Importing Libraries

```
import pandas as pd
import numpy as np
import seaborn as sns
import warnings
warnings.filterwarnings("ignore")
```

```
import tensorflow as tf
```

Loading Data

```
df=pd.read_csv("diabetes.csv")
```

```
df.head()
```

	Pregnancies	Glucose	BloodPressure	SkinThickness	Insulin	BMI	DiabetesPedigreeFunction	Age	Outcome
0	6	148	72	35	0	33.6	0.627	50	1
1	1	85	66	29	0	26.6	0.351	31	0
2	8	183	64	0	0	23.3	0.672	32	1
3	1	89	66	23	94	28.1	0.167	21	0
4	0	137	40	35	168	43.1	2.288	33	1

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 768 entries, 0 to 767
Data columns (total 9 columns):
 #   Column                    Non-Null Count  Dtype  
---  ------                    --------------  -----  
 0   Pregnancies               768 non-null    int64  
 1   Glucose                   768 non-null    int64  
 2   BloodPressure             768 non-null    int64  
                               768 non-null    int64  
                               768 non-null    int64  
 5   BMI                       768 non-null    float64
 6   DiabetesPedigreeFunction  768 non-null    float64
 7   Age                       768 non-null    int64  
 8   Outcome                   768 non-null    int64  
dtypes: float64(2), int64(7)
memory usage: 54.1 KB
```

Saving... ✕

Standardization

```
from sklearn.preprocessing import StandardScaler
SS = StandardScaler()
X=df.iloc[:,0:-1]
X = SS.fit_transform(X)
X
```

```
array([[ 0.63994726,  0.84832379,  0.14964075, ...,  0.20401277,
         0.46849198,  1.4259954 ],
       [-0.84488505, -1.12339636, -0.16054575, ..., -0.68442195,
        -0.36506078, -0.19067191],
       [ 1.23388019,  1.94372388, -0.26394125, ..., -1.10325546,
         0.60439732, -0.10558415],
       ...,
       [ 0.3429808 ,  0.00330087,  0.14964075, ..., -0.73518964,
        -0.68519336, -0.27575966],
       [-0.84488505,  0.1597866 , -0.47073225, ..., -0.24020459,
        -0.37110101,  1.17073215],
       [-0.84488505, -0.8730192 ,  0.04624525, ..., -0.20212881,
        -0.47378505, -0.87137393]])
```

```
Y=df["Outcome"].values
Y
```

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

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=0.2)
```

```
from tensorflow.keras import Sequential
from tensorflow.keras.layers import Dense
from sklearn.metrics import classification_report
from tensorflow.keras.callbacks import EarlyStopping
```

```
import pathlib
ES = EarlyStopping(monitor="val_loss",mode="min",verbose=1,patience=25)
```

Modelling

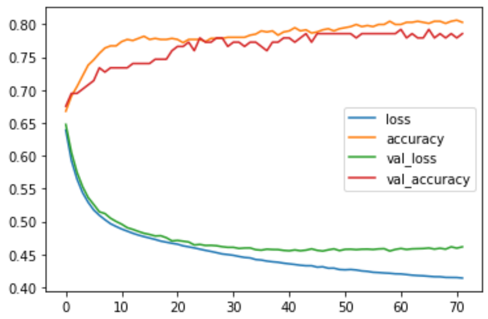
```
# step1 :- initialize the Model
ann = Sequential()

# step2 :- Add Layers into model
ann.add( Dense(units = 10, activation = "relu") )
ann.add( Dense(units = 10, activation = "relu") )
ann.add( Dense(units = 1, activation = "sigmoid") )

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

# step4 :- Fit the model
ann.fit(X_train, Y_train, batch_size = 25, epochs = 800,validation_data=(X_test,Y_test),callbacks=ES)
```

```
Epoch 1/800
25/25 [==============================] - 1s 13ms/step - loss: 0.6390 - accuracy: 0.6678 - val_loss: 0.6476 - val_accuracy: 0.6753
Epoch 2/800
25/25 [==============================] - 0s 3ms/step - loss: 0.5928 - accuracy: 0.6906 - val_loss: 0.6052 - val_accuracy: 0.6948
Epoch 3/800
25/25 [==============================] - 0s 4ms/step - loss: 0.5647 - accuracy: 0.7052 - val_loss: 0.5749 - val_accuracy: 0.6948
Epoch 4/800
25/25 [==============================] - 0s 4ms/step - loss: 0.5439 - accuracy: 0.7215 - val_loss: 0.5531 - val_accuracy: 0.7013
Epoch 5/800
25/25 [==============================] - 0s 4ms/step - loss: 0.5292 - accuracy: 0.7378 - val_loss: 0.5365 - val_accuracy: 0.7078
Epoch 6/800
25/25 [==============================] - 0s 4ms/step - loss: 0.5174 - accuracy: 0.7459 - val_loss: 0.5258 - val_accuracy: 0.7143
Epoch 7/800
25/25 [==============================] - 0s 4ms/step - loss: 0.5095 - accuracy: 0.7557 - val_loss: 0.5148 - val_accuracy: 0.7338
Epoch 8/800
25/25 [==============================] - 0s 5ms/step - loss: 0.5028 - accuracy: 0.7638 - val_loss: 0.5121 - val_accuracy: 0.7273
Epoch 9/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4966 - accuracy: 0.7671 - val_loss: 0.5050 - val_accuracy: 0.7338
Epoch 10/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4924 - accuracy: 0.7671 - val_loss: 0.5001 - val_accuracy: 0.7338
Epoch 11/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4885 - accuracy: 0.7736 - val_loss: 0.4959 - val_accuracy: 0.7338
Epoch 12/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4853 - accuracy: 0.7769 - val_loss: 0.4906 - val_accuracy: 0.7338
Epoch 13/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4821 - accuracy: 0.7752 - val_loss: 0.4879 - val_accuracy: 0.7403
Epoch 14/800
                                ======] - 0s 5ms/step - loss: 0.4793 - accuracy: 0.7785 - val_loss: 0.4848 - val_accuracy: 0.7403
   Saving...
                                ======] - 0s 4ms/step - loss: 0.4769 - accuracy: 0.7818 - val_loss: 0.4820 - val_accuracy: 0.7403
Epoch 16/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4748 - accuracy: 0.7769 - val_loss: 0.4803 - val_accuracy: 0.7403
Epoch 17/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4727 - accuracy: 0.7785 - val_loss: 0.4779 - val_accuracy: 0.7468
Epoch 18/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4702 - accuracy: 0.7769 - val_loss: 0.4783 - val_accuracy: 0.7468
Epoch 19/800
25/25 [==============================] - 0s 3ms/step - loss: 0.4685 - accuracy: 0.7769 - val_loss: 0.4753 - val_accuracy: 0.7468
Epoch 20/800
25/25 [==============================] - 0s 3ms/step - loss: 0.4669 - accuracy: 0.7785 - val_loss: 0.4705 - val_accuracy: 0.7597
Epoch 21/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4655 - accuracy: 0.7769 - val_loss: 0.4712 - val_accuracy: 0.7662
Epoch 22/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4630 - accuracy: 0.7720 - val_loss: 0.4700 - val_accuracy: 0.7662
Epoch 23/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4614 - accuracy: 0.7769 - val_loss: 0.4688 - val_accuracy: 0.7727
Epoch 24/800
25/25 [==============================] - 0s 3ms/step - loss: 0.4598 - accuracy: 0.7769 - val_loss: 0.4643 - val_accuracy: 0.7597
Epoch 25/800
25/25 [==============================] - 0s 4ms/step - loss: 0.4579 - accuracy: 0.7769 - val_loss: 0.4651 - val_accuracy: 0.7792
Epoch 26/800
25/25 [==============================] - 0s 3ms/step - loss: 0.4561 - accuracy: 0.7736 - val_loss: 0.4635 - val_accuracy: 0.7727
Epoch 27/800
25/25 [==============================] - 0s 5ms/step - loss: 0.4544 - accuracy: 0.7785 - val_loss: 0.4637 - val_accuracy: 0.7727
Epoch 28/800
25/25 [==============================] - 0s 9ms/step - loss: 0.4526 - accuracy: 0.7785 - val_loss: 0.4631 - val_accuracy: 0.7792
Epoch 29/800
25/25 [==============================] - 0s 7ms/step - loss: 0.4506 - accuracy: 0.7785 - val_loss: 0.4616 - val_accuracy: 0.7792
```

Visualizing the loss

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

```
<AxesSubplot:>
```



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

```
       [0.5836838 ]
       [0.05013222]
       [0.14037022]
       [0.9129216 ]
       [0.17780566]
       [0.14710547]
       [0.27867275]
       [0.7915911 ]
       [0.03340699]
       [0.24117239]
       [0.34339124]
       [0.12007734]
       [0.22498174]
       [0.055947  ]
       [0.03591388]
       [0.47584707]
       [0.943164  ]
       [0.9062208 ]
       [0.01757122]
       [0.03590105]
       [0.08281112]
       [0.84418625]
       [0.46636075]]
```

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

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

Saving...

```
            [1],
            [0],
            [1],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
            [1],
            [0],
            [1],
            [0],
            [0],
            [1],
            [1],
            [0],
            [0],
            [1],
            [0],
            [0],
            [0],
            [0],
            [1],
            [0],
            [0],
            [1],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
            [0],
```

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

```
                   precision    recall  f1-score   support

                0       0.83      0.88      0.86       111
                1       0.64      0.53      0.58        43

         accuracy                           0.79       154
        macro avg       0.73      0.71      0.72       154
     weighted avg       0.78      0.79      0.78       154
```