Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt
```

Reading Data

```
df = pd.read_csv("/content/breast-cancer.csv",index_col=0)
df.head()
```

:oncavity_mean	concave points_mean	symmetry_mean	...	radius_worst	texture_worst	perimeter_worst	area_worst	smoothness_worst	compactness_worst	concavity_
0.3001	0.14710	0.2419	...	25.38	17.33	184.60	2019.0	0.1622	0.6656	0
0.0869	0.07017	0.1812	...	24.99	23.41	158.80	1956.0	0.1238	0.1866	0
0.1974	0.12790	0.2069	...	23.57	25.53	152.50	1709.0	0.1444	0.4245	0
0.2414	0.10520	0.2597	...	14.91	26.50	98.87	567.7	0.2098	0.8663	0
0.1980	0.10430	0.1809	...	22.54	16.67	152.20	1575.0	0.1374	0.2050	0

Understanding Data

```
df.diagnosis.value_counts()
```

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

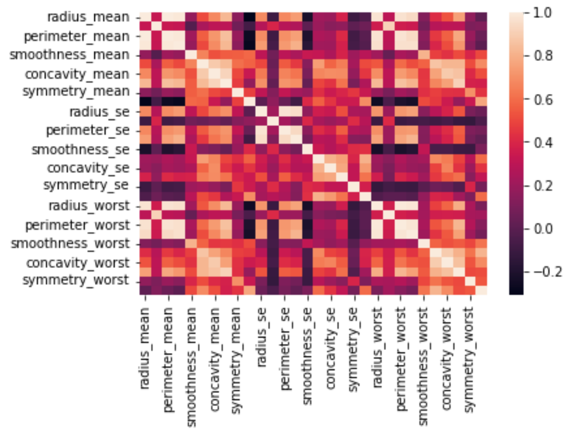
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 569 entries, 842302 to 92751
Data columns (total 31 columns):
 #   Column                   Non-Null Count  Dtype  
---  ------                   --------------  -----  
 0   diagnosis                569 non-null    object 
 1   radius_mean              569 non-null    float64
 2   texture_mean             569 non-null    float64
 3   perimeter_mean           569 non-null    float64
 4   area_mean                569 non-null    float64
 5   smoothness_mean          569 non-null    float64
 6   compactness_mean         569 non-null    float64
 7   concavity_mean           569 non-null    float64
 8   concave points_mean      569 non-null    float64
 9   symmetry_mean            569 non-null    float64
 10  fractal_dimension_mean   569 non-null    float64
 11  radius_se                569 non-null    float64
 12  texture_se               569 non-null    float64
 13  perimeter_se             569 non-null    float64
 14  area_se                  569 non-null    float64
 15  smoothness_se            569 non-null    float64
 16  compactness_se           569 non-null    float64
 17  concavity_se             569 non-null    float64
 18  concave points_se        569 non-null    float64
 19  symmetry_se              569 non-null    float64
 20  fractal_dimension_se     569 non-null    float64
 21  radius_worst             569 non-null    float64
 22  texture_worst            569 non-null    float64
 23  perimeter_worst          569 non-null    float64
 24  area_worst               569 non-null    float64
 25  smoothness_worst         569 non-null    float64
 26  compactness_worst        569 non-null    float64
 27  concavity_worst          569 non-null    float64
 28  concave points_worst     569 non-null    float64
 29  symmetry_worst           569 non-null    float64
 30  fractal_dimension_worst  569 non-null    float64
dtypes: float64(30), object(1)
memory usage: 142.2+ KB
```

Checking Correlation

```
sns.heatmap(df.corr())
```

```
<AxesSubplot:>
```



Standardization

```
from sklearn.preprocessing import StandardScaler,LabelEncoder
SS = StandardScaler()
LE = LabelEncoder()
```

```
X = df.iloc[:,1:]
X = SS.fit_transform(X)
X
```

```
array([[ 1.09706398, -2.07333501,  1.26993369, ...,  2.29607613,
         2.75062224,  1.93701461],
       [ 1.82982061, -0.35363241,  1.68595471, ...,  1.0870843 ,
        -0.24388967,  0.28118999],
       [ 1.57988811,  0.45618695,  1.56650313, ...,  1.95500035,
         1.152255  ,  0.20139121],
       ...,
       [ 0.70228425,  2.0455738 ,  0.67267578, ...,  0.41406869,
        -1.10454895, -0.31840916],
       [ 1.83834103,  2.33645719,  1.98252415, ...,  2.28998549,
         1.91908301,  2.21963528],
```

```
       [-1.80840125,  1.22179204, -1.81438851, ..., -1.74506282,
        -0.04813821, -0.75120669]])
```

```
Y = df.diagnosis
Y = LE.fit_transform(Y)
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])
```

```
LE.classes_
```

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

Splitting the Data

```
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=1)
```

```
import tensorflow as tf
from tensorflow.keras.callbacks import EarlyStopping
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
```

EarlyStopping to save resources

```
ES = EarlyStopping(monitor="val_loss",mode="min",verbose=1,patience=8)
```

Building Model

```
# step1 :- initialize the Model
ann = Sequential()

# step2 :- Add Layers into model
ann.add( Dense(units = 20, activation = "relu") )
ann.add( Dense(units = 20, activation = "relu") )

ann.add( Dense(units = 1, activation="sigmoid") )      # Output Layer

# step3 :- Establihing connection
ann.compile(optimizer='adam', loss = 'binary_crossentropy' , metrics=["accuracy"])

# step4 :- Fit the model
ann.fit(X_train, Y_train, batch_size = 15, epochs = 800,validation_data=(X_test,Y_test),callbacks=[ES])
```

```
Epoch 1/800
27/27 [==============================] - 1s 11ms/step - loss: 0.5787 - accuracy: 0.7085 - val_loss: 0.4808 - val_accuracy: 0.9006
Epoch 2/800
27/27 [==============================] - 0s 4ms/step - loss: 0.3251 - accuracy: 0.9347 - val_loss: 0.3009 - val_accuracy: 0.9298
Epoch 3/800
27/27 [==============================] - 0s 4ms/step - loss: 0.2023 - accuracy: 0.9497 - val_loss: 0.2122 - val_accuracy: 0.9474
Epoch 4/800
27/27 [==============================] - 0s 5ms/step - loss: 0.1434 - accuracy: 0.9673 - val_loss: 0.1723 - val_accuracy: 0.9532
Epoch 5/800
27/27 [==============================] - 0s 4ms/step - loss: 0.1126 - accuracy: 0.9724 - val_loss: 0.1488 - val_accuracy: 0.9649
Epoch 6/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0937 - accuracy: 0.9749 - val_loss: 0.1357 - val_accuracy: 0.9649
Epoch 7/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0807 - accuracy: 0.9824 - val_loss: 0.1274 - val_accuracy: 0.9649
Epoch 8/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0718 - accuracy: 0.9849 - val_loss: 0.1221 - val_accuracy: 0.9649
Epoch 9/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0655 - accuracy: 0.9874 - val_loss: 0.1201 - val_accuracy: 0.9649
Epoch 10/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0603 - accuracy: 0.9849 - val_loss: 0.1164 - val_accuracy: 0.9649
Epoch 11/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0563 - accuracy: 0.9849 - val_loss: 0.1159 - val_accuracy: 0.9474
Epoch 12/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0524 - accuracy: 0.9874 - val_loss: 0.1164 - val_accuracy: 0.9474
Epoch 13/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0497 - accuracy: 0.9874 - val_loss: 0.1143 - val_accuracy: 0.9474
Epoch 14/800
27/27 [==============================] - 0s 4ms/step - loss: 0.0466 - accuracy: 0.9899 - val_loss: 0.1151 - val_accuracy: 0.9474
Epoch 15/800
27/27 [==============================] - 0s 6ms/step - loss: 0.0441 - accuracy: 0.9899 - val_loss: 0.1147 - val_accuracy: 0.9474
Epoch 16/800
27/27 [==============================] - 0s 8ms/step - loss: 0.0423 - accuracy: 0.9899 - val_loss: 0.1155 - val_accuracy: 0.9474
Epoch 17/800
27/27 [==============================] - 0s 8ms/step - loss: 0.0409 - accuracy: 0.9899 - val_loss: 0.1154 - val_accuracy: 0.9474
Epoch 18/800
27/27 [==============================] - 0s 6ms/step - loss: 0.0382 - accuracy: 0.9925 - val_loss: 0.1153 - val_accuracy: 0.9474
Epoch 19/800
27/27 [==============================] - 0s 11ms/step - loss: 0.0369 - accuracy: 0.9899 - val_loss: 0.1151 - val_accuracy: 0.9474
Epoch 20/800
27/27 [==============================] - 0s 7ms/step - loss: 0.0353 - accuracy: 0.9899 - val_loss: 0.1170 - val_accuracy: 0.9474
Epoch 21/800
27/27 [==============================] - 0s 9ms/step - loss: 0.0342 - accuracy: 0.9925 - val_loss: 0.1156 - val_accuracy: 0.9474
Epoch 21: early stopping
<keras.callbacks.History at 0x7fd7b42357f0>
```

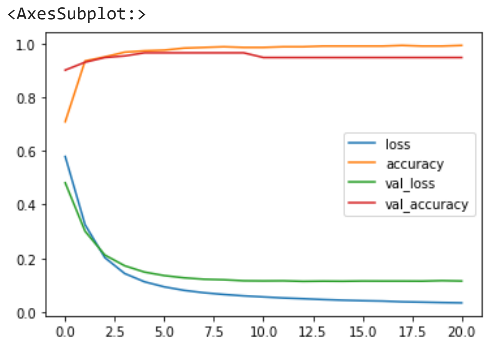
```
ann.history.history
```

```
 0.9874371886253357,
 0.9899497628211975,
 0.9899497628211975,
 0.9899497628211975,
 0.9899497628211975,
 0.9924623370170593,
 0.9899497628211975,
 0.9899497628211975,
 0.9924623370170593],
'val_loss': [0.4807799458503723,
 0.3008545935153961,
 0.21223758161067963,
 0.17229530215263367,
 0.14876030385494232,
 0.13566409051418304,
 0.12737835943698883,
 0.12212026864290237,
 0.12014926970005035,
 0.11644038558006287,
 0.1159445121884346,
 0.11635943502187729,
 0.11426496505737305,
 0.1150810644030571,
 0.11465619504451752,
 0.11554322391748428,
 0.11535472422838211,
 0.11533941328525543,
 0.11514012515544891,
 0.11695259809494019,
 0.11558444797992706],
'val_accuracy': [0.9005848169326782,
 0.9298245906829834,
 0.9473684430122375,
 0.9532163739204407,
 0.9649122953414917,
 0.9649122953414917,
 0.9649122953414917,
 0.9649122953414917,
 0.9649122953414917,
 0.9649122953414917,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375,
 0.9473684430122375]}
```

Visualising the loss

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

```
<AxesSubplot:>
```




```
# step5 :- Predict the model
Y_pred = ann.predict(X_test)
```

```
6/6 [==============================] - 0s 3ms/step
```

```
Y_pred = np.where(Y_pred>0.5,1,0)
```

Evaluation

```
from sklearn.metrics import classification_report
print(classification_report(Y_test,Y_pred))
```

```
              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
```