Binary Classification of Heart Disease of Patients using Deep Neural Network

Import Dataset

In [1]:
```
!pip install scikit-learn
```

```
     ---------                              2.1/9.2 MB 2.1 MB/s eta 0:00:04
     ---------                              2.2/9.2 MB 2.1 MB/s eta 0:00:04
     ---------                              2.3/9.2 MB 2.1 MB/s eta 0:00:04
     ----------                             2.5/9.2 MB 2.1 MB/s eta 0:00:04
     -----------                            2.6/9.2 MB 2.1 MB/s eta 0:00:04
     -----------                            2.7/9.2 MB 2.1 MB/s eta 0:00:04
     ------------                           2.8/9.2 MB 2.1 MB/s eta 0:00:04
     ------------                           2.9/9.2 MB 2.2 MB/s eta 0:00:03
     -------------                          3.0/9.2 MB 2.2 MB/s eta 0:00:03
     -------------                          3.1/9.2 MB 2.2 MB/s eta 0:00:03
     --------------                         3.2/9.2 MB 2.2 MB/s eta 0:00:03
     --------------                         3.4/9.2 MB 2.2 MB/s eta 0:00:03
     --------------                         3.4/9.2 MB 2.2 MB/s eta 0:00:03
     ---------------                        3.6/9.2 MB 2.2 MB/s eta 0:00:03
     ---------------                        3.7/9.2 MB 2.2 MB/s eta 0:00:03
     ----------------                       3.8/9.2 MB 2.2 MB/s eta 0:00:03
     ----------------                       3.9/9.2 MB 2.2 MB/s eta 0:00:03
     -----------------                      3.9/9.2 MB 2.2 MB/s eta 0:00:03
     -----------------                      4.0/9.2 MB 2.2 MB/s eta 0:00:03
     -----------------                      4.0/9.2 MB 2.2 MB/s eta 0:00:03
```

In [2]:
```
import pandas as pd
```

In [3]:
```
df=pd.read_csv("heart_data.csv")
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.columns
```

Out[5]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalach',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

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

Out[6]:
```
4242
```

Split the Dataset into Training and Testing

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_sta
```

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

Out[9]:
```
(242, 13)
```

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

Out[10]:
```
(61, 13)
```

Create Neural Networks

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'))
```

Compile 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 [==============================] - 3s 12ms/step - loss: 0.3713 - accuracy:
0.6074
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3643 - accuracy:
0.6157
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3609 - accuracy:
0.6240
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3671 - accuracy:
0.6116
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3692 - accuracy:
0.6033
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3666 - accuracy:
0.6116
Epoch 7/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3647 - accuracy:
0.6033
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3643 - accuracy:
0.6074
Epoch 9/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3603 - accuracy:
0.6116
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3633 - accuracy:
0.6116
```

Out[15]:
```
<keras.callbacks.History at 0x20059495480>
```

In [16]:
```
model.evaluate(x_test,y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.2826 - accuracy:
0.6885
```

Out[16]:
```
[0.2826370596885681, 0.688524603843689]
```

Print the Model Summary

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
_________________________________________________________________
```

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: 0.3584 - accurac
y: 0.6240
Epoch 2/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3736 - accurac
y: 0.5950
Epoch 3/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3738 - accurac
y: 0.6033
Epoch 4/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3717 - accurac
y: 0.6074
Epoch 5/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3724 - accurac
y: 0.5950
Epoch 6/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3601 - accurac
y: 0.6116
Epoch 7/200
```

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

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1279 - accuracy:
0.8525
```

Out[19]:
```
[0.12794315814971924, 0.8524590134620667]
```

Save the Model and Split the data for Validation

In [20]:
```
history = model.fit(x_train, y_train, validation_split=0.2, epochs=100,  batch_si
```

```
y: 0.8497 - val_loss: 0.1310 - val_accuracy: 0.8163
Epoch 4/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1194 - accurac
y: 0.8446 - val_loss: 0.1393 - val_accuracy: 0.8163
Epoch 5/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1279 - accurac
y: 0.8290 - val_loss: 0.1183 - val_accuracy: 0.8367
Epoch 6/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1235 - accurac
y: 0.8342 - val_loss: 0.1066 - val_accuracy: 0.8571
Epoch 7/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1285 - accurac
y: 0.8394 - val_loss: 0.1028 - val_accuracy: 0.8367
Epoch 8/100
20/20 [==============================] - 0s 5ms/step - loss: 0.1155 - accurac
y: 0.8394 - val_loss: 0.1272 - val_accuracy: 0.8163
Epoch 9/100
20/20 [==============================] - 0s 4ms/step - loss: 0.1275 - accurac
y: 0.8031 - val_loss: 0.1211 - val_accuracy: 0.8163
Epoch 10/100
```

In [21]:
```
model.evaluate(x_test,y_test)
```

```
2/2 [==============================] - 0s 2ms/step - loss: 0.1432 - accuracy:
0.8361
```

Out[21]:
```
[0.143158420920372, 0.8360655903816223]
```

Print the model accuracy

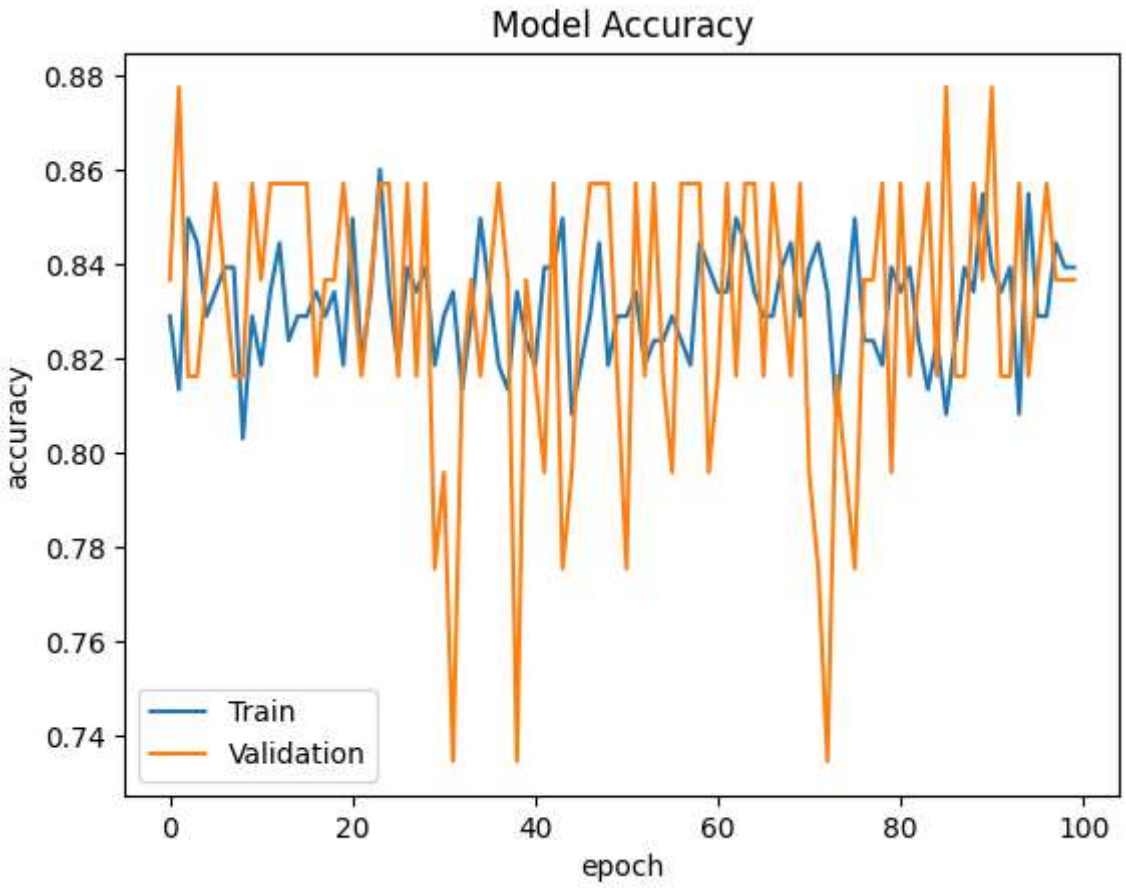
In [22]:
```
history.history.keys()
```

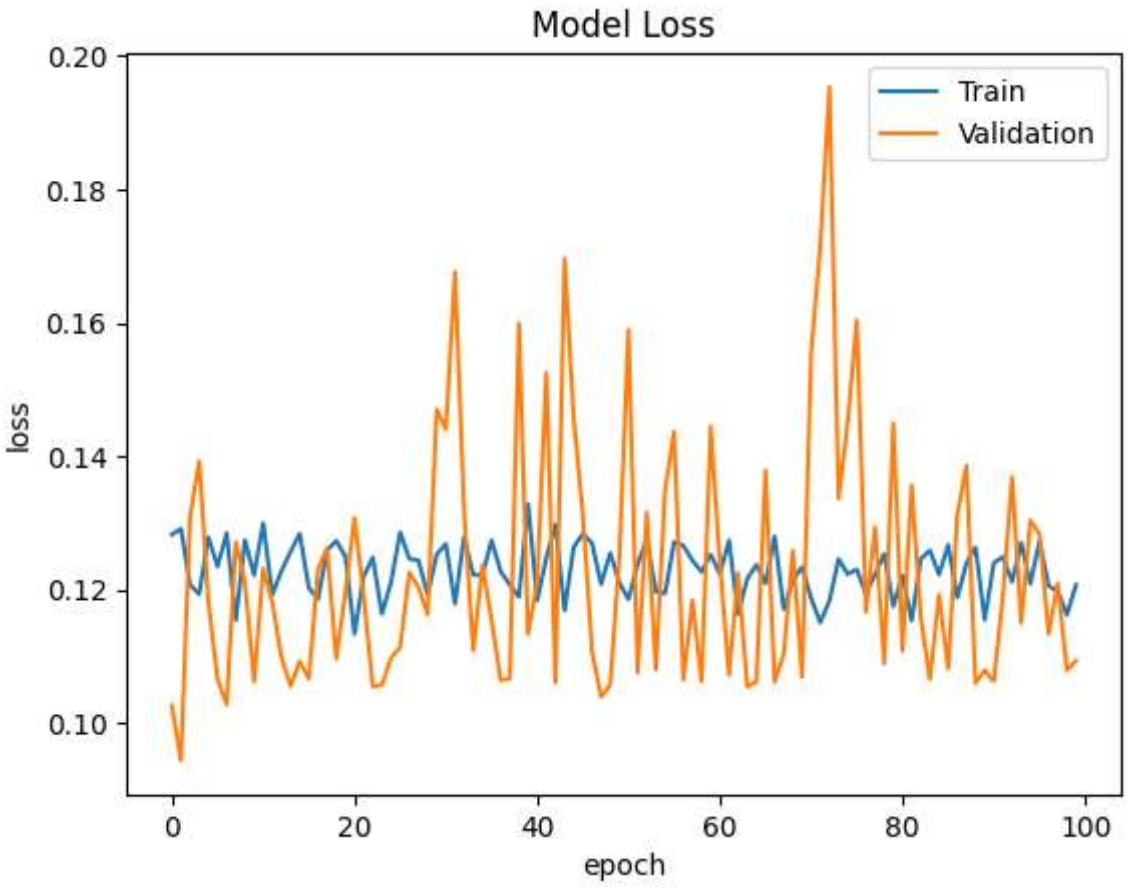
Out[22]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [23]:
```
import matplotlib.pyplot as plt
```

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

In [24]:
```
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()
```





Do further experiments

In [25]:
```
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: 0.4587 - accuracy:
0.5413
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4587 - accuracy:
0.5413
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4587 - accuracy:
0.5413
```

Out[28]:
```
<keras.callbacks.History at 0x2005e712b90>
```

In [30]:
```
model1.evaluate(x_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.4426 - accuracy:
0.5574
```

Out[30]:
```
[0.44262295961380005, 0.5573770403862]
```

In [32]:
```
history1 = model.fit(x_train, y_train, validation_split=0.2, epochs=100, batch_si
```

```
Epoch 1/100
10/10 [==============================] - 0s 10ms/step - loss: 0.1230 - accura
cy: 0.8342 - val_loss: 0.1074 - val_accuracy: 0.8776
Epoch 2/100
10/10 [==============================] - 0s 5ms/step - loss: 0.1151 - accurac
y: 0.8446 - val_loss: 0.1190 - val_accuracy: 0.8571
Epoch 3/100
10/10 [==============================] - 0s 5ms/step - loss: 0.1243 - accurac
y: 0.8290 - val_loss: 0.1185 - val_accuracy: 0.8163
Epoch 4/100
10/10 [==============================] - 0s 4ms/step - loss: 0.1152 - accurac
y: 0.8394 - val_loss: 0.1163 - val_accuracy: 0.8571
Epoch 5/100
10/10 [==============================] - 0s 5ms/step - loss: 0.1244 - accurac
y: 0.8187 - val_loss: 0.1069 - val_accuracy: 0.8776
Epoch 6/100
10/10 [==============================] - 0s 4ms/step - loss: 0.1173 - accurac
y: 0.8446 - val_loss: 0.1076 - val_accuracy: 0.8776
Epoch 7/100
```

In [33]:
```
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 [34]:
```
ls = history1.history
```

In [35]:
```
new = pd.DataFrame.from_dict(ls)
new
```

Out[35]:

	loss	accuracy	val_loss	val_accuracy
0	0.122989	0.834197	0.107391	0.877551
1	0.115098	0.844560	0.118961	0.857143
2	0.124288	0.829016	0.118453	0.816327
3	0.115204	0.839378	0.116296	0.857143
4	0.124392	0.818653	0.106862	0.877551
...	...	...	...	...
95	0.117606	0.829016	0.106387	0.857143
96	0.119126	0.865285	0.124048	0.816327
97	0.122252	0.813471	0.117347	0.857143
98	0.113644	0.829016	0.107408	0.836735
99	0.115921	0.839378	0.106518	0.877551

100 rows × 4 columns

In [36]:
```
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 [38]:
```
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: 0.4433 - accuracy:
0.4669
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.3428 - accuracy:
0.5372
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2955 - accuracy:
0.5455
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.3113 - accuracy:
0.5661
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2619 - accuracy:
0.5992
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2694 - accuracy:
0.6116
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2646 - accuracy:
0.5785
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2771 - accuracy:
0.6157
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2367 - accuracy:
0.6157
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2324 - accuracy:
0.6405
```

Out[38]:
```
<keras.callbacks.History at 0x2005fdfdf60>
```

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

```
2/2 [==============================] - 0s 4ms/step - loss: 0.2237 - accuracy:
0.6721
```

Out[40]:
```
[0.22372157871723175, 0.6721311211585999]
```

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 [44]:
```
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: 0.4796 - accuracy:
0.4876
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.3466 - accuracy:
0.5661
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2638 - accuracy:
0.5702
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2510 - accuracy:
0.5744
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2530 - accuracy:
0.5868
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2355 - accuracy:
0.6240
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2056 - accuracy:
0.7025
Epoch 8/10
9/9 [==============================] - 0s 3ms/step - loss: 0.2299 - accuracy:
0.6488
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2102 - accuracy:
0.6736
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2239 - accuracy:
0.6529
```

Out[44]:
```
<keras.callbacks.History at 0x2005e302320>
```

In [46]:
```
model3.evaluate(x_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.2123 - accuracy:
0.6885
```

Out[46]:
```
[0.2122710943222046, 0.688524603843689]
```

In [47]:
```
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 [ ]:
```

```