ROLL NO : 225229105

LAB3: Binary classification of Heart Disease of patients using Deep Neural Network

1.Load the dataset

In [15]:
```
import pandas as pd
df=pd.read_csv("heart_data.csv")
df
```

Out[15]:

	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 × 14 columns

In [5]:
```
df.shape
```

Out[5]: (303, 14)

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

Out[6]: 4242

In [7]:
```
df.columns
```

Out[7]:
```
Index(['age', 'sex', 'cp', 'trestbps', 'chol', 'fbs', 'restecg', 'thalac
h',
       'exang', 'oldpeak', 'slope', 'ca', 'thal', 'target'],
      dtype='object')
```

2.Split the dataset

In [17]:
```
X=df
y=df.pop('target')
```

In [18]:
```
#!pip install scikit-learn scipy matplotlib numpy
```

```
Requirement already satisfied: scikit-learn in c:\programdata\anaconda3
\envs\tf\lib\site-packages (1.3.0)
Requirement already satisfied: scipy in c:\programdata\anaconda3\envs\tf
\lib\site-packages (1.10.1)
Requirement already satisfied: matplotlib in c:\programdata\anaconda3\en
vs\tf\lib\site-packages (3.7.2)
Requirement already satisfied: numpy in c:\programdata\anaconda3\envs\tf
\lib\site-packages (1.25.0)
Requirement already satisfied: joblib>=1.1.1 in c:\programdata\anaconda3
\envs\tf\lib\site-packages (from scikit-learn) (1.3.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\programdata\an
aconda3\envs\tf\lib\site-packages (from scikit-learn) (3.2.0)
Requirement already satisfied: contourpy>=1.0.1 in c:\programdata\anacon
da3\envs\tf\lib\site-packages (from matplotlib) (1.1.0)
Requirement already satisfied: cycler>=0.10 in c:\programdata\anaconda3
\envs\tf\lib\site-packages (from matplotlib) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in c:\programdata\anaco
nda3\envs\tf\lib\site-packages (from matplotlib) (4.41.0)
Requirement already satisfied: kiwisolver>=1.0.1 in c:\programdata\anaco
nda3\envs\tf\lib\site-packages (from matplotlib) (1.4.4)
Requirement already satisfied: packaging>=20.0 in c:\programdata\anacond
a3\envs\tf\lib\site-packages (from matplotlib) (23.0)
Requirement already satisfied: pillow>=6.2.0 in c:\programdata\anaconda3
\envs\tf\lib\site-packages (from matplotlib) (10.0.0)
Requirement already satisfied: pyparsing<3.1,>=2.3.1 in c:\programdata\a
naconda3\envs\tf\lib\site-packages (from matplotlib) (3.0.9)
Requirement already satisfied: python-dateutil>=2.7 in c:\programdata\an
aconda3\envs\tf\lib\site-packages (from matplotlib) (2.8.2)
Requirement already satisfied: six>=1.5 in c:\programdata\anaconda3\envs
\tf\lib\site-packages (from python-dateutil>=2.7->matplotlib) (1.16.0)
```

In [20]:
```
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,
```

In [21]:
```
X_train.shape
```
Out[21]: (242, 13)

In [22]:
```
y_train.shape
```
Out[22]: (242,)

3.Create a neural network

In [23]:
```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
```

In [24]:
```
model = Sequential()
model.add(Dense(8, input_dim=13, activation='relu'))
model.add(Dense(1, activation='sigmoid'))
```

4.Complie your model

In [25]:
```
from tensorflow import keras
```

In [26]:
```
optimizer = keras.optimizers.RMSprop(learning_rate=0.001)
```

In [27]:
```
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 [==============================] - 2s 15ms/step - loss: 0.5543 - acc
uracy: 0.4380
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.5403 - accu
racy: 0.4545
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5403 - accu
racy: 0.4545
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5390 - accu
racy: 0.4504
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5406 - accu
racy: 0.4421
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5370 - accu
racy: 0.4628
Epoch 7/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5375 - accu
racy: 0.4545
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5370 - accu
racy: 0.4628
Epoch 9/10
9/9 [==============================] - 0s 1ms/step - loss: 0.5410 - accu
racy: 0.4504
Epoch 10/10
9/9 [==============================] - 0s 3ms/step - loss: 0.5357 - accu
racy: 0.4587
```

Out[27]:
```
<keras.callbacks.History at 0x1ba173597e0>
```

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

```
2/2 [==============================] - 0s 3ms/step - loss: 0.4934 - accu
racy: 0.4918
```

Out[29]:
```
[0.49344146251678467, 0.49180328845977783]
```

6.Train the model

In [30]:
```
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
_________________________________________________________________
```

In [31]:
```
model.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model.fit(X_train, y_train, epochs=200, batch_size=10, verbose=1)
```

```
Epoch 46/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3869
- accuracy: 0.4959
Epoch 47/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3658
- accuracy: 0.5537
Epoch 48/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3709
- accuracy: 0.5124
Epoch 49/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3288
- accuracy: 0.5413
Epoch 50/200
25/25 [==============================] - 0s 1ms/step - loss: 0.3215
- accuracy: 0.5909
Epoch 51/200
25/25 [==============================] - 0s 1ms/step - loss: 0.2856
- accuracy: 0.5992
Epoch 52/200
```

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

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1094 - accu
racy: 0.8689
```

Out[32]:
```
[0.1093633845448494, 0.868852436542511]
```

7.Save the trained Model

In [34]:
```
history = model.fit(X_train, y_train, validation_split=0.2, epochs=100, b
```

```
Epoch 1/100
20/20 [==============================] - 0s 6ms/step - loss: 0.1319
- accuracy: 0.8187 - val_loss: 0.1160 - val_accuracy: 0.8571
Epoch 2/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1332
- accuracy: 0.8238 - val_loss: 0.1081 - val_accuracy: 0.8776
Epoch 3/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1270
- accuracy: 0.8135 - val_loss: 0.1093 - val_accuracy: 0.8367
Epoch 4/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1195
- accuracy: 0.8342 - val_loss: 0.1752 - val_accuracy: 0.7959
Epoch 5/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1399
- accuracy: 0.7876 - val_loss: 0.1224 - val_accuracy: 0.8163
Epoch 6/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1215
- accuracy: 0.8290 - val_loss: 0.1248 - val_accuracy: 0.7755
Epoch 7/100
```

8.Evaluate

In [36]:
```
model.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.1222 - accu
racy: 0.8525
```

Out[36]:
```
[0.12221997231245041, 0.8524590134620667]
```

9.Print the model accuracy

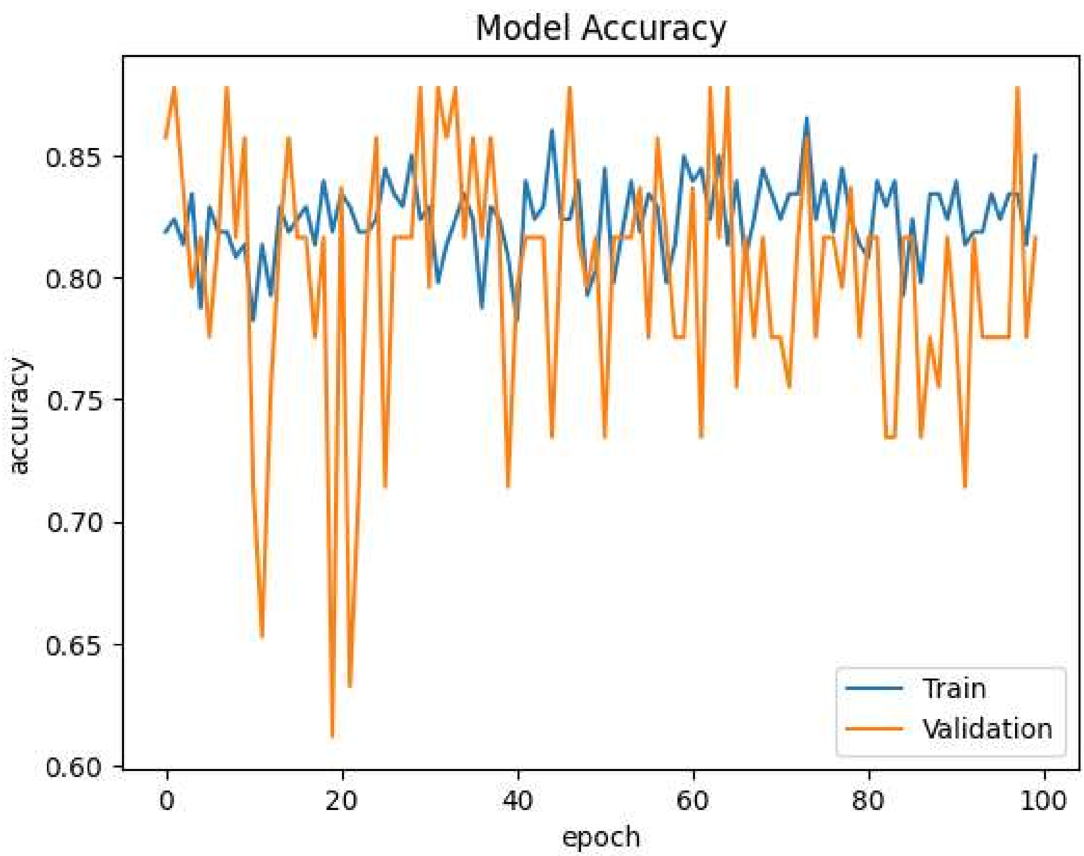
In [38]:
```
history.history.keys()
```

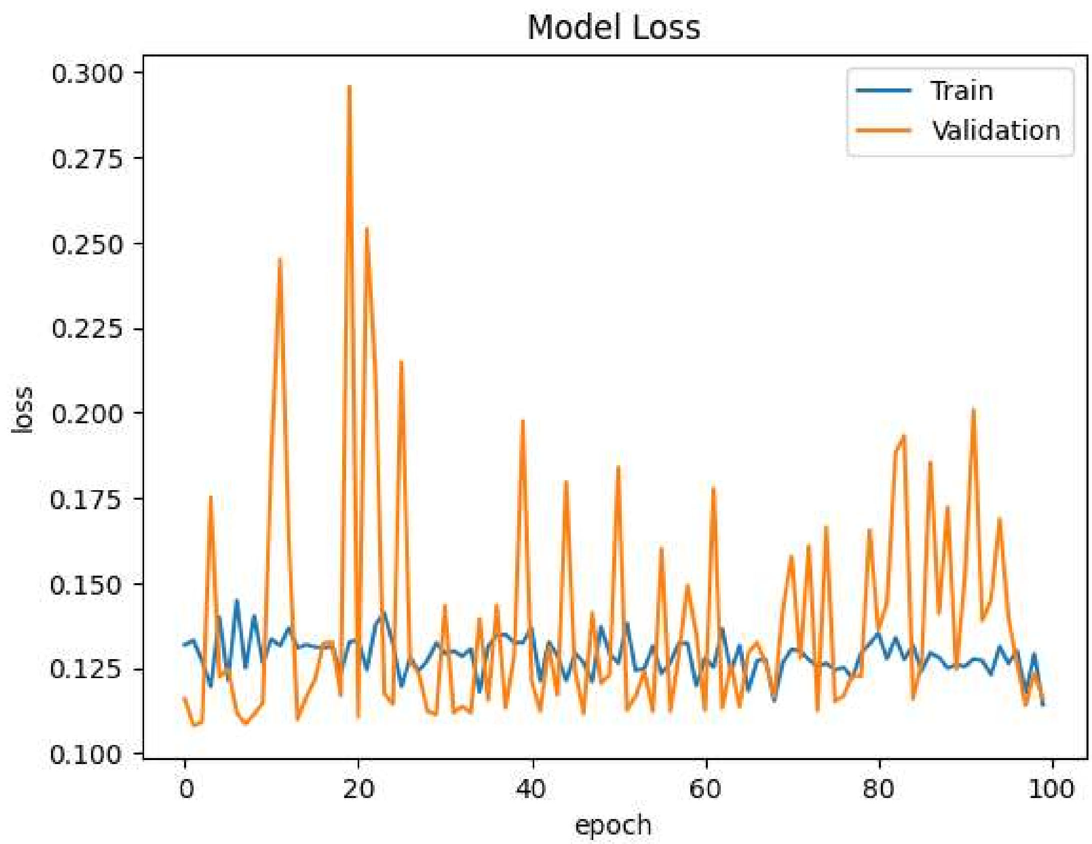
Out[38]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [39]:
```
import matplotlib.pyplot as plt
```

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

In [40]:
```
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

```
1.Add a hidden layer with 16 nodes and Relu activation function. Note that now this Dense layer
should be the first hidden layer, which is followed by the previous Dense layer with 8 nodes. Now
retrain your model, evaluate and print the accuracy and loss chart using matplotlib.
2.Add a hidden layer with 32 nodes and Relu activation function. Note that now this Dense layer
should be the first hidden layer. Now retrain your model, evaluate and print the accuracy and loss
chart using matplotlib.
3.Now, increase the nodes 64, 32, 16 for the three hidden layers. Now retrain your model, evaluate
and print the accuracy and loss chart using matplotlib.
4.Now, increase number of epochs as 150, 200, 300 and batch size as 15 and 20. Now retrain
your model, evaluate and print the accuracy and loss chart using matplotlib.
5.Now use binary_crossentropy loss function instead of mean square error loss function. Now,
compare the accuracy ad loss function values. Draw a bar chart and compare the performance.
```

In [42]:
```
model1 = Sequential()

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

In [43]:
```
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 - accu
racy: 0.5496
Epoch 2/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 3/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 7/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 8/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4504 - accu
racy: 0.5496
Epoch 10/10
9/9 [==============================] - 0s 1ms/step - loss: 0.4504 - accu
racy: 0.5496
```

Out[43]:
```
<keras.callbacks.History at 0x1ba1cf39120>
```

In [44]:
```
history1 = model.fit(X_train, y_train, validation_split=0.2, epochs=100,
```

```
20/20 [==============================] - 0s 3ms/step - loss: 0.1300
- accuracy: 0.8290 - val_loss: 0.1310 - val_accuracy: 0.7755
Epoch 30/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1210
- accuracy: 0.8394 - val_loss: 0.1181 - val_accuracy: 0.8163
Epoch 31/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1298
- accuracy: 0.8187 - val_loss: 0.1336 - val_accuracy: 0.8163
Epoch 32/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1304
- accuracy: 0.8238 - val_loss: 0.1147 - val_accuracy: 0.8571
Epoch 33/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1229
- accuracy: 0.8394 - val_loss: 0.1308 - val_accuracy: 0.8163
Epoch 34/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1226
- accuracy: 0.8290 - val_loss: 0.1627 - val_accuracy: 0.7755
Epoch 35/100
20/20 [==============================] - 0s 3ms/step - loss: 0.1309
```

In [45]:
```
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 [51]:
```
ls=history1.history
```

In [52]:
```
new = pd.DataFrame.from_dict(ls)
new
```

Out[52]:

	loss	accuracy	val_loss	val_accuracy
0	0.131734	0.818653	0.134154	0.816327
1	0.127754	0.829016	0.119010	0.857143
2	0.127098	0.839378	0.118809	0.836735
3	0.130912	0.818653	0.114429	0.877551
4	0.130033	0.823834	0.114726	0.857143
...	...	...	...	...
95	0.126855	0.839378	0.115113	0.877551
96	0.124219	0.823834	0.115173	0.816327
97	0.127046	0.829016	0.120731	0.857143
98	0.123939	0.829016	0.143220	0.755102
99	0.128357	0.823834	0.147119	0.816327

100 rows × 4 columns

In [73]:
```
history1.history.keys()
```

Out[73]:
```
dict_keys(['loss', 'accuracy', 'val_loss', 'val_accuracy'])
```

In [74]:
```
import matplotlib.pyplot as plt

# Accessing loss values from the history object
loss = history1.history['accuracy']
val_loss = history1.history['val_loss']

# Creating the loss chart
plt.plot(loss, label='Training Loss')
plt.plot(val_loss, label='Validation Loss')
plt.title('Training and Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.legend()
plt.show()
```

In [53]:
```
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 [54]:
```
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: 0.5496 - accu
racy: 0.4504
Epoch 2/10
9/9 [==============================] - 0s 2ms/step - loss: 0.4040 - accu
racy: 0.4959
Epoch 3/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2635 - accu
racy: 0.5868
Epoch 4/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2468 - accu
racy: 0.6116
Epoch 5/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2408 - accu
racy: 0.6281
Epoch 6/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2369 - accu
racy: 0.6570
Epoch 7/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2271 - accu
racy: 0.6322
Epoch 8/10
9/9 [==============================] - 0s 1ms/step - loss: 0.2662 - accu
racy: 0.6033
Epoch 9/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2431 - accu
racy: 0.6322
Epoch 10/10
9/9 [==============================] - 0s 2ms/step - loss: 0.2355 - accu
racy: 0.6281
```

Out[54]:
```
<keras.callbacks.History at 0x1ba1d865450>
```

In [55]:
```
model2.evaluate(X_test, y_test)
```

```
2/2 [==============================] - 0s 3ms/step - loss: 0.2544 - accu
racy: 0.6230
```

Out[55]:
```
[0.2543584704399109, 0.6229507923126221]
```

In [56]:
```
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 [57]:
```
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 [58]:
```
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 1ms/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
```

In [59]:

```
model3.evaluate(X_test, y_test)
```

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

Out[59]:

```
[0.4754098355770111, 0.5245901346206665]
```

In [60]:

```
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 [61]:

```
model4 = Sequential()
model4.add(Dense(150, input_dim=13, activation='relu'))
model4.add(Dense(200, activation='relu'))
model4.add(Dense(300, activation='relu'))
model4.add(Dense(8, activation='relu'))
model4.add(Dense(1, activation='sigmoid'))
```

In [66]:

```
model4.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model4.fit(X_train, y_train, epochs=10, batch_size=15, verbose=1)
```

```
Epoch 1/10
17/17 [==============================] - 1s 2ms/step - loss: 0.5286 - ac
curacy: 0.4711
Epoch 2/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 3/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 4/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 5/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 6/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 7/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 8/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 9/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 10/10
17/17 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
```

Out[66]:

```
<keras.callbacks.History at 0x1ba2364b580>
```

In [67]:

```
model4.compile(loss='mse', optimizer=optimizer, metrics=['accuracy'])
model4.fit(X_train, y_train, epochs=10, batch_size=20, verbose=1)
```

```
Epoch 1/10
13/13 [==============================] - 1s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 2/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 3/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 4/10
13/13 [==============================] - 0s 3ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 5/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 6/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 7/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 8/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 9/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
Epoch 10/10
13/13 [==============================] - 0s 2ms/step - loss: 0.5496 - ac
curacy: 0.4504
```

Out[67]:

```
<keras.callbacks.History at 0x1ba23b0a1d0>
```

In [68]:

```
model4.evaluate(X_test, y_test)
```

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

Out[68]:

```
[0.5245901346206665, 0.4754098355770111]
```

In [69]:
```
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 [ ]: