## SURIYA S

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

In [2]:
```
data=pd.read_csv('heart_data.csv')
data.head()
```

Out[2]:

| | 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 |

## part 2

In [3]:
```
X = data
y = data.pop('target')
```

In [4]:
```
X
```

Out[4]:

| | age | sex | cp | trestbps | chol | fbs | restecg | thalach | exang | oldpeak | slope | ca | thal |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| **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 |
| **...** | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... | ... |
| **298** | 57 | 0 | 0 | 140 | 241 | 0 | 1 | 123 | 1 | 0.2 | 1 | 0 | 3 |
| **299** | 45 | 1 | 3 | 110 | 264 | 0 | 1 | 132 | 0 | 1.2 | 1 | 0 | 3 |
| **300** | 68 | 1 | 0 | 144 | 193 | 1 | 1 | 141 | 0 | 3.4 | 1 | 2 | 3 |
| **301** | 57 | 1 | 0 | 130 | 131 | 0 | 1 | 115 | 1 | 1.2 | 1 | 1 | 3 |
| **302** | 57 | 0 | 1 | 130 | 236 | 0 | 0 | 174 | 0 | 0.0 | 1 | 1 | 2 |

303 rows × 13 columns

In [ ]:
```

```

In [5]:
```
from sklearn.model_selection import train_test_split
```

In [6]:
```
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.33, random_
```

In [7]:
```
X_train.shape
```

Out[7]:
```
(203, 13)
```

In [8]:
```
X_test.shape
```

Out[8]:
```
(100, 13)
```

## part 3

In [9]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [10]:
```
model = Sequential()
model.add(Dense(8, input_dim=13, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

## part 4

In [11]:
```
from tensorflow import keras
```

In [12]:
```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001)
```

In [13]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
7/7 [==============================] - 1s 2ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 2/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 3/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 4/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 5/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 6/10
7/7 [==============================] - 0s 1ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 7/10
7/7 [==============================] - 0s 1ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 8/10
7/7 [==============================] - 0s 1ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 9/10
7/7 [==============================] - 0s 1ms/step - loss: 0.4729 - accuracy:
0.5271
Epoch 10/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4729 - accuracy:
0.5271
```

Out[13]:
```
<keras.callbacks.History at 0x21433922380>
```

In [14]:
```
model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.4200 - accuracy:
0.5800
```

Out[14]:
```
[0.41999998688697815, 0.5799999833106995]
```

## part 5

In [15]:
```
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
_________________________________________________________________
```

## part 6

In [16]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=200, batch_size=10, verbose=1)
```

```
y: 0.5271
Epoch 20/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
Epoch 21/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
Epoch 22/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
Epoch 23/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
Epoch 24/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
Epoch 25/200
21/21 [==============================] - 0s 1ms/step - loss: 0.4729 - accurac
y: 0.5271
```

In [17]:
```
model.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.4200 - accuracy:
0.5800
```

Out[17]:
```
[0.41999998688697815, 0.5799999833106995]
```

## part 7

In [18]:
```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_siz
```

```
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 10/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 11/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 12/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 13/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 14/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 15/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 16/100
```

## part 8

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

```
4/4 [==============================] - 0s 2ms/step - loss: 0.4200 - accuracy:
0.5800
```

Out[19]:
```
[0.41999998688697815, 0.5799999833106995]
```

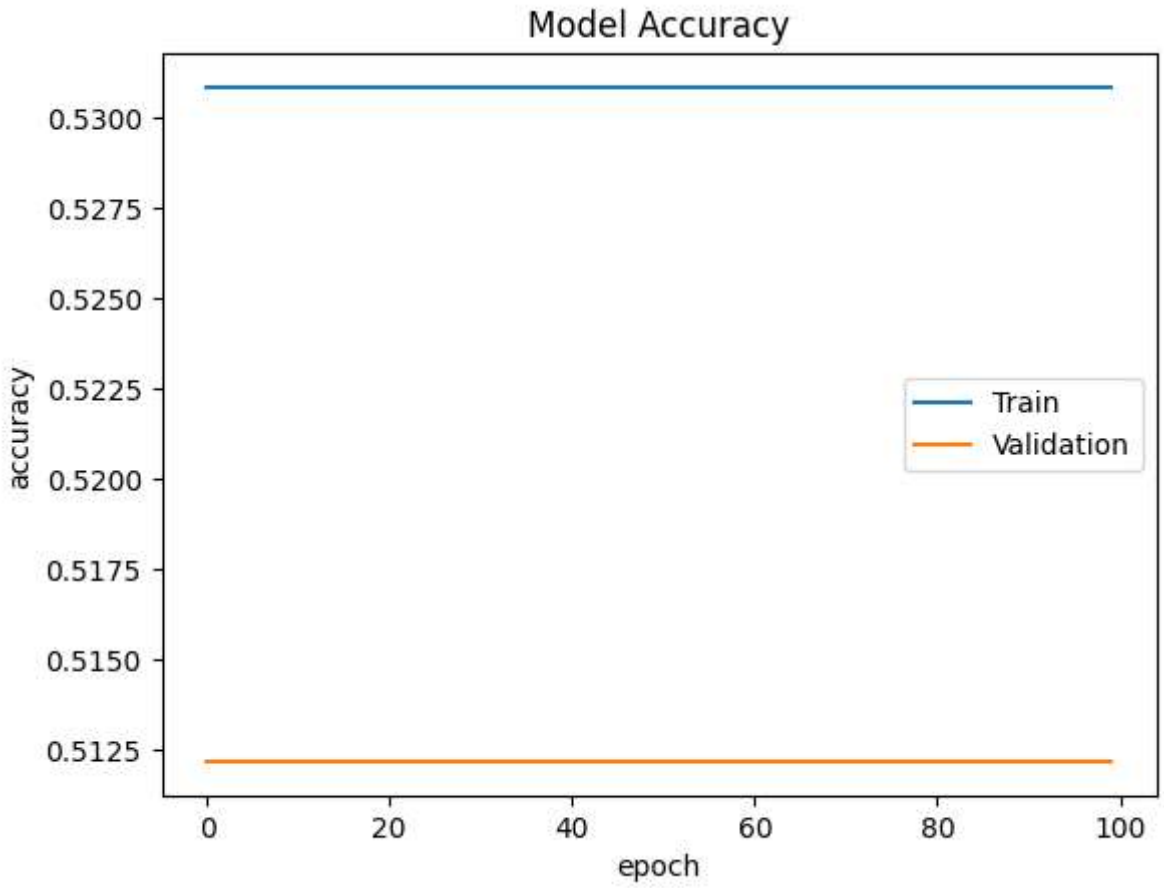
## part 9

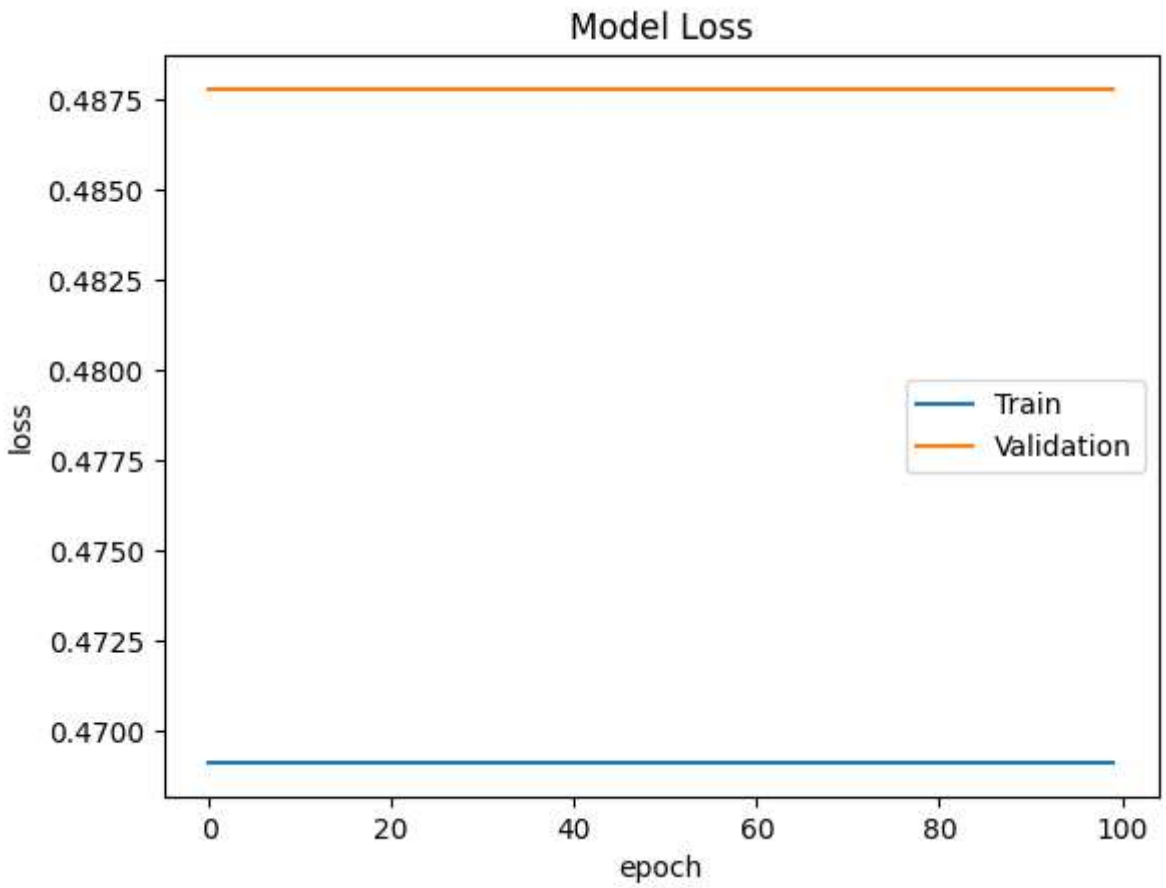
In [20]:
```
history.history.keys()
```

Out[20]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [21]:
```
import matplotlib.pyplot as plt
```

In [22]:
```
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()
```






## part 10

In [23]:
```
model1 = Sequential()

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

In [24]:
```
model1.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model1.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
7/7 [==============================] - 1s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 2/10
7/7 [==============================] - 0s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 3/10
7/7 [==============================] - 0s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 4/10
7/7 [==============================] - 0s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 5/10
7/7 [==============================] - 0s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 6/10
7/7 [==============================] - 0s 1ms/step - loss: 0.5239 - accuracy:
0.4680
Epoch 7/10
7/7 [==============================] - 0s 1ms/step - loss: 0.4925 - accuracy:
0.4039
Epoch 8/10
7/7 [==============================] - 0s 1ms/step - loss: 0.3548 - accuracy:
0.5074
Epoch 9/10
7/7 [==============================] - 0s 1ms/step - loss: 0.2705 - accuracy:
0.5813
Epoch 10/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2505 - accuracy:
0.5862
```

Out[24]:
```
<keras.callbacks.History at 0x214389a1630>
```

In [25]:
```
model1.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.2092 - accuracy:
0.6400
```

Out[25]:
```
[0.20918434858322144, 0.6399999856948853]
```

In [26]:
```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_si
```

```
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 9/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 10/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 11/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 12/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 13/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 14/100
17/17 [==============================] - 0s 3ms/step - loss: 0.4691 - accurac
y: 0.5309 - val_loss: 0.4878 - val_accuracy: 0.5122
Epoch 15/100
```

In [27]:
```
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 [30]:
```
ls = history1.history
```

In [29]:
```
new = pd.DataFrame.from_dict(ls)
new
```

Out[29]:

| | loss | accuracy | val_loss | val_accuracy |
|---|---|---|---|---|
| **0** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **1** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **2** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **3** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **4** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **...** | ... | ... | ... | ... |
| **95** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **96** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **97** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **98** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |
| **99** | 0.469136 | 0.530864 | 0.487805 | 0.512195 |

100 rows × 4 columns

In [31]:
```
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 [32]:
```
model2.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model2.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
7/7 [==============================] - 1s 2ms/step - loss: 0.5271 - accuracy:
0.4729
Epoch 2/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3904 - accuracy:
0.5567
Epoch 3/10
7/7 [==============================] - 0s 1ms/step - loss: 0.3305 - accuracy:
0.5862
Epoch 4/10
7/7 [==============================] - 0s 1ms/step - loss: 0.2764 - accuracy:
0.6059
Epoch 5/10
7/7 [==============================] - 0s 1ms/step - loss: 0.2707 - accuracy:
0.6256
Epoch 6/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2521 - accuracy:
0.6207
Epoch 7/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2707 - accuracy:
0.6158
Epoch 8/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2417 - accuracy:
0.6798
Epoch 9/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2416 - accuracy:
0.6502
Epoch 10/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2411 - accuracy:
0.6207
```

Out[32]:
```
<keras.callbacks.History at 0x21439b2fa60>
```

In [34]:
```
model2.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.1724 - accuracy:
0.7300
```

Out[34]:
```
[0.17240825295448303, 0.7300000190734863]
```

In [35]:
```
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 [36]:
```
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 [37]:
```
model3.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model3.fit(X_train, y_train, epochs=10, batch_size=30, verbose=1)
```

```
Epoch 1/10
7/7 [==============================] - 1s 2ms/step - loss: 0.4457 - accuracy:
0.4926
Epoch 2/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4446 - accuracy:
0.5025
Epoch 3/10
7/7 [==============================] - 0s 2ms/step - loss: 0.4264 - accuracy:
0.5123
Epoch 4/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3367 - accuracy:
0.5567
Epoch 5/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3232 - accuracy:
0.5764
Epoch 6/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3048 - accuracy:
0.6059
Epoch 7/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3274 - accuracy:
0.5813
Epoch 8/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3085 - accuracy:
0.5714
Epoch 9/10
7/7 [==============================] - 0s 2ms/step - loss: 0.3222 - accuracy:
0.6010
Epoch 10/10
7/7 [==============================] - 0s 2ms/step - loss: 0.2586 - accuracy:
0.6355
```

Out[37]:
```
<keras.callbacks.History at 0x2143ae7ceb0>
```

In [38]:
```
model3.evaluate(X_test, y_test)
```

```
4/4 [==============================] - 0s 2ms/step - loss: 0.1892 - accuracy:
0.7200
```

Out[38]:
```
[0.18918751180171967, 0.7200000286102295]
```

In [39]:
```
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 [ ]: