PRINCY A

REG NO 225229128

PDL Lab 3_Binary Classification of Heart Disease of Patients using DNN

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	targe
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	

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', 'thalac
h',
       '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, random_state=42)
```

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

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 [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

In [13]:

```
from tensorflow import keras
```

In [14]:

```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001)
```

In [15]:

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

Out[15]:

```
<keras.callbacks.History at 0x2dca0f91450>
```

In [16]:

```
model.evaluate(X_test, y_test)
```

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

Out[16]:

```
[0.4754098355770111, 0.5245901346206665]
```

5. Print the summary of the model:

In [17]:

```
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 [18]:

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

In [19]:

```
model.evaluate(X_test, y_test)
```

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

Out[19]:

```
[0.4754098355770111, 0.5245901346206665]
```

7.Save the trained model

In [20]:

```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_size=2)
```

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

8.Evaluate

In [21]:

```
model.evaluate(X_test, y_test)
```

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

Out[21]:

```
[0.4754098355770111, 0.5245901346206665]
```

9.Print the model accuracy

In [22]:

```
history.history.keys()
```

Out[22]:

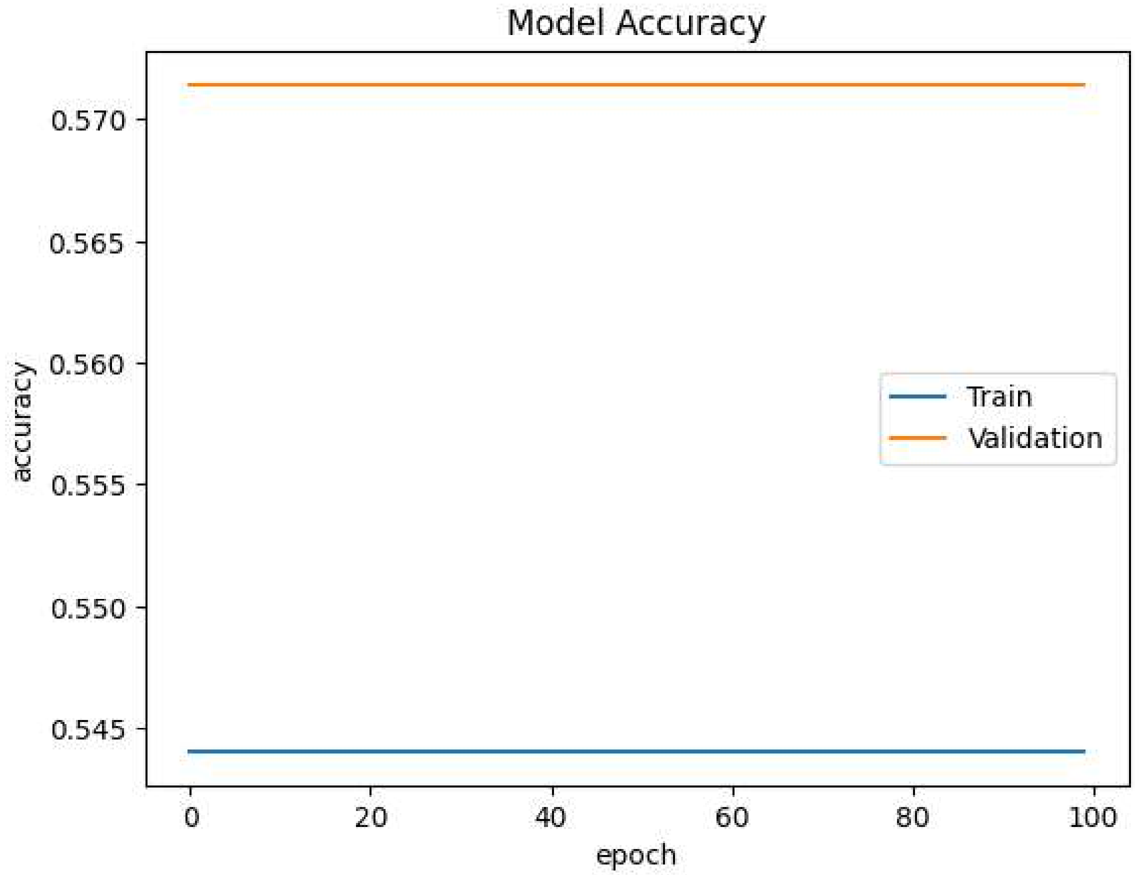
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

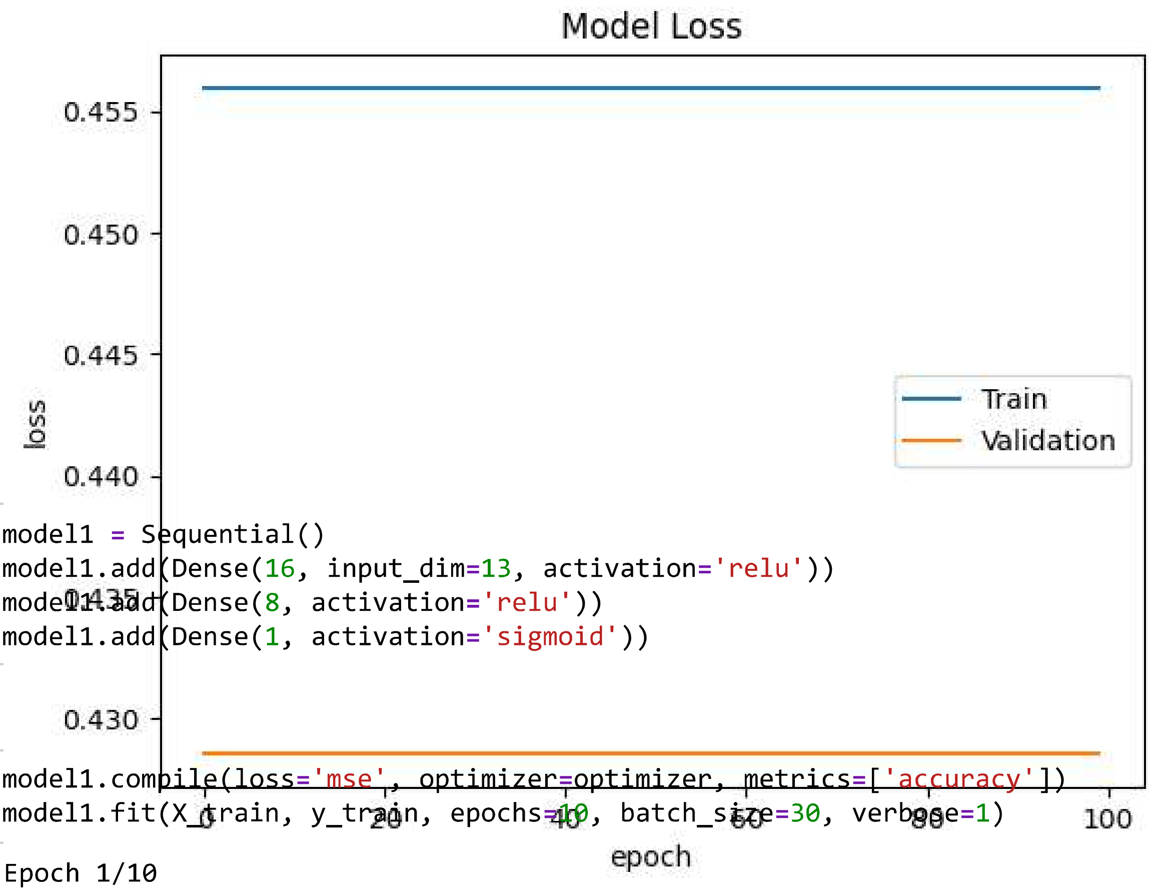
In [23]:

```
import matplotlib.pyplot as plt
```

In [25]:

```
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()
```





```
model1 = Sequential()
model1.add(Dense(16, input_dim=13, activation='relu'))
model1.add(Dense(8, activation='relu'))
model1.add(Dense(1, activation='sigmoid'))
```

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

```
Out[27]:
```

```
<keras.callbacks.History at 0x2dca5e8efb0>
```

In [28]:

```
model1.evaluate(X_test, y_test)
```

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

Out[28]:

```
[0.4754098355770111, 0.5245901346206665]
```

In [31]:

```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_size=3)
```

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

In [32]:

```
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 [33]:

```
ls = history1.history
```

In [34]:

```
new = pd.DataFrame.from_dict(ls)
new
```

Out[34]:

	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 [35]:

```
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 [ ]:

```

```