In [1]:
```
from sklearn.datasets import load_iris
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
from sklearn.decomposition import PCA
from sklearn.pipeline import Pipeline
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from xgboost import XGBClassifier
```

In [2]:
```
# Suppressing Warnings
import warnings
warnings.filterwarnings('ignore')
```

In [3]:
```
# Importing Pandas and NumPy
import pandas as pd, numpy as np
```

In [5]:
```
# Importing all datasets
diabetes = pd.read_csv("/content/sample_data/pima_indian_diabetes.csv")
diabetes.head(4)
```

Out[5]:

	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age	Diabetes
0	1	89	66	23	94	28.1	21	0
1	0	137	40	35	168	43.1	33	1
2	3	78	50	32	88	31.0	26	1
3	2	197	70	45	543	30.5	53	1

In [6]:
```
diabetes.dtypes
```

Out[6]:
```
No_Times_Pregnant      int64
Plasma_Glucose         int64
Diastolic_BP           int64
Triceps                int64
Insulin                int64
BMI                  float64
Age                    int64
Diabetes               int64
dtype: object
```

In [7]:
```
diabetes.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 392 entries, 0 to 391
Data columns (total 8 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   No_Times_Pregnant  392 non-null    int64  
 1   Plasma_Glucose     392 non-null    int64  
 2   Diastolic_BP       392 non-null    int64  
 3   Triceps            392 non-null    int64  
 4   Insulin            392 non-null    int64  
 5   BMI                392 non-null    float64
 6   Age                392 non-null    int64  
 7   Diabetes           392 non-null    int64  
dtypes: float64(1), int64(7)
memory usage: 24.6 KB
```

In [8]:
```
diabetes.columns
```

Out[8]:
```
Index(['No_Times_Pregnant', 'Plasma_Glucose', 'Diastolic_BP', 'Triceps',
       'Insulin', 'BMI', 'Age', 'Diabetes'],
      dtype='object')
```

Rescaling the Features

In [9]:
```
from sklearn.preprocessing import MinMaxScaler
scaler = MinMaxScaler()
```

In [11]:
```
# Apply scaler() to all the columns except the 'yes-no' and 'dummy' variables
num_vars = ['No_Times_Pregnant', 'Plasma_Glucose', 'Diastolic_BP', 'Triceps',
       'Insulin', 'BMI', 'Age', 'Diabetes']

diabetes[num_vars] = scaler.fit_transform(diabetes[num_vars])

diabetes.head()
```

Out[11]:

	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age	Diabetes
0	0.058824	0.232394	0.488372	0.285714	0.096154	0.202454	0.000000	0.0
1	0.000000	0.570423	0.186047	0.500000	0.185096	0.509202	0.200000	1.0
2	0.176471	0.154930	0.302326	0.446429	0.088942	0.261759	0.083333	1.0
3	0.117647	0.992958	0.534884	0.678571	0.635817	0.251534	0.533333	1.0
4	0.058824	0.936620	0.418605	0.285714	1.000000	0.243354	0.633333	1.0

In [12]:
```
## Checking for Outliers
```

In [14]:
```
# Checking for outliers in the continuous variables
num_diabetes = diabetes[['No_Times_Pregnant', 'Plasma_Glucose', 'Diastolic_BP', 'Triceps',
       'Insulin', 'BMI', 'Age', 'Diabetes']]
```

In [15]:
```
# Checking outliers at 25%, 50%, 75%, 90%, 95% and 99%
num_diabetes.describe(percentiles=[.25, .5, .75, .90, .95, .99])
```

Out[15]:

	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age	Diabetes
count	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000	392.000000
mean	0.194178	0.469208	0.542596	0.395454	0.170741	0.304422	0.164413	0.331633
std	0.188907	0.217329	0.145303	0.187793	0.142839	0.143715	0.170013	0.471401
min	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	0.058824	0.302817	0.441860	0.250000	0.075421	0.208589	0.033333	0.000000
50%	0.117647	0.443662	0.534884	0.392857	0.134014	0.306748	0.100000	0.000000
75%	0.294118	0.612676	0.627907	0.535714	0.211538	0.386503	0.250000	1.000000
90%	0.470588	0.809155	0.720930	0.642857	0.335096	0.488139	0.416667	1.000000
95%	0.588235	0.880282	0.767442	0.704464	0.459736	0.553067	0.524167	1.000000
99%	0.764706	0.985915	0.911163	0.803571	0.681358	0.719059	0.650000	1.000000
max	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000	1.000000

In [16]:
```
Q1 = num_diabetes.quantile(0.25)
Q3 = num_diabetes.quantile(0.75)
IQR = Q3 - Q1
print(IQR)
```

```
No_Times_Pregnant    0.235294
Plasma_Glucose       0.309859
Diastolic_BP         0.186047
Triceps              0.285714
Insulin              0.136118
BMI                  0.177914
Age                  0.216667
```

```
Diabetes              1.000000
dtype: float64
```

In [17]:
```
### Quantile-based Flooring and Capping
```

In [18]:
```
print(diabetes.quantile(0.05))
print(diabetes.quantile(0.95))
```

```
No_Times_Pregnant    0.000000
Plasma_Glucose       0.176056
Diastolic_BP         0.302326
Triceps              0.107143
Insulin              0.034315
BMI                  0.082924
Age                  0.000000
Diabetes             0.000000
Name: 0.05, dtype: float64
No_Times_Pregnant    0.588235
Plasma_Glucose       0.880282
Diastolic_BP         0.767442
Triceps              0.704464
Insulin              0.459736
BMI                  0.553067
Age                  0.524167
Diabetes             1.000000
Name: 0.95, dtype: float64
```

In [19]:
```
diabetes = diabetes[~((diabetes < (Q1 - 1.5 * IQR)) | (diabetes > (Q3 + 1.5 * IQR))).any(axis=1)]
print(diabetes.shape)
```

```
(332, 8)
```

In [20]:
```
### Test-Train Split
```

In [21]:
```
from sklearn.model_selection import train_test_split
```

In [22]:
```
diabetes.head(3)
```

Out[22]:

	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age	Diabetes
0	0.058824	0.232394	0.488372	0.285714	0.096154	0.202454	0.000000	0.0
1	0.000000	0.570423	0.186047	0.500000	0.185096	0.509202	0.200000	1.0
2	0.176471	0.154930	0.302326	0.446429	0.088942	0.261759	0.083333	1.0

In [24]:
```
# Putting feature variable to X
X = diabetes.drop(['Diabetes'], axis=1)
X.head()
```

Out[24]:

	No_Times_Pregnant	Plasma_Glucose	Diastolic_BP	Triceps	Insulin	BMI	Age
0	0.058824	0.232394	0.488372	0.285714	0.096154	0.202454	0.000000
1	0.000000	0.570423	0.186047	0.500000	0.185096	0.509202	0.200000
2	0.176471	0.154930	0.302326	0.446429	0.088942	0.261759	0.083333
5	0.294118	0.774648	0.558140	0.214286	0.193510	0.155419	0.500000
6	0.000000	0.436620	0.697674	0.714286	0.259615	0.564417	0.166667

In [25]:
```
# Putting response variable to y
y = diabetes['Diabetes']
y.head()
```

Out[25]:
```
0    0.0
1    1.0
2    1.0
5    1.0
6    1.0
Name: Diabetes, dtype: float64
```

In [26]:
```
# Splitting the data into train and test
X_train, X_test, y_train, y_test = train_test_split(X, y, train_size=0.7, test_size=0.3, random_state=100)
```

In [27]:
```
print(X_train.shape)
print(X_test.shape)
print(y_train.shape)
print(y_test.shape)
```

```
(232, 7)
(100, 7)
(232,)
(100,)
```

In [ ]:
```
#!pip install keras-tuner
```

In [32]:
```
import pandas as pd
```

```
import numpy as np
import itertools
from sklearn import metrics
from sklearn.metrics import confusion_matrix
from tensorflow.keras.preprocessing.image import ImageDataGenerator, img_to_array, load_img
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Conv2D,Flatten,Dropout,Dense
from tensorflow.keras.optimizers import Adam
from tensorflow import keras
from tensorflow.keras import layers
from kerastuner.tuners import RandomSearch
```

In [34]:

```
import tensorflow as tf
from tensorflow import keras
import numpy as np
```

In [71]:

```
# initialising sequential model and adding layers to it
model = Sequential([
    Dense(32, activation='relu', input_shape=(7,)),
    Dense(32, activation='relu'),
    Dense(32, activation='leaky_relu')
])
model.add(Dropout(0.2))
model.add(Dense(500, activation="relu"))
model.add(Dense(300, activation="relu"))

model.add(Dense(1, activation='sigmoid'))
```

In [72]:

```
model.compile(optimizer="adam", loss="categorical_crossentropy", metrics=["accuracy"])
```

In [73]:

```
model.fit(X_train, y_train, epochs=100, batch_size=100, verbose=1)
```

```
Epoch 1/100
3/3 [==============================] - 1s 8ms/step - loss: 0.0000e+00 - accuracy: 0.6207
Epoch 2/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 3/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 4/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 5/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 6/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
```

```
Epoch 7/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 8/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 9/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 10/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 11/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 12/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 13/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 14/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 15/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 16/100
3/3 [==============================] - 0s 11ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 17/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 18/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 19/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 20/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 21/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 22/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 23/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 24/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 25/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 26/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 27/100
3/3 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 28/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 29/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 30/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 31/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 32/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 33/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
```

```
Epoch 34/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 35/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 36/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 37/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 38/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 39/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 40/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 41/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 42/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 43/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 44/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 45/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 46/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 47/100
3/3 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 48/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 49/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 50/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 51/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 52/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 53/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 54/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 55/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 56/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 57/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 58/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 59/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 60/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
```

```
Epoch 61/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 62/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 63/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 64/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 65/100
3/3 [==============================] - 0s 12ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 66/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 67/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 68/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 69/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 70/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 71/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 72/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 73/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 74/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 75/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 76/100
3/3 [==============================] - 0s 11ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 77/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 78/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 79/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 80/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 81/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 82/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 83/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 84/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 85/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 86/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 87/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
```

```
Epoch 88/100
3/3 [==============================] - 0s 7ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 89/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 90/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 91/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 92/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 93/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 94/100
3/3 [==============================] - 0s 11ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 95/100
3/3 [==============================] - 0s 9ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 96/100
3/3 [==============================] - 0s 10ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 97/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 98/100
3/3 [==============================] - 0s 11ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 99/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
Epoch 100/100
3/3 [==============================] - 0s 8ms/step - loss: 0.0000e+00 - accuracy: 0.7112
```

Out[73]:
```
<keras.callbacks.History at 0x7f651c1abcd0>
```

In [74]:
```
accuracy = model.evaluate(X_test, y_test)
print(accuracy)
```

```
4/4 [==============================] - 0s 4ms/step - loss: 0.0000e+00 - accuracy: 0.7400
[0.0, 0.7400000095367432]
```

In [77]:
```
y_pred = model.predict(X_test)
```

In [75]:
```
from sklearn.metrics import classification_report
```

In [81]:
```
print(classification_report(y_test, y_pred))
```

```
              precision    recall  f1-score   support

         0.0       0.74      1.00      0.85        74
         1.0       0.00      0.00      0.00        26

    accuracy                           0.74       100
   macro avg       0.37      0.50      0.43       100
```

```
weighted avg       0.55      0.74      0.63       100
```

In [ ]: