```
pip install keras
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: keras in /usr/local/lib/python3.10/dist-packages (2.12.0)
```

```
pip install tensorflow
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: tensorflow in /usr/local/lib/python3.10/dist-packages (2.12.0)
Requirement already satisfied: tensorflow-estimator<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.1
Requirement already satisfied: termcolor>=1.1.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.3.0)
Requirement already satisfied: typing-extensions>=3.6.6 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (4.5.0)
Requirement already satisfied: tensorflow-io-gcs-filesystem>=0.23.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0
Requirement already satisfied: absl-py>=1.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.4.0)
Requirement already satisfied: six>=1.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.16.0)
Requirement already satisfied: flatbuffers>=2.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.3.3)
Requirement already satisfied: opt-einsum>=2.3.2 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.3.0)
Requirement already satisfied: jax>=0.3.15 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.8)
Requirement already satisfied: wrapt<1.15,>=1.11.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.14.1)
Requirement already satisfied: grpcio<2.0,>=1.24.3 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.54.0)
Requirement already satisfied: google-pasta>=0.1.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.2.0)
Requirement already satisfied: numpy<1.24,>=1.22 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.22.4)
Requirement already satisfied: astunparse>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (1.6.3)
Requirement already satisfied: setuptools in /usr/local/lib/python3.10/dist-packages (from tensorflow) (67.7.2)
Requirement already satisfied: libclang>=13.0.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (16.0.0)
Requirement already satisfied: h5py>=2.9.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (3.8.0)
Requirement already satisfied: tensorboard<2.13,>=2.12 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.2)
Requirement already satisfied: packaging in /usr/local/lib/python3.10/dist-packages (from tensorflow) (23.1)
Requirement already satisfied: keras<2.13,>=2.12.0 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (2.12.0)
Requirement already satisfied: gast<=0.4.0,>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from tensorflow) (0.4.0)
Requirement already satisfied: protobuf!=4.21.0,!=4.21.1,!=4.21.2,!=4.21.3,!=4.21.4,!=4.21.5,<5.0.0dev,>=3.20.3 in /usr/local/lib/p
Requirement already satisfied: wheel<1.0,>=0.23.0 in /usr/local/lib/python3.10/dist-packages (from astunparse>=1.6.0->tensorflow) (
Requirement already satisfied: scipy>=1.7 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow) (1.10.1)
Requirement already satisfied: ml-dtypes>=0.0.3 in /usr/local/lib/python3.10/dist-packages (from jax>=0.3.15->tensorflow) (0.1.0)
Requirement already satisfied: google-auth<3,>=1.6.3 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tens
Requirement already satisfied: werkzeug>=1.0.1 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow
Requirement already satisfied: google-auth-oauthlib<1.1,>=0.5 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2
Requirement already satisfied: requests<3,>=2.21.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensor
Requirement already satisfied: tensorboard-data-server<0.8.0,>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2
Requirement already satisfied: tensorboard-plugin-wit>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.
Requirement already satisfied: markdown>=2.6.8 in /usr/local/lib/python3.10/dist-packages (from tensorboard<2.13,>=2.12->tensorflow
Requirement already satisfied: cachetools<6.0,>=2.0.0 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tenso
Requirement already satisfied: pyasn1-modules>=0.2.1 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensor
Requirement already satisfied: rsa<5,>=3.1.4 in /usr/local/lib/python3.10/dist-packages (from google-auth<3,>=1.6.3->tensorboard<2.
Requirement already satisfied: requests-oauthlib>=0.7.0 in /usr/local/lib/python3.10/dist-packages (from google-auth-oauthlib<1.1,>
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorbo
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tensorboard<2.13,
Requirement already satisfied: charset-normalizer~=2.0.0 in /usr/local/lib/python3.10/dist-packages (from requests<3,>=2.21.0->tens
Requirement already satisfied: MarkupSafe>=2.1.1 in /usr/local/lib/python3.10/dist-packages (from werkzeug>=1.0.1->tensorboard<2.13
Requirement already satisfied: pyasn1<0.6.0,>=0.4.6 in /usr/local/lib/python3.10/dist-packages (from pyasn1-modules>=0.2.1->google-
Requirement already satisfied: oauthlib>=3.0.0 in /usr/local/lib/python3.10/dist-packages (from requests-oauthlib>=0.7.0->google-au
```

part 1 Import necessary packages, data processing and analysis tools

```
import seaborn as sns
import sys
import pandas as pd
import numpy as np
import sklearn
from sklearn import model_selection
from sklearn.metrics import classification_report, accuracy_score
import matplotlib
import matplotlib.pyplot as plt
import keras
from keras.utils.np_utils import to_categorical
```

Loading the dataset

```
data=pd.read_csv("/content/ECG-Dataset.csv")
```

```
data.columns = ['age','sex','smoker','years_of_smoking','LDL_cholesterol','chest_pain_type','hei
                'activity', 'lifestyle', 'cardiac intervention', 'heart_rate', 'diabets', 'blood
                'hypertention', 'Interventricular_septal_end_diastole', 'ecg_pattern', 'Q_wave'
```

Review heart disease dataset samples

```
data.head()
```

	age	sex	smoker	years_of_smoking	LDL_cholesterol	chest_pain_type	height	weight	fami
0	65	0	0	0	69.0	4	168	111.0	
1	54	1	0	0	117.0	2	145	81.0	
2	61	0	1	45	86.2	2	160	72.0	
3	57	0	0	0	76.0	2	176	78.0	
4	62	1	0	0	160.0	3	154	61.0	

5 rows × 21 columns

```
data.tail()
```

```
data.shape
```

```
(333, 21)
```

```
data.describe()
```

	age	sex	smoker	years_of_smoking	LDL_cholesterol	chest_pain_type
count	333.000000	333.000000	333.000000	333.000000	333.000000	333.000000
mean	55.117117	0.534535	0.195195	4.798799	112.926246	2.885886
std	14.159210	0.499557	0.396947	11.249835	37.972983	1.032110
min	20.000000	0.000000	0.000000	0.000000	26.000000	1.000000
25%	44.000000	0.000000	0.000000	0.000000	86.200000	2.000000
50%	57.000000	1.000000	0.000000	0.000000	110.000000	3.000000
75%	67.000000	1.000000	0.000000	0.000000	137.000000	4.000000
max	90.000000	1.000000	1.000000	50.000000	260.000000	4.000000

8 rows × 21 columns

Exploratory Data Analysis Checking for Missing Values

```
data.isnull().sum()
```

```
age                                     0
sex                                     0
smoker                                  0
years_of_smoking                        0
LDL_cholesterol                         0
chest_pain_type                         0
height                                  0
weight                                  0
familyhist                              0
activity                                0
lifestyle                               0
cardiac intervention                    0
heart_rate                              0
diabets                                 0
blood_pressure_sys                      0
blood_pressure_dias                     0
hypertention                            0
Interventricular_septal_end_diastole    0
ecg_pattern                             0
Q_wave                                  0
target                                  0
```

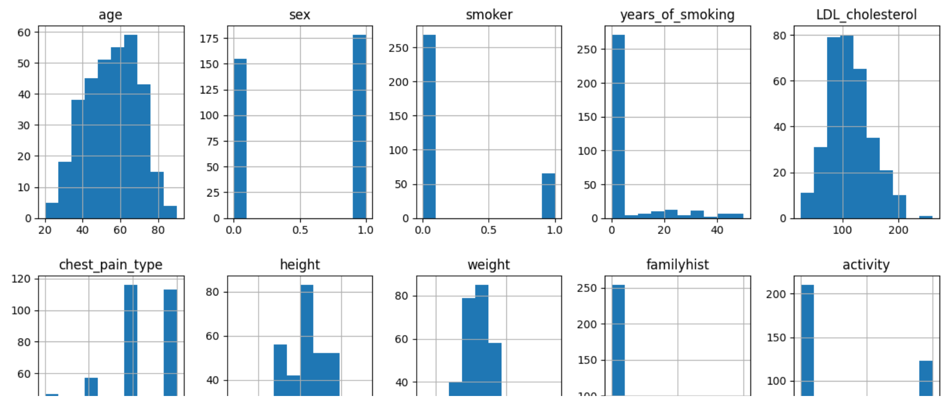
```
dtype: int64
```

It is shown that the dataset has no missing value.

Review All Features Data Distribution of All Participants

```
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
data.hist(ax = ax)
plt.show()
```

```
<ipython-input-8-62628a64ed50>:3: UserWarning: To output multiple subplots, the figure conta
  data.hist(ax = ax)
```

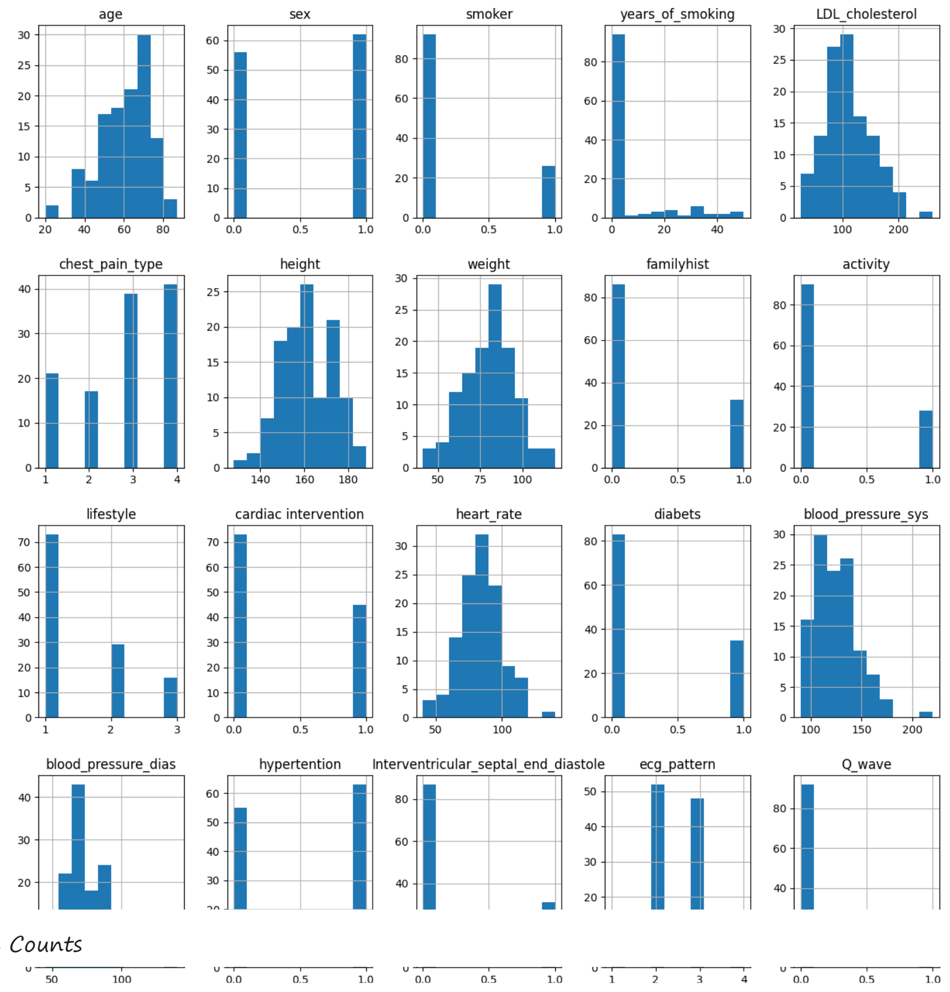


The data on the Q wave column is poorly balanced

Only Heart Disease Partipants

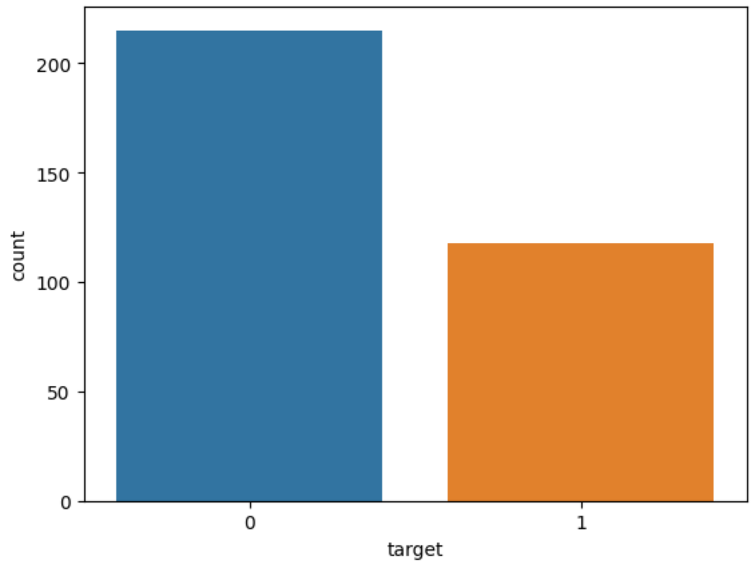
```
dataset_copy=data[data['target']==1]
columns=data.columns[:21]
fig = plt.figure(figsize = (15,20))
ax = fig.gca()
dataset_copy.hist(ax = ax)
plt.show()
```

```
<ipython-input-9-e8279b0a47db>:5: UserWarning: To output multiple subplots, the figure conta
  dataset_copy.hist(ax = ax)
```



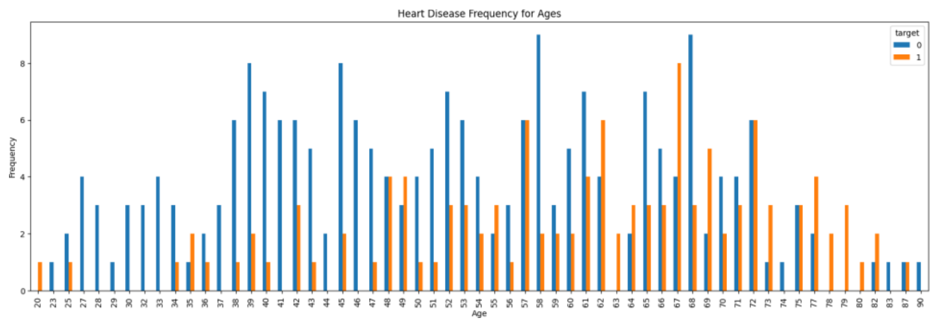
Case Counts

```
sns.countplot(x='target',data=data)
plt.show()
```



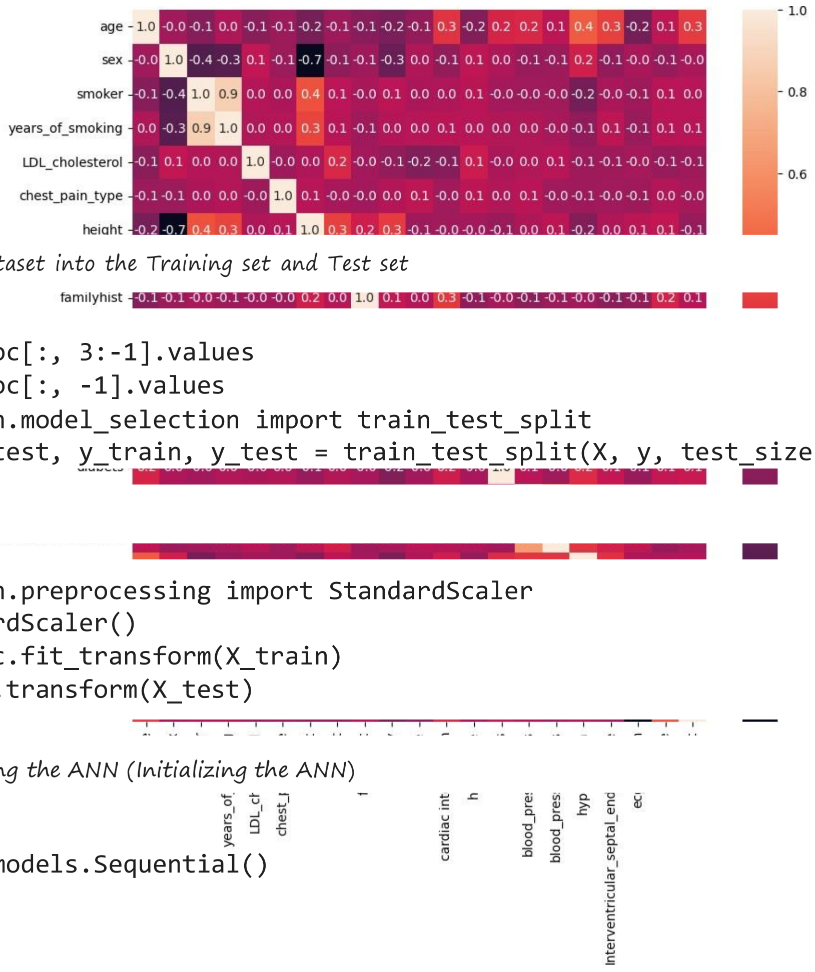
Heart Disease Frequency for Ages

```
pd.crosstab(data.age,data.target).plot(kind="bar",figsize=(20,6))
plt.title('Heart Disease Frequency for Ages')
plt.xlabel('Age')
plt.ylabel('Frequency')
plt.show()
```



Histogram Equalization of The Dataset in the form of heat map

```
plt.figure(figsize=(10,10))
sns.heatmap(data.corr(),annot=True,fmt='.1f')
plt.show()
```



Splitting the dataset into the Training set and Test set

```
X = data.iloc[:, 3:-1].values
y = data.iloc[:, -1].values
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, random_state = 0)
```

```
from sklearn.preprocessing import StandardScaler
sc = StandardScaler()
X_train = sc.fit_transform(X_train)
X_test = sc.transform(X_test)
```

Part 2 - Building the ANN (Initializing the ANN)

```
ann =keras.models.Sequential()
```

```
ann.add(keras.layers.Dense(units=8, activation='relu'))
```

Adding the second hidden layer

```
ann.add(keras.layers.Dense(units=8, activation='relu'))
ann.add(keras.layers.Dense(units=8, activation='relu'))
ann.add(keras.layers.Dense(units=8, activation='relu'))
ann.add(keras.layers.Dense(units=8, activation='relu'))
ann.add(keras.layers.Dense(units=8, activation='relu'))
```

Adding the output layer

```
ann.add(keras.layers.Dense(units=1, activation='sigmoid'))
```

Part 3 - Training the ANN (Compiling the ANN)

```
his=ann.fit(X_train, y_train, batch_size = 32, epochs =80,validation_split=0.33)
```

```
Epoch 1/80
6/6 [==============================] - 0s 35ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.2642 - val_accuracy: 0.9659
Epoch 2/80
6/6 [==============================] - 0s 22ms/step - loss: 0.0042 - accuracy: 1.0000 - val_loss: 0.2644 - val_accuracy: 0.9659
Epoch 3/80
6/6 [==============================] - 0s 16ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2646 - val_accuracy: 0.9659
Epoch 4/80
6/6 [==============================] - 0s 28ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2648 - val_accuracy: 0.9659
Epoch 5/80
6/6 [==============================] - 0s 21ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2651 - val_accuracy: 0.9659
```

```
Epoch 6/80
6/6 [==============================] - 0s 18ms/step - loss: 0.0041 - accuracy: 1.0000 - val_loss: 0.2653 - val_accuracy: 0.9659
Epoch 7/80
6/6 [==============================] - 0s 21ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.2655 - val_accuracy: 0.9659
Epoch 8/80
```

```
6/6 [==============================] - 0s 26ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.2657 - val_accuracy: 0.9659
Epoch 9/80
6/6 [==============================] - 0s 23ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.2659 - val_accuracy: 0.9659
Epoch 10/80
6/6 [==============================] - 0s 42ms/step - loss: 0.0040 - accuracy: 1.0000 - val_loss: 0.2661 - val_accuracy: 0.9659
Epoch 11/80
6/6 [==============================] - 0s 56ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.2663 - val_accuracy: 0.9659
Epoch 12/80
6/6 [==============================] - 0s 73ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.2665 - val_accuracy: 0.9659
Epoch 13/80
6/6 [==============================] - 0s 93ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.2667 - val_accuracy: 0.9659
Epoch 14/80
6/6 [==============================] - 0s 69ms/step - loss: 0.0039 - accuracy: 1.0000 - val_loss: 0.2669 - val_accuracy: 0.9659
Epoch 15/80
6/6 [==============================] - 0s 63ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.2671 - val_accuracy: 0.9659
Epoch 16/80
6/6 [==============================] - 0s 30ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.2673 - val_accuracy: 0.9659
Epoch 17/80
6/6 [==============================] - 0s 18ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.2675 - val_accuracy: 0.9659
Epoch 18/80
6/6 [==============================] - 0s 20ms/step - loss: 0.0038 - accuracy: 1.0000 - val_loss: 0.2678 - val_accuracy: 0.9659
Epoch 19/80
6/6 [==============================] - 0s 14ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.2680 - val_accuracy: 0.9659
Epoch 20/80
6/6 [==============================] - 0s 18ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.2682 - val_accuracy: 0.9659
Epoch 21/80
6/6 [==============================] - 0s 15ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.2684 - val_accuracy: 0.9659
Epoch 22/80
6/6 [==============================] - 0s 41ms/step - loss: 0.0037 - accuracy: 1.0000 - val_loss: 0.2686 - val_accuracy: 0.9659
Epoch 23/80
6/6 [==============================] - 0s 24ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.2688 - val_accuracy: 0.9659
Epoch 24/80
6/6 [==============================] - 0s 26ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.2690 - val_accuracy: 0.9659
Epoch 25/80
6/6 [==============================] - 0s 18ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.2692 - val_accuracy: 0.9659
Epoch 26/80
6/6 [==============================] - 0s 21ms/step - loss: 0.0036 - accuracy: 1.0000 - val_loss: 0.2694 - val_accuracy: 0.9659
Epoch 27/80
6/6 [==============================] - 0s 19ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.2696 - val_accuracy: 0.9659
Epoch 28/80
6/6 [==============================] - 0s 25ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.2698 - val_accuracy: 0.9659
Epoch 29/80
6/6 [==============================] - 0s 19ms/step - loss: 0.0035 - accuracy: 1.0000 - val_loss: 0.2700 - val_accuracy: 0.9659
```

Part 4 - Making the predictions and evaluating the model (Predicting the Test set results)

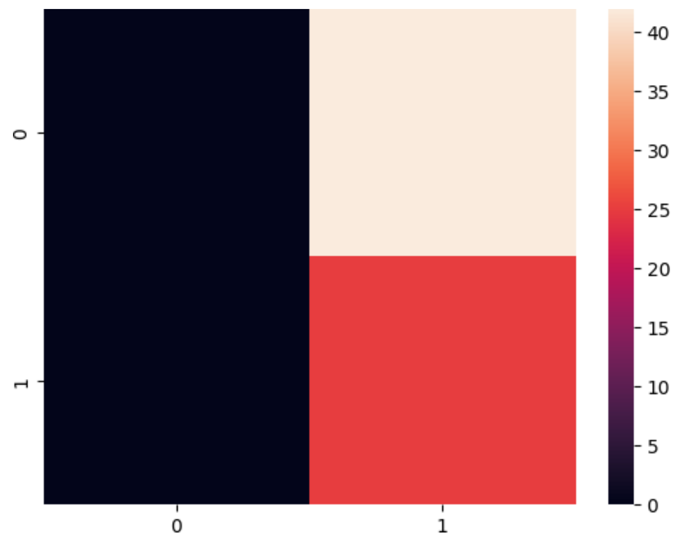
```
y_pred = ann.predict(X_test)
y_pred = (y_pred > 0.5)
print(np.concatenate((y_pred.reshape(len(y_pred),1), y_test.reshape(len(y_test),1)),1))
```

```
3/3 [==============================] - 0s 4ms/step
[[1 1]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 1]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 1]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 0]
 [1 0]
 [1 1]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 1]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 0]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 1]
 [1 1]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 1]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 1]
 [1 0]
 [1 0]
 [1 1]
 [1 1]
```

Making the Confusion Matrix

```
from sklearn.metrics import confusion_matrix, accuracy_score
cm = confusion_matrix(y_test, y_pred)
sns.heatmap(cm)
print(cm)
print("Accuracy: {:.2f}%".format(accuracy_score(y_test, y_pred)*100))
```

```
[[ 0 42]
 [ 0 25]]
Accuracy: 37.31%
```



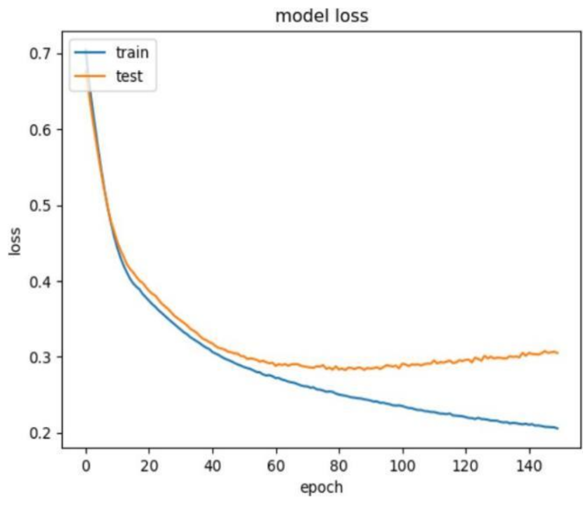
```
print(ann.metrics_names)
```

```
[]
```

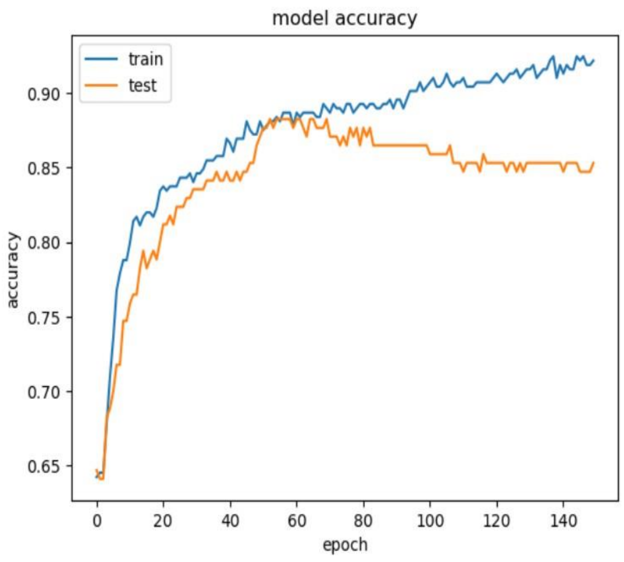
```
print(his.history.keys())
```

```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

```
plt.plot(his.history['loss'])
plt.plot(his.history['val_loss'])
plt.title('model loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```



```
plt.plot(his.history['accuracy'])
plt.plot(his.history['val_accuracy'])
plt.title('model accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['train', 'test'], loc='upper left')
plt.show()
```


model accuracy
train
test
accuracy
epoch