: 0.1563 - accuracy: 0.9573
```

```
ypred=ypred>0.5
```

```
ypred=np.where(ypred<0.5,0,1)
ypred
```

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

```
print(classification_report(ypred,y_test))
```

```
              precision    recall  f1-score   support

           0       0.97      0.97      0.97       108
           1       0.95      0.95      0.95        63

    accuracy                           0.96       171
   macro avg       0.96      0.96      0.96       171
weighted avg       0.96      0.96      0.96       171
```

```
# model predicted 96% accuracy
```

```
## Early stopping concept
```

```
from tensorflow.keras.callbacks import EarlyStopping
early_stop=EarlyStopping(monitor='val_loss',mode='min',verbose=1,patience=13)
```

```
#initialize the model
model=Sequential()
```

```
# add layers in data
model.add(Dense(20,activation='relu'))
model.add(Dense(20,activation='relu'))
# output layer
model.add(Dense(1,activation='sigmoid'))
```

```
# establish connection between layers
model.compile(optimizer='sgd',loss='binary_crossentropy',metrics=['accuracy'])
```

```
# train the model
model.fit(x_train,y_train,epochs=600,validation_data=(x_test,y_test),verbose=1,batch_size=124,callbacks=[early_stop])
```

```
Epoch 316/600
4/4 [==============================] - 0s 22ms/step - loss: 0.0539 - accuracy: 0.9849 - val_loss: 0.1248 - val_accuracy: 0.9474
Epoch 317/600
4/4 [==============================] - 0s 29ms/step - loss: 0.0538 - accuracy: 0.9824 - val_loss: 0.1248 - val_accuracy: 0.9474
Epoch 318/600
4/4 [==============================] - 0s 28ms/step - loss: 0.0537 - accuracy: 0.9849 - val_loss: 0.1248 - val_accuracy: 0.9474
Epoch 319/600
4/4 [==============================] - 0s 30ms/step - loss: 0.0536 - accuracy: 0.9849 - val_loss: 0.1248 - val_accuracy: 0.9474
Epoch 320/600
4/4 [==============================] - 0s 37ms/step - loss: 0.0535 - accuracy: 0.9849 - val_loss: 0.1247 - val_accuracy: 0.9474
Epoch 321/600
4/4 [==============================] - 0s 22ms/step - loss: 0.0534 - accuracy: 0.9849 - val_loss: 0.1247 - val_accuracy: 0.9474
Epoch 322/600
4/4 [==============================] - 0s 27ms/step - loss: 0.0533 - accuracy: 0.9849 - val_loss: 0.1246 - val_accuracy: 0.9474
Epoch 323/600
4/4 [==============================] - 0s 28ms/step - loss: 0.0532 - accuracy: 0.9849 - val_loss: 0.1246 - val_accuracy: 0.9474
Epoch 324/600
4/4 [==============================] - 0s 32ms/step - loss: 0.0531 - accuracy: 0.9849 - val_loss: 0.1245 - val_accuracy: 0.9474
Epoch 325/600
4/4 [==============================] - 0s 22ms/step - loss: 0.0530 - accuracy: 0.9849 - val_loss: 0.1245 - val_accuracy: 0.9474
Epoch 325: early stopping
<keras.callbacks.History at 0x7fac4b0f09a0>
```

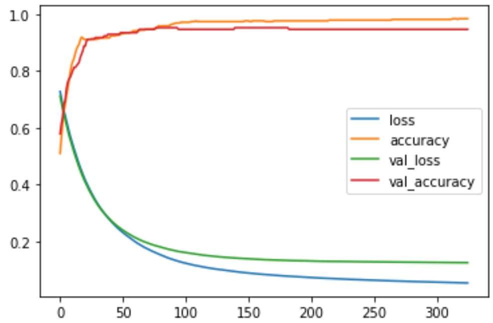
```
# in this dataset early stopping concept stopped at 325
```

```
model.history.history
```

```
  0.2881026864051819,
  0.282707065343856,
  0.27772170305252075,
  0.2726327180862427,
  0.2676842510700226,
  0.2629072070121765,
  0.2584548890590668,
  0.2540280818939209,
  0.2497902810573578,
  0.24557721614837646,
  0.24174371361732483,
  0.23778511583805084,
  0.23396353423595428,
  0.2303638905286789,
  0.2266959398984909,
  0.22337095439434052,
  0.21996936202049255,
  0.21664325892925262,
  0.21333716809749603,
  0.21010537445545197,
  0.20647157728672028,
  0.2032080590724945,
  0.20021799206733704,
  0.197351129617691,
  0.19439764320850372,
  0.191611185669899,
  0.18903546035289764,
  0.1863665133714676,
  0.1837153285741806,
  0.1812102198600769,
  0.17861822247505188,
  0.17633208632469177,
  0.1740250587463379,
  0.17196054756641388,
  0.16950520873069763,
  0.1673615425825119,
  0.16539570689201355,
  0.16339091956615448,
  0.1612885445356369,
  0.15942221879959106,
  0.15758073329925537,
  0.1555544137954712,
  0.15368545055389404,
  0.15183527767658234,
  0.14995335042476654,
  0.14782847464084625,
  0.14609244465827942,
  0.14431695640087128,
  0.14262010157108307,
  0.14098292589187622,
  0.13946032524108887,
  0.13796046376228333,
  0.13653117418289185,
  0.1351422220468521,
  0.13357658684253693,
  0.13221560418605804,
  0.1307910978794098,
  0.12942788004875183,
  0.1281657963991165?
```

```
lossdf=pd.DataFrame(model.history.history)
lossdf.plot()
```

```
<AxesSubplot:>
```



```
ypred=model.predict(x_test)
ypred=ypred>0.5
```

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

```
from sklearn.metrics import classification_report
print(classification_report(y_test,ypred))
```

```
              precision    recall  f1-score   support

           0       0.95      0.96      0.96       108
           1       0.94      0.92      0.93        63

    accuracy                           0.95       171
   macro avg       0.94      0.94      0.94       171
weighted avg       0.95      0.95      0.95       171
```

```
# model predicted 95% accuracy
```