```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import warnings
warnings.filterwarnings('ignore')
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import OrdinalEncoder
import tensorflow as tf
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
sns.set_style('darkgrid')
```

Import the data

```
df=pd.read_csv('data.csv')
```

Examine first few rows

```
df.head()

         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

   smoothness_mean  compactness_mean  concavity_mean  concave
points_mean  \
0          0.11840           0.27760          0.3001
0.14710
1          0.08474           0.07864          0.0869
0.07017
2          0.10960           0.15990          0.1974
0.12790
3          0.14250           0.28390          0.2414
0.10520
4          0.10030           0.13280          0.1980
0.10430

   ...  texture_worst  perimeter_worst  area_worst
smoothness_worst  \
```

```
0  ...         17.33           184.60      2019.0            0.1622

1  ...         23.41           158.80      1956.0            0.1238

2  ...         25.53           152.50      1709.0            0.1444

3  ...         26.50            98.87       567.7            0.2098

4  ...         16.67           152.20      1575.0            0.1374


   compactness_worst  concavity_worst  concave points_worst
symmetry_worst  \
0             0.6656           0.7119                0.2654
0.4601
1             0.1866           0.2416                0.1860
0.2750
2             0.4245           0.4504                0.2430
0.3613
3             0.8663           0.6869                0.2575
0.6638
4             0.2050           0.4000                0.1625
0.2364

   fractal_dimension_worst  Unnamed: 32
0                  0.11890          NaN
1                  0.08902          NaN
2                  0.08758          NaN
3                  0.17300          NaN
4                  0.07678          NaN

[5 rows x 33 columns]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 569 entries, 0 to 568
Data columns (total 33 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
 32  Unnamed: 32              0 non-null      float64
dtypes: float64(31), int64(1), object(1)
memory usage: 146.8+ KB
```

Attribute Information: ID number Diagnosis (M = malignant, B = benign) Ten real-valued features are computed for each cell nucleus: 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) symmetry fractal dimension ("coastline approximation" - 1)

Total 569 entries and 33 columns. column consist of values with float, object and integer datatype.

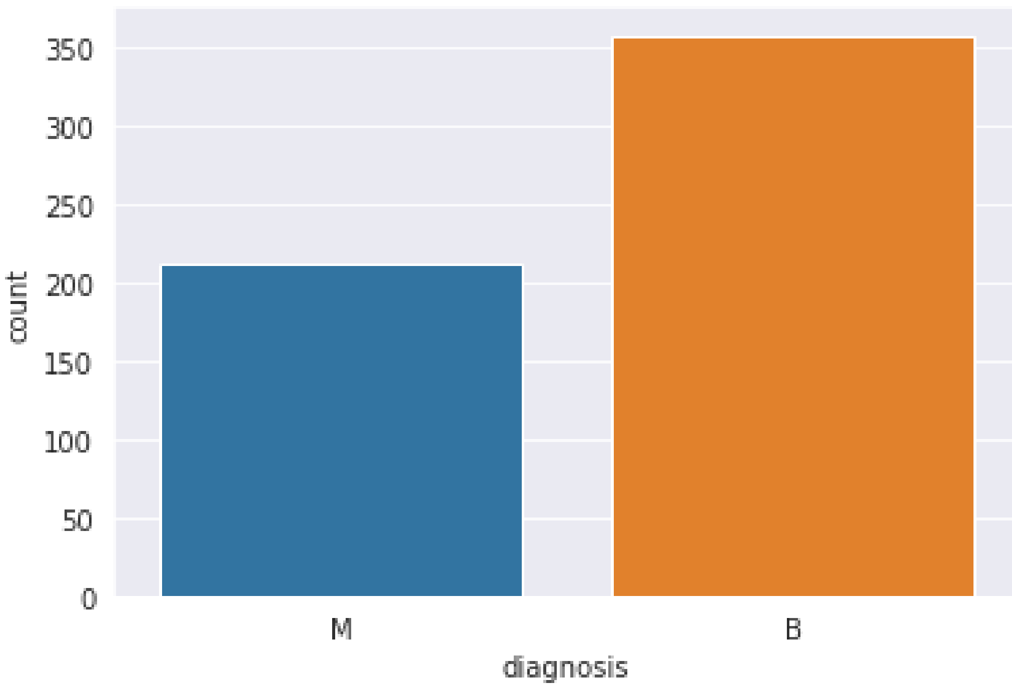
```
#Droped unamed:32 and id column
df.drop(['Unnamed: 32','id'], axis=1, inplace=True)  #Droped unamed:32 and id column
```

```
#No of rows and columns
df.shape
```

```
(569, 31)
```

```
#Target class(diagnosis) distribution
sns.countplot(df["diagnosis"])
print(df['diagnosis'].value_counts())
```

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



The dataset is mildly imbalanced

```
#Missing attribute values
df.isnull().sum()
```

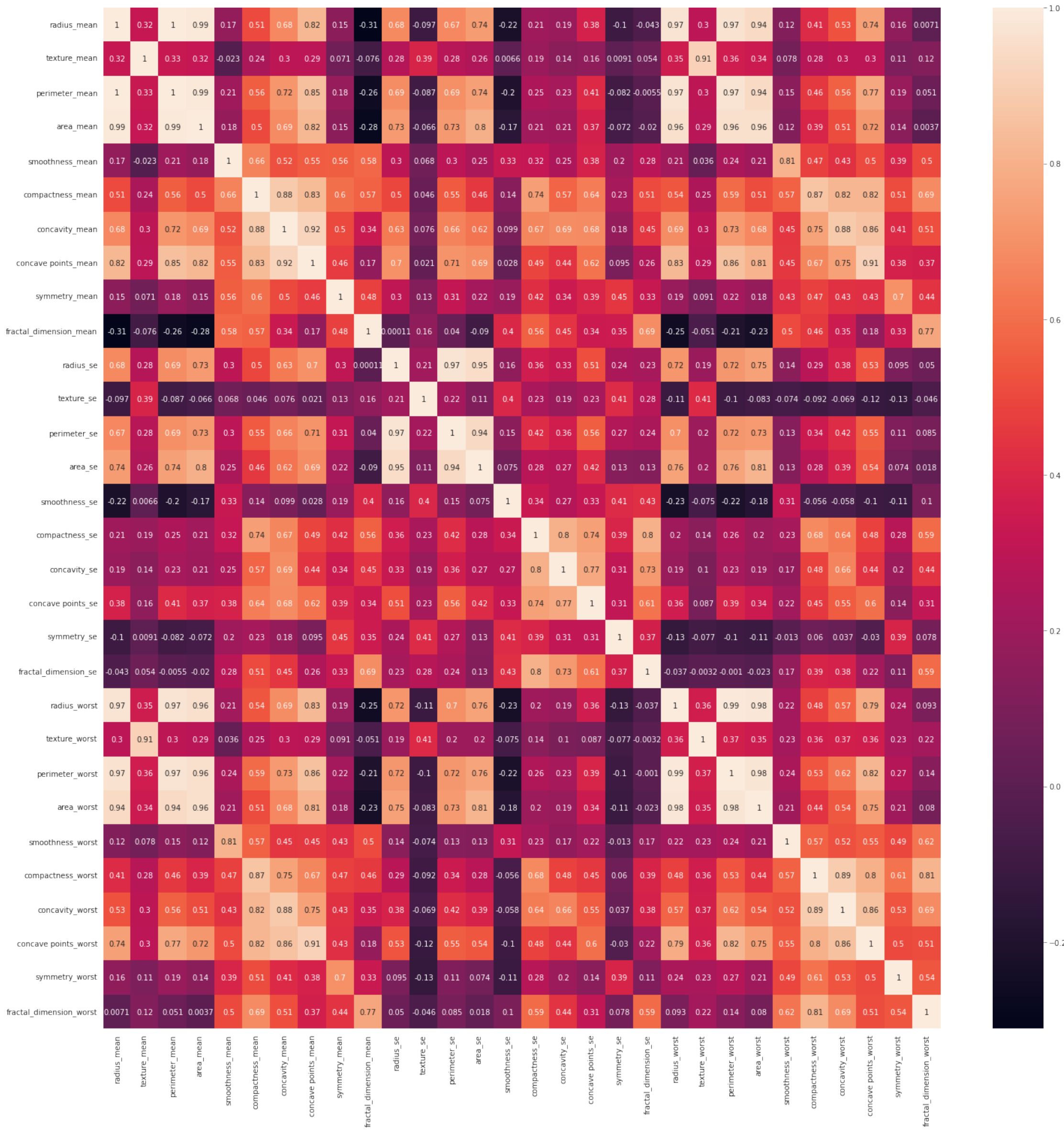
```
diagnosis                  0
radius_mean                0
texture_mean               0
perimeter_mean             0
area_mean                  0
smoothness_mean            0
compactness_mean           0
concavity_mean             0
concave points_mean        0
symmetry_mean              0
fractal_dimension_mean     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
```

No Missing values

```
#Visualising using heatmap: Helps to understand correlation between features
plt.figure(figsize=(25,25))
sns.heatmap(df.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd819fccb90>
```



```
#Replacing string values of diagnosis column into 0&1
df['diagnosis']= df['diagnosis'].apply(lambda x: 1 if x=='M' else 0) #
B with 0 and M with 1

#Most 10 correlated attributes with diagnosis
df.corr()['diagnosis'][:].sort_values()[:10]

smoothness_se             -0.067016
fractal_dimension_mean    -0.012838
texture_se                -0.008303
symmetry_se               -0.006522
fractal_dimension_se       0.077972
concavity_se               0.253730
compactness_se             0.292999
fractal_dimension_worst    0.323872
```

```
symmetry_mean              0.330499
smoothness_mean            0.358560
Name: diagnosis, dtype: float64
```

```
#Selecting only highly correlated data
df_corr = df[['smoothness_se','fractal_dimension_mean',
'texture_se','symmetry_se','fractal_dimension_se',

'concavity_se','compactness_se','fractal_dimension_worst','symmetry_me
an','smoothness_mean','diagnosis']]
```

```
#First 5 samples of df_corr
df_corr.head()
```

```
   smoothness_se  fractal_dimension_mean  texture_se  symmetry_se  \
0       0.006399                 0.07871      0.9053      0.03003
1       0.005225                 0.05667      0.7339      0.01389
2       0.006150                 0.05999      0.7869      0.02250
3       0.009110                 0.09744      1.1560      0.05963
4       0.011490                 0.05883      0.7813      0.01756

   fractal_dimension_se  concavity_se  compactness_se  \
0              0.006193       0.05373         0.04904
1              0.003532       0.01860         0.01308
2              0.004571       0.03832         0.04006
3              0.009208       0.05661         0.07458
4              0.005115       0.05688         0.02461

   fractal_dimension_worst  symmetry_mean  smoothness_mean  diagnosis

0                  0.11890         0.2419          0.11840          1

1                  0.08902         0.1812          0.08474          1

2                  0.08758         0.2069          0.10960          1

3                  0.17300         0.2597          0.14250          1

4                  0.07678         0.1809          0.10030          1
```

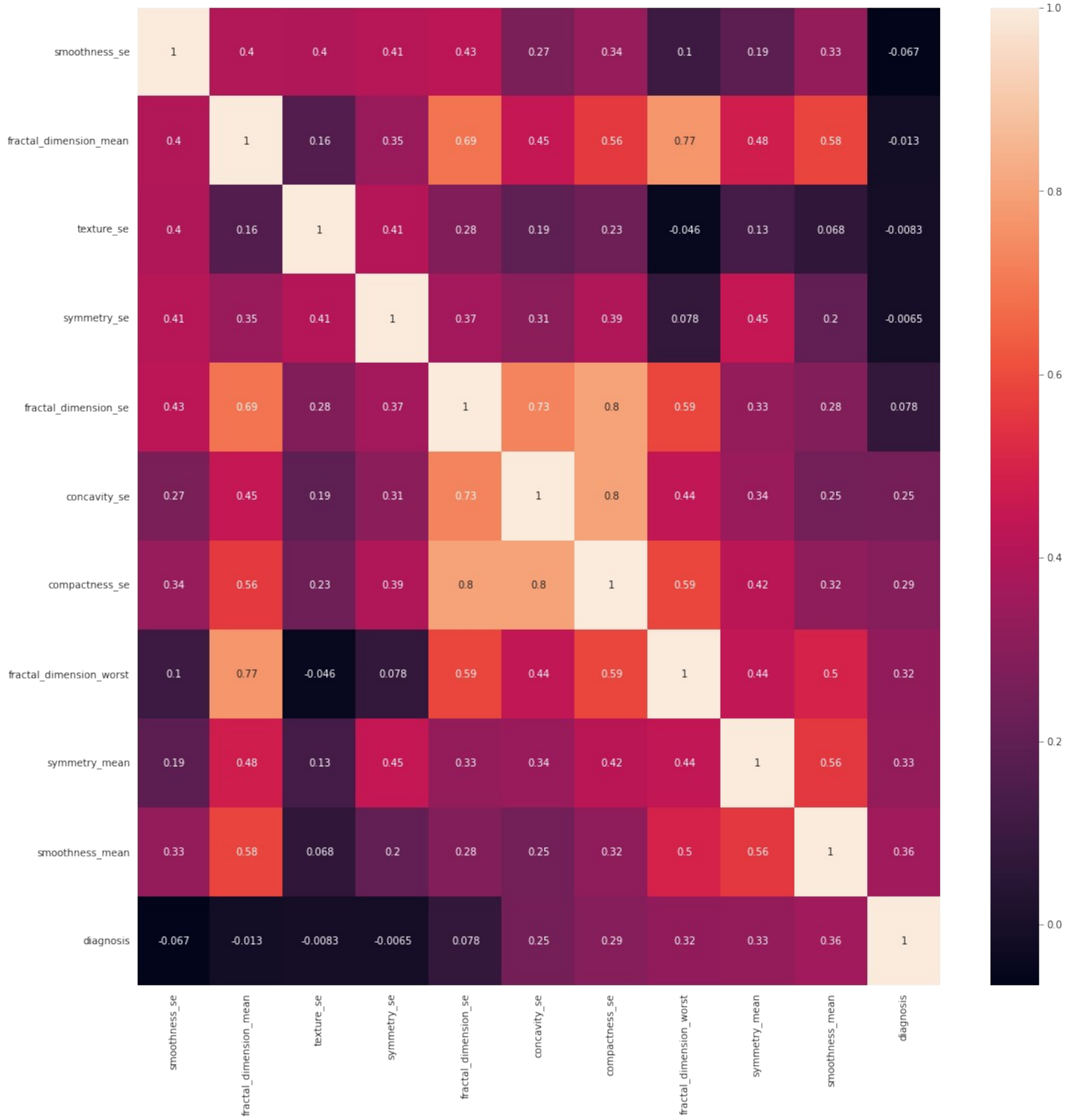
```
#Descriptibe statistics
df_corr.describe()
```

```
       smoothness_se  fractal_dimension_mean  texture_se  symmetry_se
\
count     569.000000              569.000000  569.000000   569.000000

mean        0.007041                0.062798    1.216853     0.020542

std         0.003003                0.007060    0.551648     0.008266
```

```
min         0.001713                0.049960    0.360200    0.007882

25%         0.005169                0.057700    0.833900    0.015160

50%         0.006380                0.061540    1.108000    0.018730

75%         0.008146                0.066120    1.474000    0.023480

max         0.031130                0.097440    4.885000    0.078950


       fractal_dimension_se  concavity_se  compactness_se  \
count            569.000000    569.000000      569.000000
mean               0.003795      0.031894        0.025478
std                0.002646      0.030186        0.017908
min                0.000895      0.000000        0.002252
25%                0.002248      0.015090        0.013080
50%                0.003187      0.025890        0.020450
75%                0.004558      0.042050        0.032450
max                0.029840      0.396000        0.135400

       fractal_dimension_worst  symmetry_mean  smoothness_mean
diagnosis
count               569.000000     569.000000       569.000000
569.000000
mean                  0.083946       0.181162         0.096360
0.372583
std                   0.018061       0.027414         0.014064
0.483918
min                   0.055040       0.106000         0.052630
0.000000
25%                   0.071460       0.161900         0.086370
0.000000
50%                   0.080040       0.179200         0.095870
0.000000
75%                   0.092080       0.195700         0.105300
1.000000
max                   0.207500       0.304000         0.163400
1.000000
```

```
plt.figure(figsize=(18,18))
sns.heatmap(df_corr.corr(),annot=True)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fd81a0d6810>
```




```
#Spliting X & Y for model building
x=df.drop(['diagnosis'],axis=1)
x

     radius_mean  texture_mean  perimeter_mean  area_mean
smoothness_mean  \
0          17.99         10.38          122.80     1001.0
0.11840
1          20.57         17.77          132.90     1326.0
0.08474
2          19.69         21.25          130.00     1203.0
0.10960
3          11.42         20.38           77.58      386.1
0.14250
4          20.29         14.34          135.10     1297.0
```

```
0.10030
..           ...           ...             ...        ...
...
564        21.56         22.39          142.00     1479.0
0.11100
565        20.13         28.25          131.20     1261.0
0.09780
566        16.60         28.08          108.30      858.1
0.08455
567        20.60         29.33          140.10     1265.0
0.11780
568         7.76         24.54           47.92      181.0
0.05263

     compactness_mean  concavity_mean  concave points_mean
symmetry_mean  \
0             0.27760         0.30010              0.14710
0.2419
1             0.07864         0.08690              0.07017
0.1812
2             0.15990         0.19740              0.12790
0.2069
3             0.28390         0.24140              0.10520
0.2597
4             0.13280         0.19800              0.10430
0.1809
..                ...             ...                  ...
...
564           0.11590         0.24390              0.13890
0.1726
565           0.10340         0.14400              0.09791
0.1752
566           0.10230         0.09251              0.05302
0.1590
567           0.27700         0.35140              0.15200
0.2397
568           0.04362         0.00000              0.00000
0.1587

     fractal_dimension_mean  ...  radius_worst  texture_worst  \
0                   0.07871  ...        25.380          17.33
1                   0.05667  ...        24.990          23.41
2                   0.05999  ...        23.570          25.53
3                   0.09744  ...        14.910          26.50
4                   0.05883  ...        22.540          16.67
..                      ...  ...           ...            ...
564                 0.05623  ...        25.450          26.40
565                 0.05533  ...        23.690          38.25
566                 0.05648  ...        18.980          34.12
567                 0.07016  ...        25.740          39.42
```

```
568                 0.05884  ...         9.456          30.37

     perimeter_worst  area_worst  smoothness_worst  compactness_worst
\
0             184.60      2019.0           0.16220            0.66560

1             158.80      1956.0           0.12380            0.18660

2             152.50      1709.0           0.14440            0.42450

3              98.87       567.7           0.20980            0.86630

4             152.20      1575.0           0.13740            0.20500

..               ...         ...               ...                ...

564           166.10      2027.0           0.14100            0.21130

565           155.00      1731.0           0.11660            0.19220

566           126.70      1124.0           0.11390            0.30940

567           184.60      1821.0           0.16500            0.86810

568            59.16       268.6           0.08996            0.06444


     concavity_worst  concave points_worst  symmetry_worst  \
0             0.7119                0.2654          0.4601
1             0.2416                0.1860          0.2750
2             0.4504                0.2430          0.3613
3             0.6869                0.2575          0.6638
4             0.4000                0.1625          0.2364
..               ...                   ...             ...
564           0.4107                0.2216          0.2060
565           0.3215                0.1628          0.2572
566           0.3403                0.1418          0.2218
567           0.9387                0.2650          0.4087
568           0.0000                0.0000          0.2871

     fractal_dimension_worst
0                    0.11890
1                    0.08902
2                    0.08758
3                    0.17300
4                    0.07678
..                       ...
564                  0.07115
565                  0.06637
```

```
566                   0.07820
567                   0.12400
568                   0.07039

[569 rows x 30 columns]
```

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

```
0      1
1      1
2      1
3      1
4      1
      ..
564    1
565    1
566    1
567    1
568    0
Name: diagnosis, Length: 569, dtype: int64
```

```
#Spliting data using inbuilt function train_test_split to get  traing
and testing values of x & y
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.3,random_st
ate=1)
```

```
#Scaled features values between 0-1 using standardization.
#Standardization use standard normal distribution to scale down values
ss=StandardScaler()
xtrain=ss.fit_transform(xtrain)
xtest=ss.transform(xtest)
```

```
from tensorflow.keras.layers import Dropout   #Dropout library is
imported to overcome the overfitting of model
ann=Sequential()
ann.add(Dense(units=9,activation="relu")) #Input layers with 9 neurons
ann.add(Dropout(0.3))  #Dropout rate
ann.add(Dense(units=9,activation="relu"))  #Hidden layer with 9 neuron
ann.add(Dense(units=1,activation="sigmoid")) #output layer
ann.compile(optimizer='adam',loss="binary_crossentropy")
ann.fit(xtrain,ytrain, epochs=95,validation_data=(xtest,ytest))
```

```
Epoch 1/95
13/13 [==============================] - 2s 52ms/step - loss: 0.7085 -
val_loss: 0.6715
Epoch 2/95
13/13 [==============================] - 0s 14ms/step - loss: 0.6184 -
val_loss: 0.6073
Epoch 3/95
```

```
13/13 [==============================] - 0s 10ms/step - loss: 0.5569 -
val_loss: 0.5506
Epoch 4/95
13/13 [==============================] - 0s 9ms/step - loss: 0.5198 -
val_loss: 0.4991
Epoch 5/95
13/13 [==============================] - 0s 10ms/step - loss: 0.4677 -
val_loss: 0.4479
Epoch 6/95
13/13 [==============================] - 0s 14ms/step - loss: 0.4091 -
val_loss: 0.3987
Epoch 7/95
13/13 [==============================] - 0s 10ms/step - loss: 0.3796 -
val_loss: 0.3529
Epoch 8/95
13/13 [==============================] - 0s 12ms/step - loss: 0.3227 -
val_loss: 0.3150
Epoch 9/95
13/13 [==============================] - 0s 16ms/step - loss: 0.3077 -
val_loss: 0.2830
Epoch 10/95
13/13 [==============================] - 0s 13ms/step - loss: 0.2786 -
val_loss: 0.2534
Epoch 11/95
13/13 [==============================] - 0s 9ms/step - loss: 0.2407 -
val_loss: 0.2302
Epoch 12/95
13/13 [==============================] - 0s 8ms/step - loss: 0.2248 -
val_loss: 0.2108
Epoch 13/95
13/13 [==============================] - 0s 9ms/step - loss: 0.2061 -
val_loss: 0.1941
Epoch 14/95
13/13 [==============================] - 0s 8ms/step - loss: 0.1889 -
val_loss: 0.1811
Epoch 15/95
13/13 [==============================] - 0s 9ms/step - loss: 0.1618 -
val_loss: 0.1712
Epoch 16/95
13/13 [==============================] - 0s 14ms/step - loss: 0.1520 -
val_loss: 0.1631
Epoch 17/95
13/13 [==============================] - 0s 14ms/step - loss: 0.1559 -
val_loss: 0.1562
Epoch 18/95
13/13 [==============================] - 0s 16ms/step - loss: 0.1640 -
val_loss: 0.1500
Epoch 19/95
13/13 [==============================] - 0s 9ms/step - loss: 0.1578 -
val_loss: 0.1442
```

```
Epoch 20/95
13/13 [==============================] - 0s 13ms/step - loss: 0.1415 -
val_loss: 0.1397
Epoch 21/95
13/13 [==============================] - 0s 14ms/step - loss: 0.1304 -
val_loss: 0.1362
Epoch 22/95
13/13 [==============================] - 0s 13ms/step - loss: 0.1307 -
val_loss: 0.1331
Epoch 23/95
13/13 [==============================] - 0s 21ms/step - loss: 0.1140 -
val_loss: 0.1305
Epoch 24/95
13/13 [==============================] - 0s 22ms/step - loss: 0.1220 -
val_loss: 0.1281
Epoch 25/95
13/13 [==============================] - 0s 14ms/step - loss: 0.1082 -
val_loss: 0.1262
Epoch 26/95
13/13 [==============================] - 0s 18ms/step - loss: 0.1052 -
val_loss: 0.1245
Epoch 27/95
13/13 [==============================] - 0s 18ms/step - loss: 0.0892 -
val_loss: 0.1230
Epoch 28/95
13/13 [==============================] - 0s 16ms/step - loss: 0.1139 -
val_loss: 0.1220
Epoch 29/95
13/13 [==============================] - 0s 10ms/step - loss: 0.1073 -
val_loss: 0.1218
Epoch 30/95
13/13 [==============================] - 0s 13ms/step - loss: 0.1043 -
val_loss: 0.1218
Epoch 31/95
13/13 [==============================] - 0s 14ms/step - loss: 0.1105 -
val_loss: 0.1218
Epoch 32/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0849 -
val_loss: 0.1218
Epoch 33/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0852 -
val_loss: 0.1208
Epoch 34/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0830 -
val_loss: 0.1205
Epoch 35/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0981 -
val_loss: 0.1201
Epoch 36/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0700 -
```

```
val_loss: 0.1198
Epoch 37/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0838 -
val_loss: 0.1195
Epoch 38/95
13/13 [==============================] - 0s 17ms/step - loss: 0.0860 -
val_loss: 0.1195
Epoch 39/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0803 -
val_loss: 0.1190
Epoch 40/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0802 -
val_loss: 0.1186
Epoch 41/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0733 -
val_loss: 0.1187
Epoch 42/95
13/13 [==============================] - 0s 7ms/step - loss: 0.0789 -
val_loss: 0.1192
Epoch 43/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0740 -
val_loss: 0.1194
Epoch 44/95
13/13 [==============================] - 0s 21ms/step - loss: 0.0766 -
val_loss: 0.1194
Epoch 45/95
13/13 [==============================] - 0s 15ms/step - loss: 0.0683 -
val_loss: 0.1192
Epoch 46/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0854 -
val_loss: 0.1192
Epoch 47/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0702 -
val_loss: 0.1199
Epoch 48/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0700 -
val_loss: 0.1200
Epoch 49/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0778 -
val_loss: 0.1196
Epoch 50/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0713 -
val_loss: 0.1199
Epoch 51/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0690 -
val_loss: 0.1203
Epoch 52/95
13/13 [==============================] - 0s 16ms/step - loss: 0.0675 -
val_loss: 0.1209
Epoch 53/95
```

```
13/13 [==============================] - 0s 19ms/step - loss: 0.0567 -
val_loss: 0.1216
Epoch 54/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0720 -
val_loss: 0.1249
Epoch 55/95
13/13 [==============================] - 0s 18ms/step - loss: 0.0606 -
val_loss: 0.1237
Epoch 56/95
13/13 [==============================] - 0s 19ms/step - loss: 0.0617 -
val_loss: 0.1241
Epoch 57/95
13/13 [==============================] - 0s 23ms/step - loss: 0.0532 -
val_loss: 0.1245
Epoch 58/95
13/13 [==============================] - 0s 19ms/step - loss: 0.0444 -
val_loss: 0.1243
Epoch 59/95
13/13 [==============================] - 0s 19ms/step - loss: 0.0756 -
val_loss: 0.1230
Epoch 60/95
13/13 [==============================] - 0s 7ms/step - loss: 0.0516 -
val_loss: 0.1244
Epoch 61/95
13/13 [==============================] - 0s 16ms/step - loss: 0.0697 -
val_loss: 0.1248
Epoch 62/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0722 -
val_loss: 0.1245
Epoch 63/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0554 -
val_loss: 0.1250
Epoch 64/95
13/13 [==============================] - 0s 14ms/step - loss: 0.0642 -
val_loss: 0.1249
Epoch 65/95
13/13 [==============================] - 0s 15ms/step - loss: 0.0623 -
val_loss: 0.1250
Epoch 66/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0585 -
val_loss: 0.1254
Epoch 67/95
13/13 [==============================] - 0s 16ms/step - loss: 0.0572 -
val_loss: 0.1251
Epoch 68/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0532 -
val_loss: 0.1255
Epoch 69/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0574 -
val_loss: 0.1259
```

```
Epoch 70/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0382 -
val_loss: 0.1263
Epoch 71/95
13/13 [==============================] - 0s 10ms/step - loss: 0.0480 -
val_loss: 0.1262
Epoch 72/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0488 -
val_loss: 0.1256
Epoch 73/95
13/13 [==============================] - 0s 14ms/step - loss: 0.0573 -
val_loss: 0.1267
Epoch 74/95
13/13 [==============================] - 0s 10ms/step - loss: 0.0490 -
val_loss: 0.1273
Epoch 75/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0483 -
val_loss: 0.1278
Epoch 76/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0465 -
val_loss: 0.1285
Epoch 77/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0560 -
val_loss: 0.1285
Epoch 78/95
13/13 [==============================] - 0s 10ms/step - loss: 0.0317 -
val_loss: 0.1294
Epoch 79/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0409 -
val_loss: 0.1294
Epoch 80/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0463 -
val_loss: 0.1312
Epoch 81/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0493 -
val_loss: 0.1319
Epoch 82/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0508 -
val_loss: 0.1317
Epoch 83/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0451 -
val_loss: 0.1354
Epoch 84/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0480 -
val_loss: 0.1339
Epoch 85/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0501 -
val_loss: 0.1314
Epoch 86/95
13/13 [==============================] - 0s 7ms/step - loss: 0.0511 -
```

```
val_loss: 0.1310
Epoch 87/95
13/13 [==============================] - 0s 7ms/step - loss: 0.0484 -
val_loss: 0.1324
Epoch 88/95
13/13 [==============================] - 0s 9ms/step - loss: 0.0455 -
val_loss: 0.1329
Epoch 89/95
13/13 [==============================] - 0s 11ms/step - loss: 0.0404 -
val_loss: 0.1337
Epoch 90/95
13/13 [==============================] - 0s 8ms/step - loss: 0.0523 -
val_loss: 0.1337
Epoch 91/95
13/13 [==============================] - 0s 12ms/step - loss: 0.0382 -
val_loss: 0.1352
Epoch 92/95
13/13 [==============================] - 0s 15ms/step - loss: 0.0408 -
val_loss: 0.1348
Epoch 93/95
13/13 [==============================] - 0s 16ms/step - loss: 0.0438 -
val_loss: 0.1350
Epoch 94/95
13/13 [==============================] - 0s 13ms/step - loss: 0.0346 -
val_loss: 0.1337
Epoch 95/95
13/13 [==============================] - 0s 18ms/step - loss: 0.0395 -
val_loss: 0.1347
```

```
<keras.callbacks.History at 0x7fd811385290>
```

```
ann.history.history #Stores the values of accuracy and loss
```

```
{'loss': [0.7085155844688416,
  0.6183828115463257,
  0.5568632483482361,
  0.5198155641555786,
  0.46768566966056824,
  0.4091082215309143,
  0.37956905364990234,
  0.3226575553417206,
  0.3076944649219513,
  0.2786177694797516,
  0.2406948357820511,
  0.2248343825340271,
  0.20606398582458496,
  0.1888991892337799,
  0.16176488995552063,
  0.15198779106140137,
  0.15585193037986755,
  0.16397114098072052,
```

0.15778663754463196,
0.1414880007505417,
0.13036173582077026,
0.1306859403848648,
0.11395561695098877,
0.12198159843683243,
0.10818268358707428,
0.10524007678031921,
0.08915788680315018,
0.11387239396572113,
0.10732096433639526,
0.1042557880282402,
0.11052108556032181,
0.08485647290945053,
0.0852065235376358,
0.08299896866083145,
0.09811722487211227,
0.07004676014184952,
0.08376562595367432,
0.08602239936590195,
0.08025651425123215,
0.08023986965417862,
0.07328962534666061,
0.07886822521686554,
0.07401341944932938,
0.07664569467306137,
0.06827549636363983,
0.08536963909864426,
0.07020195573568344,
0.06996796280145645,
0.07778391242027283,
0.07126965373754501,
0.06895831972360611,
0.06751521676778793,
0.05666523799300194,
0.07204030454158783,
0.06063591316342354,
0.061707351356744766,
0.053173888474702835,
0.04440544173121452,
0.07563692331314087,
0.051616210490465164,
0.06967682391405106,
0.07220861315727234,
0.05543136969208717,
0.06424283236265182,
0.06226936727762222,
0.05848338082432747,
0.05719596520066261,
0.053188011050224304,

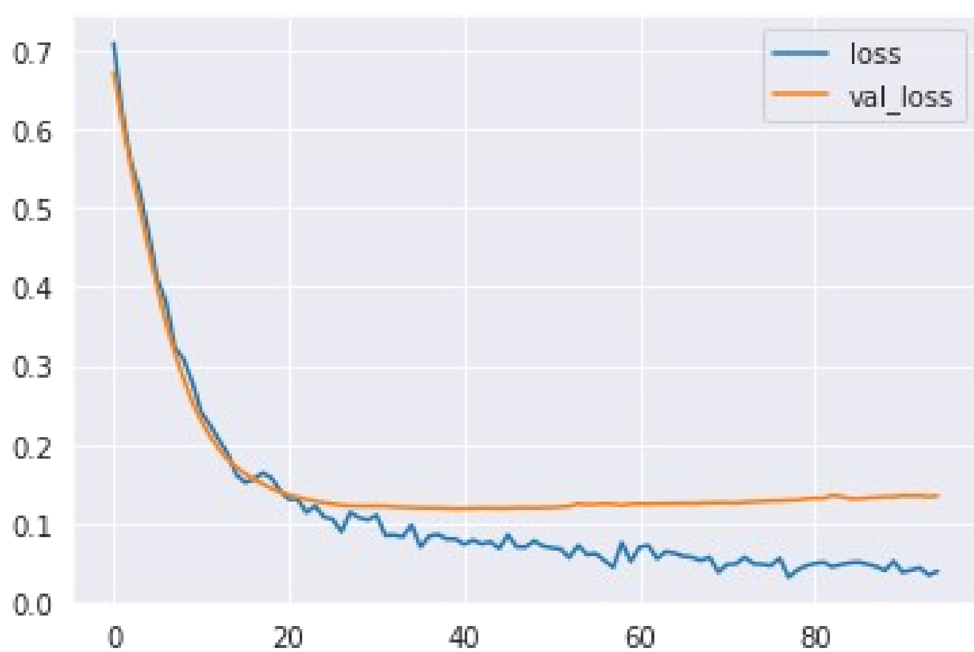
```
 0.05741815268993378,
 0.03823012113571167,
 0.048032086342573166,
 0.04876413941383362,
 0.05730469524860382,
 0.04895208403468132,
 0.04829488694667816,
 0.04651566967368126,
 0.056036632508039474,
 0.0316941998898983,
 0.04094119369983673,
 0.04627348855137825,
 0.04934763163328171,
 0.05080067738890648,
 0.04511204734444618,
 0.047957032918930054,
 0.05009686201810837,
 0.051059287041425705,
 0.048359405249357224,
 0.04547073319554329,
 0.04040654003620148,
 0.05230729281902313,
 0.03822548687458038,
 0.04083818569779396,
 0.04376918449997902,
 0.034638773649930954,
 0.0394633449614048],
'val_loss': [0.6715471744537354,
 0.6073440313339233,
 0.5505558252334595,
 0.4991188645362854,
 0.4479442834854126,
 0.398729145526886,
 0.352931410074234,
 0.3149756193161011,
 0.28295186161994934,
 0.2533795237541199,
 0.2301672101020813,
 0.21077904105186462,
 0.19410014152526855,
 0.18113777041435242,
 0.17122909426689148,
 0.16313831508159637,
 0.15620113909244537,
 0.15004046261310577,
 0.14417916536331177,
 0.13973785936832428,
 0.13619877398014069,
 0.13311965763568878,
 0.1304766833782196,
```

```
0.12814542651176453,
0.1261638104915619,
0.12453979253768921,
0.12303834408521652,
0.121981181204319,
0.12180085480213165,
0.12183047831058502,
0.1218178197741508S,
0.12181733548641205,
0.12083949148654938,
0.12050126492977142,
0.12014751881361008,
0.11984813213348389,
0.11952053010463715,
0.11948549747467041,
0.11904438585042953,
0.11859016865491867,
0.11865921318531036,
0.1191963404417038,
0.11942481994628906,
0.11943542957305908,
0.11924048513174057,
0.11918630450963974,
0.11986076831817627,
0.11995445191860199,
0.11963730305433273,
0.11989534646272659,
0.12025191634893417,
0.12092764675617218,
0.12156343460083008,
0.12487431615591049,
0.123672254383564,
0.12412488460540771,
0.12454016506671906,
0.12430323660373688,
0.1230112686753273,
0.12441370636224747,
0.12475844472646713,
0.12447383999824524,
0.12498413771390915,
0.12485883384943008,
0.12498756498098373,
0.1254122108221054,
0.12506616115570068,
0.12554317712783813,
0.12592335045337677,
0.12625554203987122,
0.12615403532981873,
0.1255824863910675,
0.12666140496730804,
```

```
 0.1272813081741333,
 0.127815842628479,
 0.1285279542207718,
 0.1284901350736618,
 0.12944534420967102,
 0.1293952763080597,
 0.13118453323841095,
 0.13191631436347961,
 0.1316794455051422,
 0.13537167012691498,
 0.13390785455703735,
 0.13139313459396362,
 0.13098473846912384,
 0.1323702186346054,
 0.13287129998207092,
 0.1336919069290161,
 0.1336599588394165,
 0.13520963490009308,
 0.13476471602916718,
 0.13495008647441864,
 0.13372690975666046,
 0.13474908471107483]}
```

```
lossdf=pd.DataFrame(ann.history.history)
lossdf.plot()
print('             Loss Vs Validation loss')
```

```
             Loss Vs Validation loss
```



```
ypred=ann.predict(xtest)
ypred=ypred>0.4  #Set threshold to 0.4
```

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

```
              precision    recall  f1-score   support

           0       0.96      0.96      0.96       108
           1       0.94      0.94      0.94        63

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

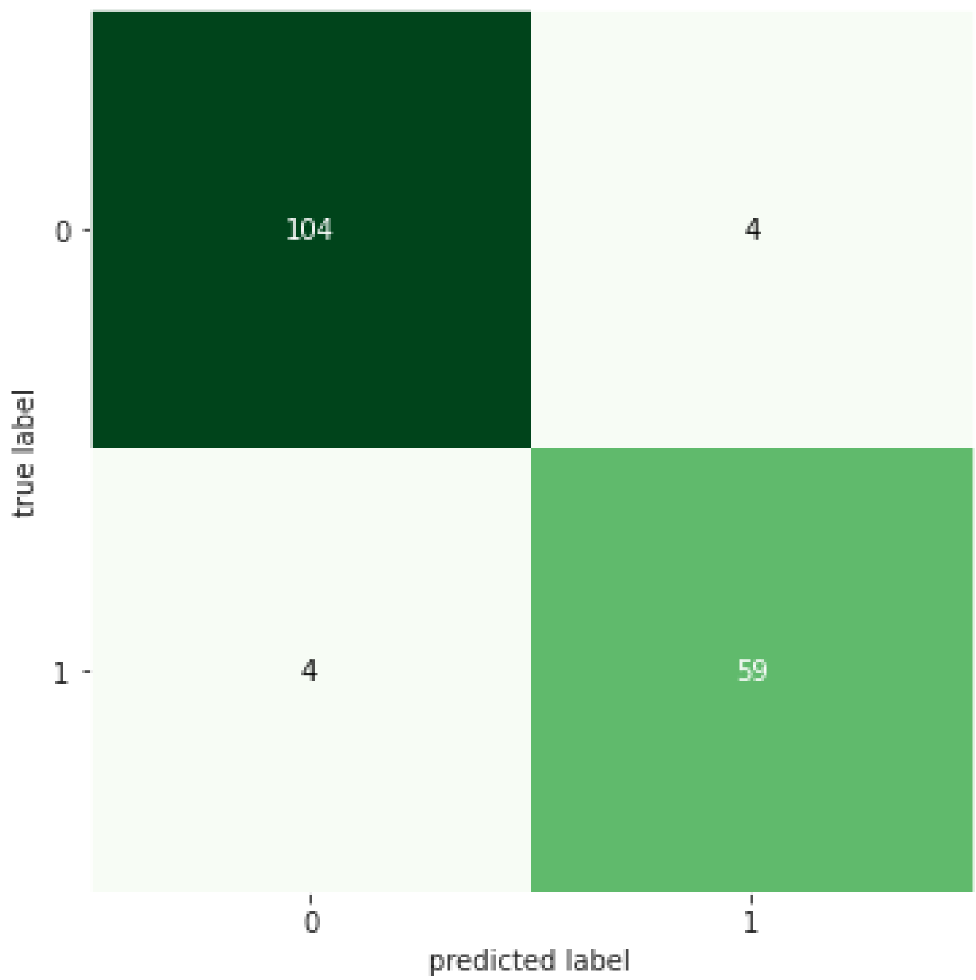
CLASSIFICATION REPORT : It helps to interpret the precision,recall, fl_score, support and accuracy.

```
print(confusion_matrix(ytest,ypred))
```

```
[[104   4]
 [  4  59]]
```

CONFUSION MATRIX: It helps to interpret the values such as True Negative(104), Flase positive(4), False Negative(3) and True Positive(60).

```
#Graphically representation of confusion matrix
from mlxtend.plotting import plot_confusion_matrix
fig, ax = plot_confusion_matrix(confusion_matrix(ytest,ypred),
figsize=(6, 6), cmap=plt.cm.Greens)
```


104
4
4
59
0
1
0
1
true label
predicted label