Name:P.Asha Belcilda

Rollno:225229104

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

1. Load the dataset: “heart_data.csv” and explore the features

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 for training and testing (test size = 20%)

In [7]:
```
X = df
y = df.pop('target')
```

In [8]:
```
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, randc
```

In [9]:
```
X_train.shape
```

Out[9]: (242, 13)

In [10]:
```
X_test.shape
```

Out[10]: (61, 13)

3. Create a neural network based on the following requirements

In [11]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [12]:
```
model = Sequential()
model.add(Dense(8, input_dim=13, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

4. Compile your model with learning rate = 0.001, optimizer as ‘RMSprop’, Mean square error loss and metrics as ‘accuracy’.

In [13]:
```
from tensorflow import keras
```

In [14]:
```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001)
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 [==============================] - 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 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 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 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
```

Out[14]:
```
<keras.callbacks.History at 0x1ecca6e7460>
```

In [15]:
```
model.evaluate(X_test, y_test)
```

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

Out[15]:
```
[0.4754098355770111, 0.5245901346206665]
```

5. Print the summary of the model: model.summary()

In [16]:
```
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 for 200 epochs and batch size as 10

In [17]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=200, batch_size=10, verbose=1)
```

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

In [18]:
```
model.evaluate(X_test, y_test)
```

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

Out[18]:
```
[0.4754098355770111, 0.5245901346206665]
```

7. Save the trained model in a variable, such as, history. Also, you can split your training data for validation such as 20% of training data

In [19]:
```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_
```

```
Epoch 1/100
97/97 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 2/100
97/97 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 3/100
97/97 [==============================] - 0s 1ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 4/100
97/97 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 5/100
97/97 [==============================] - 0s 1ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 6/100
97/97 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 7/100
```

8. Evaluate the trained model to predict the probability values for the test data set (ie., xtest and ytest)

In [20]:
```
model.evaluate(X_test, y_test)
```

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

Out[20]:
```
[0.4754098355770111, 0.5245901346206665]
```

9. Print the model accuracy and model loss as below (Use can use the 'history' object we have saved). Sample code is given below.

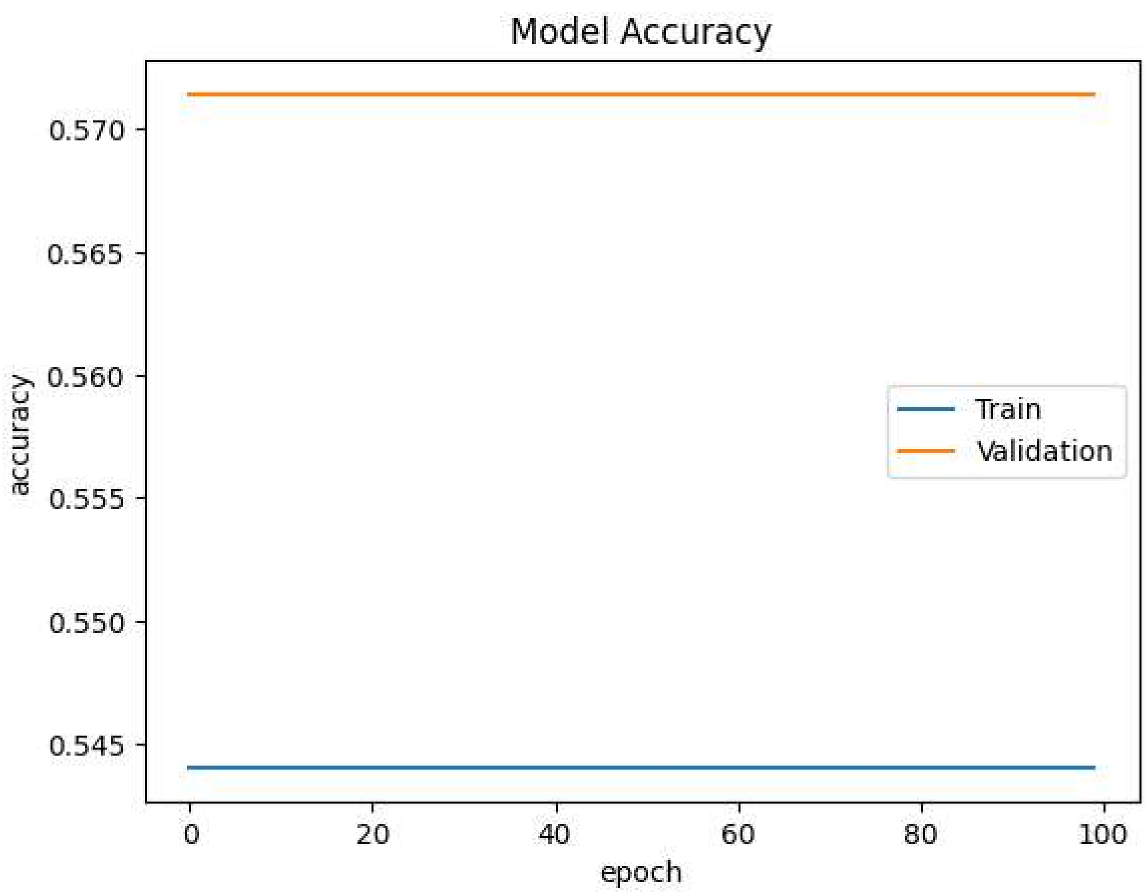
In [22]:
```
history.history.keys()
```

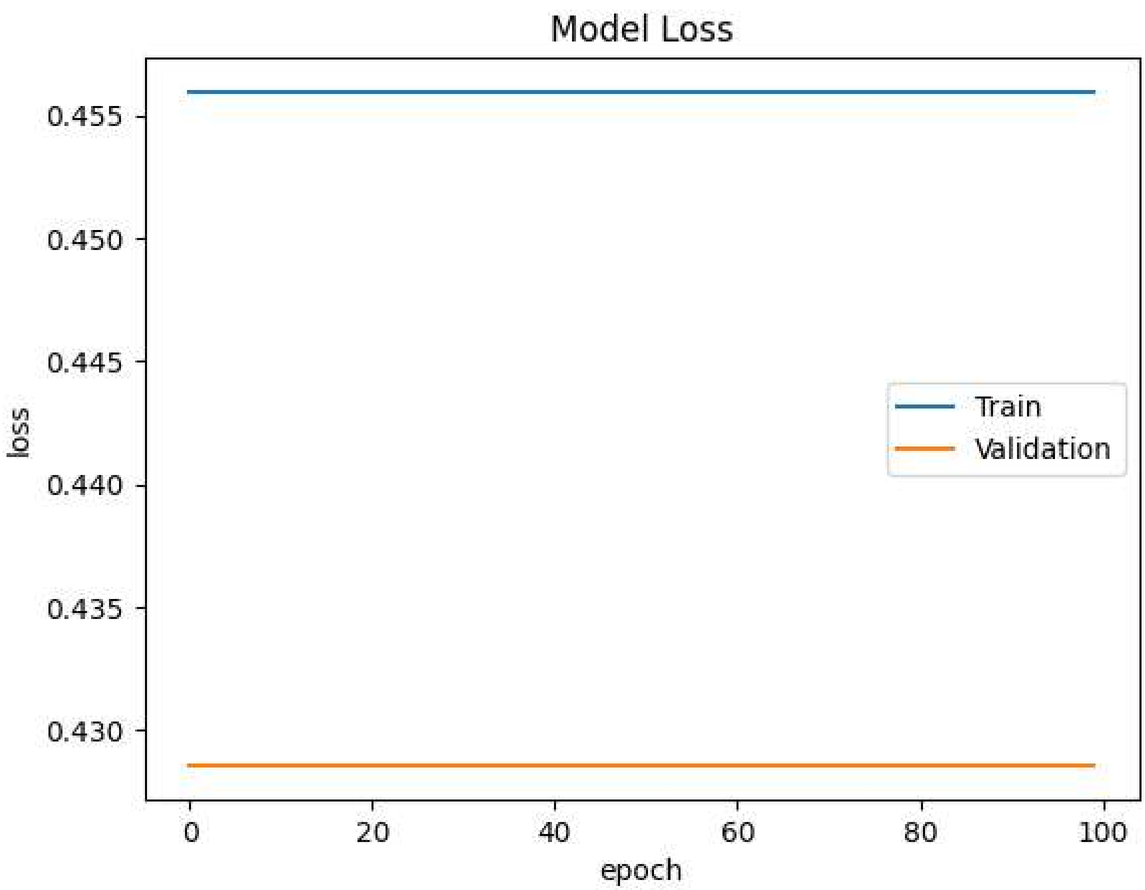
Out[22]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [25]:
```
import matplotlib.pyplot as plt
```

```
Matplotlib is building the font cache; this may take a moment.
```

In [26]:
```
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 [27]:
```
model1 = Sequential()
model1.add(Dense(16, input_dim=13, activation='relu'))
model1.add(Dense(8, activation='relu'))
model1.add(Dense(1, activation='sigmoid'))
```

In [28]:
```
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: 0.4190 - accuracy:
0.5207
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4139 - accuracy:
0.4793
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3729 - accuracy:
0.5165
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3698 - accuracy:
0.5331
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3327 - accuracy:
0.5496
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3742 - accuracy:
0.4959
Epoch 7/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3223 - accuracy:
0.5702
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3172 - accuracy:
0.5785
Epoch 9/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3447 - accuracy:
0.5455
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3131 - accuracy:
0.5785
```

Out[28]:
```
<keras.callbacks.History at 0x1ecd0ca8040>
```

In [29]:
```
model1.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.4532 - accuracy:
0.5246
```

Out[29]:
```
[0.4531889259815216, 0.5245901346206665]
```

In [30]:
```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch
```

```
Epoch 1/100
65/65 [==============================] - 0s 3ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 2/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 3/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 4/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 5/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 6/100
65/65 [==============================] - 0s 2ms/step - loss: 0.4560 - accu
racy: 0.5440 - val_loss: 0.4286 - val_accuracy: 0.5714
Epoch 7/100
```

In [31]:
```
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 [32]:
```
ls = history1.history
new = pd.DataFrame.from_dict(ls)
new
```

Out[32]:

	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 [33]:
```
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 [34]:
```
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 1ms/step - loss: 0.4736 - accuracy:
0.5000
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3240 - accuracy:
0.6116
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2572 - accuracy:
0.6157
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2685 - accuracy:
0.5785
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2583 - accuracy:
0.6198
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2586 - accuracy:
0.6033
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2534 - accuracy:
0.6322
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2666 - accuracy:
0.6198
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2301 - accuracy:
0.6364
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2305 - accuracy:
0.6157
```

Out[34]:
```
<keras.callbacks.History at 0x1ecd1134910>
```

In [35]:
```
model2.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.3780 - accuracy:
0.4754
```

Out[35]:
```
[0.3780178129673004, 0.4754098355770111]
```

In [36]:
```
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 [37]:
```
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 [38]:
```
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: 0.3389 - accuracy:
0.6033
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3138 - accuracy:
0.6281
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2996 - accuracy:
0.6322
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2598 - accuracy:
0.6529
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2316 - accuracy:
0.6364
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2276 - accuracy:
0.6364
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2202 - accuracy:
0.6322
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2360 - accuracy:
0.5579
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2829 - accuracy:
0.6653
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2180 - accuracy:
0.6405
```

Out[38]:
```
<keras.callbacks.History at 0x1ecd0f8e710>
```

In [39]:
```
model3.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.2285 - accuracy:
0.5082
```

Out[39]:
```
[0.22848190367221832, 0.5081967115402222]
```

In [40]:
```
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
_________________________________________________________________
```