PDL Lab3:. Binary Classification of Heart Disease of Patients using Deep Neural Network

SWETHA JENIFER 225229142

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.info
```

Out[6]:
```
<bound method DataFrame.info of      age  sex  cp  trestbps  chol  fbs  restecg
thalach  exang  oldpeak  \
0     63    1   3       145   233    1        0      150      0      2.3
1     37    1   2       130   250    0        1      187      0      3.5
2     41    0   1       130   204    0        0      172      0      1.4
3     56    1   1       120   236    0        1      178      0      0.8
4     57    0   0       120   354    0        1      163      1      0.6
..   ...  ...  ..       ...   ...  ...      ...      ...    ...      ...
298   57    0   0       140   241    0        1      123      1      0.2
299   45    1   3       110   264    0        1      132      0      1.2
300   68    1   0       144   193    1        1      141      0      3.4
301   57    1   0       130   131    0        1      115      1      1.2
302   57    0   1       130   236    0        0      174      0      0.0

     slope  ca  thal  target
0        0   0     1       1
1        0   0     2       1
2        2   0     2       1
3        2   0     2       1
4        2   0     2       1
..     ...  ..   ...     ...
298      1   0     3       0
299      1   0     3       0
300      1   2     3       0
301      1   1     3       0
302      1   1     2       0

[303 rows x 14 columns]>
```

In [7]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 303 entries, 0 to 302
Data columns (total 14 columns):
 #   Column    Non-Null Count  Dtype  
---  ------    --------------  -----  
 0   age       303 non-null    int64  
 1   sex       303 non-null    int64  
 2   cp        303 non-null    int64  
 3   trestbps  303 non-null    int64  
 4   chol      303 non-null    int64  
 5   fbs       303 non-null    int64  
 6   restecg   303 non-null    int64  
 7   thalach   303 non-null    int64  
 8   exang     303 non-null    int64  
 9   oldpeak   303 non-null    float64
 10  slope     303 non-null    int64  
 11  ca        303 non-null    int64  
 12  thal      303 non-null    int64  
 13  target    303 non-null    int64  
dtypes: float64(1), int64(13)
memory usage: 33.3 KB
```

2. Split the dataset

In [8]:
```
X=df[['age','sex','cp','trestbps','chol','fbs','restecg','thalach','exang','oldpe
```

In [9]:
```
y=df[['target']]
```

In [11]:
```
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 [12]:
```
X_train.shape
```

Out[12]:
```
(242, 13)
```

In [13]:
```
X_test.shape
```

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

3. Create a neural network based on the following requirements

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

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 [==============================] - 3s 23ms/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 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 5/10
9/9 [==============================] - 0s 1ms/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 1ms/step - loss: 0.4504 - accuracy:
0.5496
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accuracy:
0.5496
```

Out[18]:
```
<keras.callbacks.History at 0x22d50f8d270>
```

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: model.summary()

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 for 200 epochs and batch size as 10

In [21]:
```
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 - accurac
y: 0.5496
Epoch 2/200
25/25 [==============================] - 0s 1ms/step - loss: 0.4504 - accurac
y: 0.5496
Epoch 3/200
25/25 [==============================] - 0s 11ms/step - loss: 0.4504 - accura
cy: 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
```

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 a variable, such as, history. Also, you can split your training data for validation such as 20% of training data

In [24]:
```
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 the trained model to predict the probability values for the test data set (ie., xtest and ytest)

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

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

Out[25]:
```
[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.

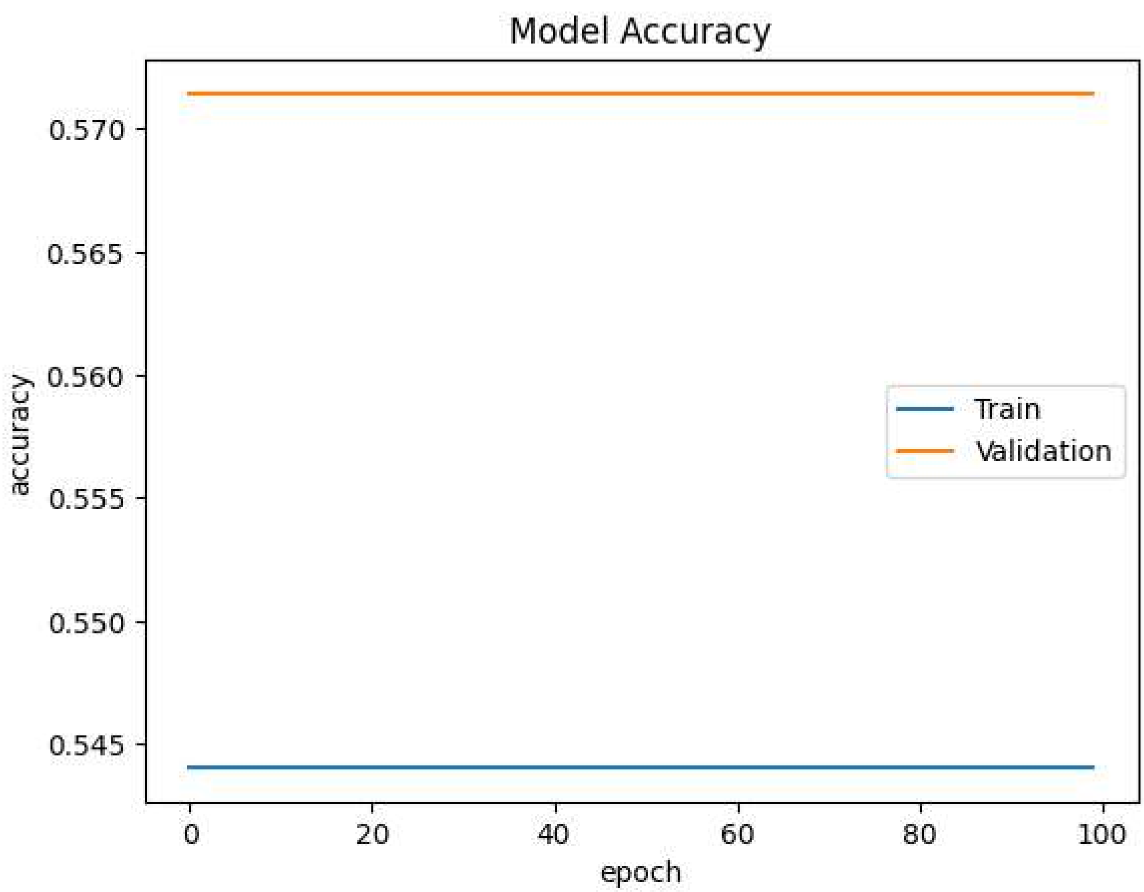
In [26]:
```
history.history.keys()
```

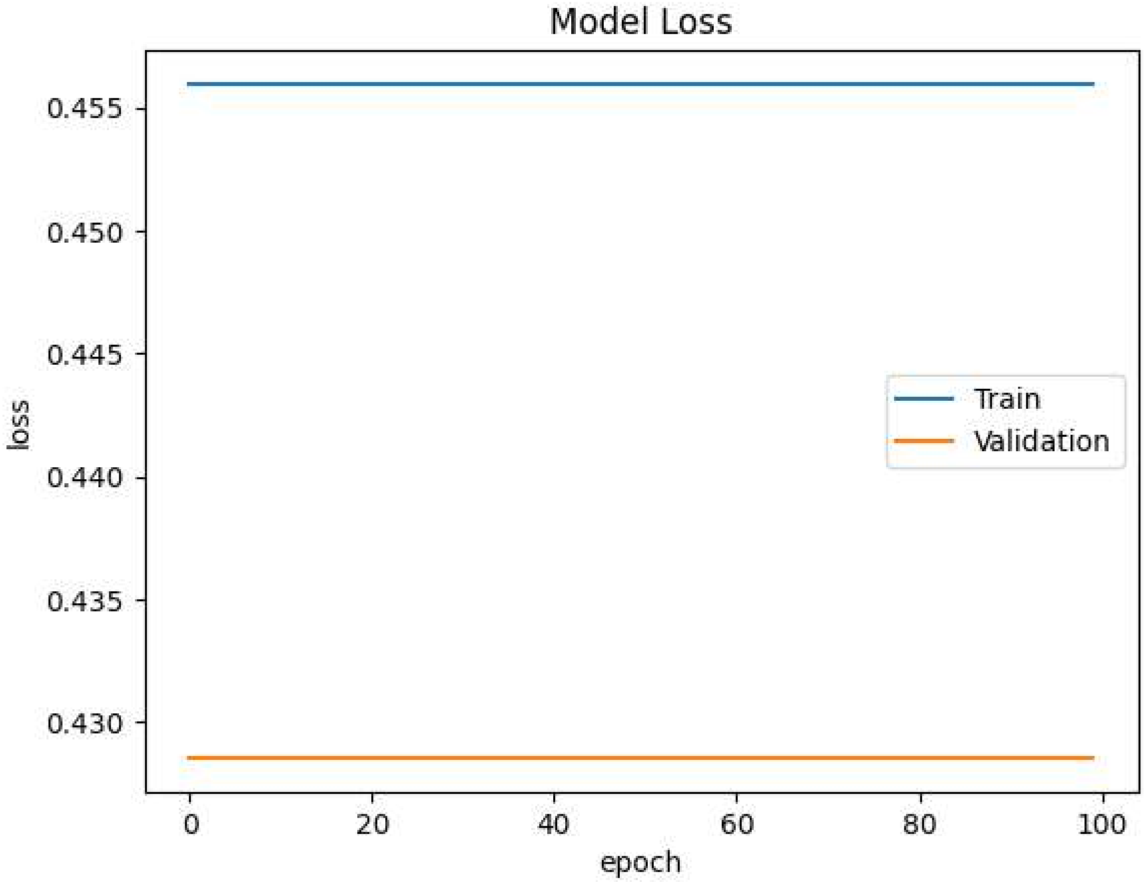
Out[26]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [28]:
```
import matplotlib.pyplot as plt
```

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

In [29]:
```
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 [30]:
```
model1 = Sequential()

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

In [31]:
```
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: 0.5111 - accuracy:
0.4504
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2543 - accuracy:
0.6198
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2492 - accuracy:
0.6281
Epoch 4/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2566 - accuracy:
0.6446
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2464 - accuracy:
0.6240
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2396 - accuracy:
0.6322
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2450 - accuracy:
0.6198
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2513 - accuracy:
0.6157
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2375 - accuracy:
0.6488
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2347 - accuracy:
0.6446
```

Out[31]:
```
<keras.callbacks.History at 0x22d56351240>
```

In [32]:
```
model1.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1707 - accuracy:
0.7541
```

Out[32]:
```
[0.17073306441307068, 0.7540983557701111]
```

In [33]:
```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100, batch_si
```

```
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
```

In [34]:
```
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 [35]:
```
ls = history1.history
```

In [36]:
```
new = pd.DataFrame.from_dict(ls)
new
```

Out[36]:

	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 [37]:
```
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 1ms/step - loss: 0.5496 - accuracy:
0.4504
Epoch 2/10
9/9 [==============================] - 0s 1ms/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: 0.5455 - accuracy:
0.4545
Epoch 5/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5455 - accuracy:
0.4545
Epoch 6/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5455 - accuracy:
0.4545
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5494 - accuracy:
0.4504
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5617 - accuracy:
0.4339
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5502 - accuracy:
0.4463
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5454 - accuracy:
0.4545
```

Out[38]:
```
<keras.callbacks.History at 0x22d57637f40>
```

In [39]:
```
model2.evaluate(X_test, y_test)
```

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

Out[39]:
```
[0.5245802402496338, 0.4754098355770111]
```

In [40]:
```
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 [41]:
```
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 [42]:
```
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 2ms/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 10ms/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[42]:
```
<keras.callbacks.History at 0x22d557c2b90>
```

In [43]:
```
model3.evaluate(X_test, y_test)
```

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

Out[43]:
```
[0.4754098355770111, 0.5245901346206665]
```

In [44]:
```
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
_________________________________________________________________
```