In [2]:
```
import numpy as np
import pandas as pd
import keras
from keras.models import Sequential
from keras.layers import Dense, Dropout
```

In [3]:
```
data = pd.read_csv('dataset.csv')
```

In [4]:
```
print(data)
```

```
     Year  Month    Rain  Humidity  Temperature  Pressure  Flood
0    1981      1     7.0     14.71        27.72    100.68      1
1    1981      2     6.8     14.28        30.08    100.57      1
2    1981      3    28.5     16.36        31.55    100.47      1
3    1981      4    75.9     18.55        30.81    100.35      1
4    1981      5   166.3     18.86        28.93    100.26      1
..    ...    ...     ...       ...          ...       ...    ...
451  2018      8  1398.9     18.74        25.72    100.42      1
452  2018      9   423.6     18.19        26.87    100.42      1
453  2018     10   356.1     18.37        27.52    100.47      1
454  2018     11   125.4     17.82        27.79    100.53      1
455  2018     12    65.1     16.78        27.85    100.58      1

[456 rows x 7 columns]
```

In [5]:
```
X = data[['Rain', 'Humidity', 'Temperature', 'Pressure']].values
```

In [6]:
```
print(X)
```

```
[[  7.    14.71  27.72 100.68]
 [  6.8   14.28  30.08 100.57]
 [ 28.5   16.36  31.55 100.47]
 ...
 [356.1   18.37  27.52 100.47]
 [125.4   17.82  27.79 100.53]
 [ 65.1   16.78  27.85 100.58]]
```

In [10]:
```
Y = data['Flood'].values
```

In [12]:
```
print(Y)
```

```
[1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0
 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1
 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 0 0 0 0 0 0 0 0 0
 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 1 0 0 0 0 0 0 0 0 0 0 0 0 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1 1
 1 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0
 1 1 1 1 1 1 1 1 1 1 1 1]
```

In [13]:
```
model = Sequential()
model.add(Dense(64, input_dim=4, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(32, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(1, activation='sigmoid'))
```

In [14]:
```
print(model)
```

```
<keras.engine.sequential.Sequential object at 0x000002CC30BC0D30>
```

In [15]:
```
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [16]:
```
model.fit(X, y, epochs=50, batch_size=32, validation_split=0.2)
```

```
Epoch 1/50
12/12 [==============================] - 1s 21ms/step - loss: 23.3871 - accuracy:
0.5549 - val_loss: 6.1184 - val_accuracy: 0.5217
Epoch 2/50
12/12 [==============================] - 0s 5ms/step - loss: 15.3550 - accuracy:
0.6209 - val_loss: 6.1571 - val_accuracy: 0.5217
Epoch 3/50
12/12 [==============================] - 0s 4ms/step - loss: 18.1938 - accuracy:
0.5852 - val_loss: 2.9676 - val_accuracy: 0.5217
Epoch 4/50
12/12 [==============================] - 0s 4ms/step - loss: 13.0556 - accuracy:
0.5714 - val_loss: 5.8310 - val_accuracy: 0.5217
Epoch 5/50
12/12 [==============================] - 0s 4ms/step - loss: 15.9567 - accuracy:
0.5247 - val_loss: 6.7618 - val_accuracy: 0.5217
Epoch 6/50
12/12 [==============================] - 0s 4ms/step - loss: 13.0104 - accuracy:
0.5440 - val_loss: 5.2969 - val_accuracy: 0.5217
Epoch 7/50
12/12 [==============================] - 0s 5ms/step - loss: 9.9337 - accuracy: 0.
5797 - val_loss: 2.6941 - val_accuracy: 0.5000
Epoch 8/50
12/12 [==============================] - 0s 4ms/step - loss: 8.3799 - accuracy: 0.
5549 - val_loss: 2.3674 - val_accuracy: 0.5000
Epoch 9/50
12/12 [==============================] - 0s 5ms/step - loss: 6.5247 - accuracy: 0.
5962 - val_loss: 2.0971 - val_accuracy: 0.5000
Epoch 10/50
12/12 [==============================] - 0s 5ms/step - loss: 7.1552 - accuracy: 0.
5742 - val_loss: 1.1556 - val_accuracy: 0.4783
Epoch 11/50
12/12 [==============================] - 0s 5ms/step - loss: 5.4689 - accuracy: 0.
5687 - val_loss: 0.8637 - val_accuracy: 0.4457
Epoch 12/50
12/12 [==============================] - 0s 5ms/step - loss: 5.4682 - accuracy: 0.
5467 - val_loss: 0.7764 - val_accuracy: 0.4674
Epoch 13/50
12/12 [==============================] - 0s 5ms/step - loss: 6.4473 - accuracy: 0.
5330 - val_loss: 0.7557 - val_accuracy: 0.4783
Epoch 14/50
12/12 [==============================] - 0s 5ms/step - loss: 4.1236 - accuracy: 0.
5440 - val_loss: 0.7023 - val_accuracy: 0.5435
Epoch 15/50
12/12 [==============================] - 0s 4ms/step - loss: 3.7340 - accuracy: 0.
5769 - val_loss: 0.6770 - val_accuracy: 0.5000
Epoch 16/50
12/12 [==============================] - 0s 4ms/step - loss: 3.8632 - accuracy: 0.
5385 - val_loss: 0.6712 - val_accuracy: 0.5000
Epoch 17/50
12/12 [==============================] - 0s 4ms/step - loss: 3.0452 - accuracy: 0.
5577 - val_loss: 0.6750 - val_accuracy: 0.5326
Epoch 18/50
12/12 [==============================] - 0s 5ms/step - loss: 2.5349 - accuracy: 0.
5824 - val_loss: 0.6779 - val_accuracy: 0.4891
Epoch 19/50
12/12 [==============================] - 0s 5ms/step - loss: 2.6632 - accuracy: 0.
5604 - val_loss: 0.6747 - val_accuracy: 0.5435
Epoch 20/50
12/12 [==============================] - 0s 4ms/step - loss: 2.4364 - accuracy: 0.
5907 - val_loss: 0.6719 - val_accuracy: 0.5000
Epoch 21/50
12/12 [==============================] - 0s 4ms/step - loss: 2.5411 - accuracy: 0.
5879 - val_loss: 0.6712 - val_accuracy: 0.5217
Epoch 22/50
```

```
12/12 [==============================] - 0s 4ms/step - loss: 2.0773 - accuracy: 0.
5742 - val_loss: 0.6697 - val_accuracy: 0.5000
Epoch 23/50
12/12 [==============================] - 0s 4ms/step - loss: 1.9884 - accuracy: 0.
5604 - val_loss: 0.6730 - val_accuracy: 0.5217
Epoch 24/50
12/12 [==============================] - 0s 4ms/step - loss: 1.7651 - accuracy: 0.
5632 - val_loss: 0.6791 - val_accuracy: 0.5217
Epoch 25/50
12/12 [==============================] - 0s 4ms/step - loss: 1.7586 - accuracy: 0.
6099 - val_loss: 0.6870 - val_accuracy: 0.5326
Epoch 26/50
12/12 [==============================] - 0s 4ms/step - loss: 1.6718 - accuracy: 0.
5989 - val_loss: 0.6937 - val_accuracy: 0.4565
Epoch 27/50
12/12 [==============================] - 0s 5ms/step - loss: 1.3781 - accuracy: 0.
5934 - val_loss: 0.6940 - val_accuracy: 0.5543
Epoch 28/50
12/12 [==============================] - 0s 4ms/step - loss: 1.2822 - accuracy: 0.
5989 - val_loss: 0.6913 - val_accuracy: 0.5435
Epoch 29/50
12/12 [==============================] - 0s 4ms/step - loss: 1.3542 - accuracy: 0.
5934 - val_loss: 0.6911 - val_accuracy: 0.5326
Epoch 30/50
12/12 [==============================] - 0s 4ms/step - loss: 1.4999 - accuracy: 0.
6099 - val_loss: 0.6932 - val_accuracy: 0.5326
Epoch 31/50
12/12 [==============================] - 0s 5ms/step - loss: 1.1716 - accuracy: 0.
6126 - val_loss: 0.6939 - val_accuracy: 0.5217
Epoch 32/50
12/12 [==============================] - 0s 5ms/step - loss: 1.2055 - accuracy: 0.
6099 - val_loss: 0.6946 - val_accuracy: 0.5217
Epoch 33/50
12/12 [==============================] - 0s 4ms/step - loss: 1.0229 - accuracy: 0.
6291 - val_loss: 0.6945 - val_accuracy: 0.5217
Epoch 34/50
12/12 [==============================] - 0s 4ms/step - loss: 1.0645 - accuracy: 0.
6401 - val_loss: 0.6931 - val_accuracy: 0.5217
Epoch 35/50
12/12 [==============================] - 0s 5ms/step - loss: 1.2200 - accuracy: 0.
6401 - val_loss: 0.6924 - val_accuracy: 0.5217
Epoch 36/50
12/12 [==============================] - 0s 5ms/step - loss: 0.9025 - accuracy: 0.
6154 - val_loss: 0.6927 - val_accuracy: 0.5217
Epoch 37/50
12/12 [==============================] - 0s 5ms/step - loss: 1.0240 - accuracy: 0.
6429 - val_loss: 0.6942 - val_accuracy: 0.5217
Epoch 38/50
12/12 [==============================] - 0s 5ms/step - loss: 0.9181 - accuracy: 0.
6264 - val_loss: 0.6949 - val_accuracy: 0.5217
Epoch 39/50
12/12 [==============================] - 0s 5ms/step - loss: 1.0486 - accuracy: 0.
6401 - val_loss: 0.6951 - val_accuracy: 0.5217
Epoch 40/50
12/12 [==============================] - 0s 5ms/step - loss: 1.0900 - accuracy: 0.
6236 - val_loss: 0.6938 - val_accuracy: 0.5217
Epoch 41/50
12/12 [==============================] - 0s 5ms/step - loss: 0.8502 - accuracy: 0.
6401 - val_loss: 0.6932 - val_accuracy: 0.5217
Epoch 42/50
12/12 [==============================] - 0s 5ms/step - loss: 1.0197 - accuracy: 0.
6264 - val_loss: 0.6944 - val_accuracy: 0.5217
Epoch 43/50
12/12 [==============================] - 0s 5ms/step - loss: 0.8395 - accuracy: 0.
```

```
6566 - val_loss: 0.6963 - val_accuracy: 0.5217
Epoch 44/50
12/12 [==============================] - 0s 5ms/step - loss: 0.8486 - accuracy: 0.
6236 - val_loss: 0.6951 - val_accuracy: 0.5217
Epoch 45/50
12/12 [==============================] - 0s 5ms/step - loss: 0.8134 - accuracy: 0.
6456 - val_loss: 0.6952 - val_accuracy: 0.5217
Epoch 46/50
12/12 [==============================] - 0s 5ms/step - loss: 0.6897 - accuracy: 0.
6731 - val_loss: 0.6965 - val_accuracy: 0.5217
Epoch 47/50
12/12 [==============================] - 0s 5ms/step - loss: 0.9026 - accuracy: 0.
6676 - val_loss: 0.6959 - val_accuracy: 0.5217
Epoch 48/50
12/12 [==============================] - 0s 5ms/step - loss: 0.7162 - accuracy: 0.
6538 - val_loss: 0.6956 - val_accuracy: 0.5217
Epoch 49/50
12/12 [==============================] - 0s 6ms/step - loss: 0.7319 - accuracy: 0.
6566 - val_loss: 0.6944 - val_accuracy: 0.5217
Epoch 50/50
12/12 [==============================] - 0s 5ms/step - loss: 0.8400 - accuracy: 0.
6813 - val_loss: 0.6958 - val_accuracy: 0.5217
```

Out[16]:

```
<keras.callbacks.History at 0x2cc30db6dd0>
```

In [17]:

```python
loss, accuracy = model.evaluate(X, Y)
print('Accuracy: %.2f' % (accuracy*100))
```

```
15/15 [==============================] - 0s 2ms/step - loss: 0.6679 - accuracy: 0.
6360
Accuracy: 63.60
```

In [18]:

```python
# Save the model to a file
model.save('C:\Users\AMAL\Downloads\flood\model1.h5')
```

```
  Cell In[18], line 2
    model.save('C:\Users\AMAL\Downloads\flood\model1.h5')
                                                        ^
SyntaxError: (unicode error) 'unicodeescape' codec can't decode bytes in position
2-3: truncated \UXXXXXXXX escape
```

In [19]:

```python
model.save('model.h5')
```

In [20]:

```python
import pandas as pd

# Load the new data from a CSV file
new_data = pd.read_csv('newdata1.csv')

# Extract the input features into a separate DataFrame or NumPy array
X1 = new_data[['Rain', 'Humidity', 'Temperature', 'Pressure']].values
```

In [21]:

```python
print(X1)
```

```
[[  7.4   14.47  28.   100.55]
 [ 11.    13.98  29.95 100.4 ]
 [ 21.    16.78  31.12 100.44]
 [171.1   18.92  29.52 100.33]
 [ 95.3   19.53  27.9  100.3 ]
 [430.3   18.92  26.89 100.33]
 [362.6   19.1   26.41 100.25]
 [501.6   18.43  26.29 100.4 ]
 [241.1   18.55  26.75 100.45]
 [187.5   18.37  27.59 100.41]
 [112.9   17.03  27.3  100.45]
 [  9.4   15.87  28.28 100.41]]
```

In [23]:
```
y1 = model.predict(X1)
```

```
1/1 [==============================] - 0s 150ms/step
```

In [24]:
```
print(y1)
```

```
[[0.442963  ]
 [0.44283986]
 [0.44276398]
 [0.43929195]
 [0.43962273]
 [0.43787363]
 [0.43831837]
 [0.43758154]
 [0.43957958]
 [0.43952155]
 [0.4392235 ]
 [0.443015  ]]
```

In [25]:
```
print(X1)
```

```
[[  7.4   14.47  28.   100.55]
 [ 11.    13.98  29.95 100.4 ]
 [ 21.    16.78  31.12 100.44]
 [171.1   18.92  29.52 100.33]
 [ 95.3   19.53  27.9  100.3 ]
 [430.3   18.92  26.89 100.33]
 [362.6   19.1   26.41 100.25]
 [501.6   18.43  26.29 100.4 ]
 [241.1   18.55  26.75 100.45]
 [187.5   18.37  27.59 100.41]
 [112.9   17.03  27.3  100.45]
 [  9.4   15.87  28.28 100.41]]
```

In [26]:
```
import pandas as pd

# Load the new data from a CSV file
new_data = pd.read_csv('newdata1.csv')

# Extract the input features into a separate DataFrame or NumPy array
X1 = new_data[['Rain', 'Humidity', 'Temperature', 'Pressure']].values
```

In [27]:
```
print(X1)
```

```
[[  7.    14.71  27.72 100.68]
 [  6.8   14.28  30.08 100.57]
 [ 28.5   16.36  31.55 100.47]
 [ 75.9   18.55  30.81 100.35]
 [166.3   18.86  28.93 100.26]
 [912.4   18.37  26.52 100.32]
 [489.8   18.55  26.01 100.31]
 [495.6   18.37  25.93 100.37]
 [376.6   18.43  26.05 100.32]
 [265.    18.01  26.75 100.44]
 [138.6   17.09  26.59 100.46]
 [ 43.3   15.56  26.98 100.67]]
```

In [28]:
```
y2 = model.predict(X1)
```

```
1/1 [==============================] - 0s 35ms/step
```

In [29]:
```
print(y2)
```

```
[[0.44297394]
 [0.44297874]
 [0.44254854]
 [0.43989822]
 [0.4393175 ]
 [0.4369484 ]
 [0.4376521 ]
 [0.43765008]
 [0.4383279 ]
 [0.43935314]
 [0.43897954]
 [0.44221348]]
```

In [31]:
```
out = y2[0][0]
if out >= 0.5:
    show = 'Flooding will occur'
else:
    show = 'Flooding will not occur'

print('Prediction:', show)
```

```
Prediction: Flooding will not occur
```

In [32]:
```
import pandas as pd
new_data = pd.read_csv('newdata2.csv')
X1 = new_data[['Rain', 'Humidity', 'Temperature', 'Pressure']].values
```

In [33]:
```
print(X1)
```

```
[[  29.1    14.77   28.09  100.5 ]
 [  52.1    14.28   30.37  100.53]
 [  48.6    17.09   31.32  100.39]
 [ 116.4    19.35   30.18  100.29]
 [ 183.8    19.78   28.44  100.21]
 [ 625.4    19.59   27.02  100.3 ]
 [1048.5    19.23   26.19  100.35]
 [1398.9    18.74   25.72  100.42]
 [ 423.6    18.19   26.87  100.42]
 [ 356.1    18.37   27.52  100.47]
 [ 125.4    17.82   27.79  100.53]
 [  65.1    16.78   27.85  100.58]]
```

In [34]:
```
y3 = model.predict(X1)
```

```
1/1 [==============================] - 0s 35ms/step
```

In [35]:
```
out = y3[0][0]
if out >= 0.5:
    show = 'Flooding will occur'
else:
    show = 'Flooding will not occur'

print('Prediction:', show)
```

```
Prediction: Flooding will not occur
```

In [36]:
```
print(y3)
```

```
[[0.44239047]
 [0.44137648]
 [0.44178596]
 [0.43925947]
 [0.43947312]
 [0.43718553]
 [0.4366421 ]
 [0.43517625]
 [0.43798524]
 [0.43832406]
 [0.4390607 ]
 [0.44075117]]
```

In [ ]: