Lab 3 : Binary Classification of Heart Disease of Patients using Deep Neural Network

Dinesh Kumar K

225229108

1. Load the dataset

In [1]:
```
import pandas as pd
```

In [2]:
```
df = pd.read_csv("heart_data.csv")
```

In [3]:
```
df.head()
```

Out[3]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1

In [4]:
```
df.shape
```

Out[4]:
```
(303, 14)
```

In [5]:
```
df.size
```

Out[5]:
```
4242
```

In [6]:
```
df.columns
```

Out[6]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

2. Split the dataset

In [9]:
```
X = df.drop('target', axis=1)
y = df['target']
```

In [10]:
```
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_s
```

In [11]:
```
X_train.shape
```

Out[11]:
```
(242, 13)
```

In [12]:
```
X_test.shape
```

Out[12]:
```
(61, 13)
```

3. Create a neural network

In [13]:
```
#Input size = No. of features in X_train = 13
#No. of neurons/units in the Dense layer = 8, with Relu activation function
#No. of neurons/units in output layer = 1, with sigmoid activation function
```

In [14]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [15]:
```
model = Sequential()
model.add(Dense(8, input_dim=13, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

4. Compile your model

In [16]:
```
from tensorflow import keras
```

In [17]:
```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001)
```

In [18]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
9/9 [==============================] - 2s 12ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 7/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
```

Out[18]:
```
<keras.callbacks.History at 0x28abb93efb0>
```

In [19]:
```
model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.4754 - accuracy:
0.5246
```

Out[19]:
```
[0.4754098355770111, 0.5245901346206665]
```

5. Print the summary of the model

In [20]:
```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense (Dense)               (None, 8)                 112

 dense_1 (Dense)             (None, 1)                 9

=================================================================
Total params: 121
Trainable params: 121
Non-trainable params: 0
_________________________________________________________________
```

6. Train the model

In [21]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=200, batch_size=10, verbose=1)
```

```
y: 0.5496
Epoch 2/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 3/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 4/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 5/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 6/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 7/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 8/200
```

In [22]:
```
model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 4ms/step - loss: 0.4754 - accuracy:
0.5246
```

Out[22]:
```
[0.4754098355770111, 0.5245901346206665]
```

7. Save the trained model

In [23]:
```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_siz
```

```
Epoch 1/100
20/20 [==============================] - 0s 6ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 2/100
20/20 [==============================] - 0s 3ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 3/100
20/20 [==============================] - 0s 3ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 4/100
20/20 [==============================] - 0s 3ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 5/100
20/20 [==============================] - 0s 3ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 6/100
20/20 [==============================] - 0s 3ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 7/100
```

8. Evaluate

In [24]:
```
model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.4754 - accuracy:
0.5246
```

Out[24]:
```
[0.4754098355770111, 0.5245901346206665]
```

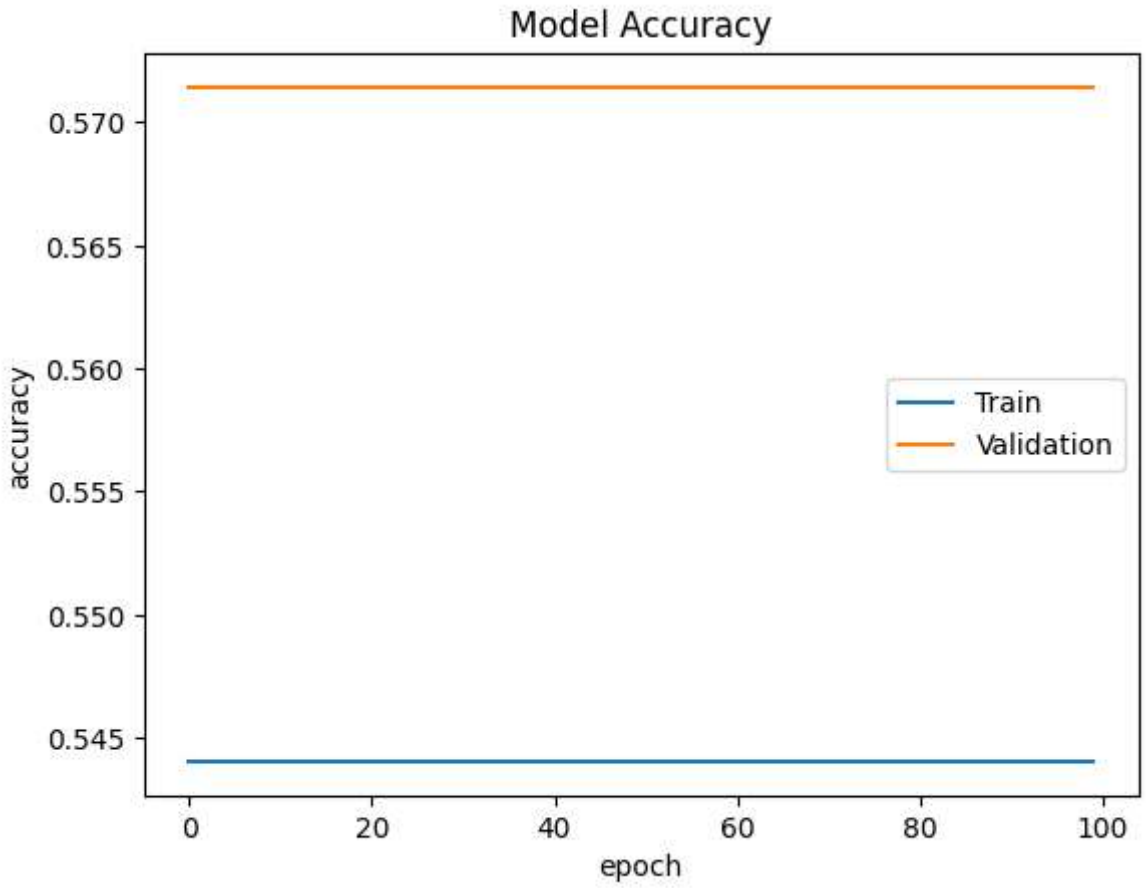
9. Print the model accuracy

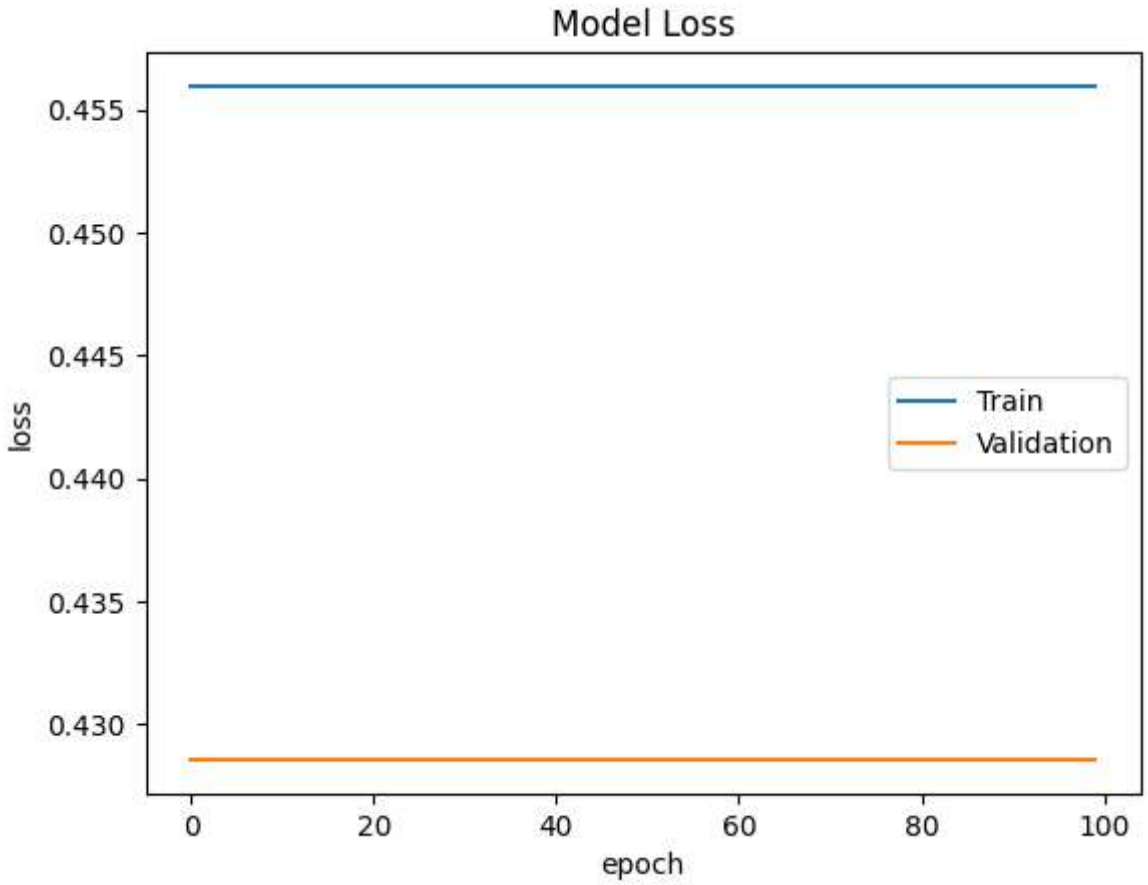
In [28]:
```
history.history.keys()
```

Out[28]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [29]:
```
import matplotlib.pyplot as plt
```

In [30]:
```
plt.plot(history.history['accuracy'])
plt.plot(history.history['val_accuracy'])
plt.title('Model Accuracy')
plt.ylabel('accuracy')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'])
plt.show()
plt.plot(history.history['loss'])
plt.plot(history.history['val_loss'])
plt.title('Model Loss')
plt.ylabel('loss')
plt.xlabel('epoch')
plt.legend(['Train', 'Validation'])
plt.show()
```





10. Do further experiments

In [31]:
```
model1 = Sequential()
model1.add(Dense(16, input_dim=13, activation='relu'))
model1.add(Dense(8, activation='relu'))
model1.add(Dense(1, activation='sigmoid'))
```

In [32]:
```
model1.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model1.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
9/9 [==============================] - 1s 2ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 9/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5496 - accuracy:
0.4504
```

Out[32]:
```
<keras.callbacks.History at 0x28ac0d19cf0>
```

In [33]:
```
model1.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 2ms/step - loss: 0.5246 - accuracy:
0.4754
```

Out[33]:
```
[0.5245901346206665, 0.4754098355770111]
```

In [34]:
```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_si
```

```
Epoch 1/100
7/7 [==============================] - 0s 15ms/step - loss: 0.4560 - accurac
y: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 2/100
7/7 [==============================] - 0s 6ms/step - loss: 0.4560 - accuracy:
0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 3/100
7/7 [==============================] - 0s 6ms/step - loss: 0.4560 - accuracy:
0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 4/100
7/7 [==============================] - 0s 6ms/step - loss: 0.4560 - accuracy:
0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 5/100
7/7 [==============================] - 0s 6ms/step - loss: 0.4560 - accuracy:
0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 6/100
7/7 [==============================] - 0s 5ms/step - loss: 0.4560 - accuracy:
0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 7/100
```

In [35]:
```
model1.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_2 (Dense)             (None, 16)                224

 dense_3 (Dense)             (None, 8)                 136

 dense_4 (Dense)             (None, 1)                 9

=================================================================
Total params: 369
Trainable params: 369
Non-trainable params: 0
_________________________________________________________________
```

In [36]:
```
ls = history1.history
```

In [37]:
```
new = pd.DataFrame.from_dict(ls)
new
```

Out[37]:

	loss	accuracy	val_loss	val_accuracy
0	0.455959	0.544041	0.428571	0.571429
1	0.455959	0.544041	0.428571	0.571429
2	0.455959	0.544041	0.428571	0.571429
3	0.455959	0.544041	0.428571	0.571429
4	0.455959	0.544041	0.428571	0.571429
...	...	...	...	...
95	0.455959	0.544041	0.428571	0.571429
96	0.455959	0.544041	0.428571	0.571429
97	0.455959	0.544041	0.428571	0.571429
98	0.455959	0.544041	0.428571	0.571429
99	0.455959	0.544041	0.428571	0.571429

100 rows × 4 columns

In [38]:
```
model2 = Sequential()
model2.add(Dense(32, input_dim=13, activation='relu'))
model2.add(Dense(16, activation='relu'))
model2.add(Dense(8, activation='relu'))
model2.add(Dense(1, activation='sigmoid'))
```

In [39]:
```
model2.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model2.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
9/9 [==============================] - 1s 2ms/step - loss: 0.5427 - accuracy:
0.4421
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.3125 - accuracy:
0.4876
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2681 - accuracy:
0.5041
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2501 - accuracy:
0.5868
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2387 - accuracy:
0.5950
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2403 - accuracy:
0.6074
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2658 - accuracy:
0.5661
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2274 - accuracy:
0.6240
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2452 - accuracy:
0.6281
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2221 - accuracy:
0.6240
```

Out[39]:
```
<keras.callbacks.History at 0x28ac1fbadd0>
```

In [40]:
```
model2.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.3232 - accuracy:
0.5082
```

Out[40]:
```
[0.3232260048389435, 0.5081967115402222]
```

In [41]:
```
model2.summary()
```

```
Model: "sequential_2"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_5 (Dense)             (None, 32)                448

 dense_6 (Dense)             (None, 16)                528

 dense_7 (Dense)             (None, 8)                 136

 dense_8 (Dense)             (None, 1)                 9

=================================================================
Total params: 1,121
Trainable params: 1,121
Non-trainable params: 0
_________________________________________________________________
```

In [42]:
```
model3 = Sequential()
model3.add(Dense(64, input_dim=13, activation='relu'))
model3.add(Dense(32, activation='relu'))
model3.add(Dense(16, activation='relu'))
model3.add(Dense(8, activation='relu'))
model3.add(Dense(1, activation='sigmoid'))
```

In [43]:
```
model3.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model3.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
9/9 [==============================] - 1s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accuracy:
0.5496
```

Out[43]:
```
<keras.callbacks.History at 0x28ac1f2d900>
```

In [44]:
```
model3.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 4ms/step - loss: 0.4754 - accuracy:
0.5246
```

Out[44]:
```
[0.4754098355770111, 0.5245901346206665]
```

In [45]:
```
model3.summary()
```

```
Model: "sequential_3"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 dense_9 (Dense)             (None, 64)                896

 dense_10 (Dense)            (None, 32)                2080

 dense_11 (Dense)            (None, 16)                528

 dense_12 (Dense)            (None, 8)                 136

 dense_13 (Dense)            (None, 1)                 9

=================================================================
Total params: 3,649
Trainable params: 3,649
Non-trainable params: 0
_________________________________________________________________
```

In [ ]:
```

```