In [1]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Activation,Dense
```

```
C:\Users\Anusha V\anaconda3\lib\site-packages\scipy\__init__.py:155: UserW
arning: A NumPy version >=1.18.5 and <1.25.0 is required for this version
of SciPy (detected version 1.26.2
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In [6]:

```
import pandas as pd
from sklearn.model_selection import train_test_split
from keras.models import Sequential
from keras.layers import Activation,Dense
```

In [7]:

```
import pandas as pd
data=pd.read_csv(r"C:\Users\Anusha V\Downloads\heart1.csv")
data.head()
```

Out[7]:

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	targ
0	52	1	0	125	212	0	1	168	0	1.0	2	2	3	
1	53	1	0	140	203	1	0	155	1	3.1	0	0	3	
2	70	1	0	145	174	0	1	125	1	2.6	0	0	3	
3	61	1	0	148	203	0	1	161	0	0.0	2	1	3	
4	62	0	0	138	294	1	1	106	0	1.9	1	3	2	

In [8]:

```
x = data.drop(columns=['age'])
y = data['sex']
```

In [9]:

```
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.2
```

In [11]:

```
model = Sequential()
model.add(Dense(32, activation='relu', input_shape=(x_train.shape[1],))
model.add(Dense(16, activation='relu'))
model.add(Dense(1, activation='sigmoid'))

```

In [12]:

```
model.compile(optimizer='adam', loss='binary_crossentropy', metrics=['a
model.fit(x_train, y_train, epochs=20, batch_size=32, validation_split=

Epoch 1/20
21/21 [==============================] - 3s 41ms/step - loss: 14.4664 - ac
curacy: 0.3095 - val_loss: 1.3308 - val_accuracy: 0.5915
Epoch 2/20
21/21 [==============================] - 0s 15ms/step - loss: 1.7481 - acc
uracy: 0.7088 - val_loss: 1.3201 - val_accuracy: 0.6585
Epoch 3/20
21/21 [==============================] - 0s 14ms/step - loss: 0.8365 - acc
uracy: 0.6159 - val_loss: 0.8179 - val_accuracy: 0.6280
Epoch 4/20
21/21 [==============================] - 0s 14ms/step - loss: 0.6823 - acc
uracy: 0.6585 - val_loss: 0.7886 - val_accuracy: 0.5610
Epoch 5/20
21/21 [==============================] - 0s 9ms/step - loss: 0.6579 - accu
racy: 0.6784 - val_loss: 0.7662 - val_accuracy: 0.5427
Epoch 6/20
21/21 [==============================] - 0s 8ms/step - loss: 0.6467 - accu
racy: 0.6723 - val_loss: 0.7319 - val_accuracy: 0.6098
Epoch 7/20
21/21 [==============================] - 0s 12ms/step - loss: 0.6341 - acc
uracy: 0.6768 - val_loss: 0.7145 - val_accuracy: 0.6463
Epoch 8/20
21/21 [==============================] - 0s 10ms/step - loss: 0.6305 - acc
uracy: 0.6768 - val_loss: 0.6915 - val_accuracy: 0.6098
Epoch 9/20
21/21 [==============================] - 0s 15ms/step - loss: 0.5989 - acc
uracy: 0.7088 - val_loss: 0.6830 - val_accuracy: 0.6159
Epoch 10/20
21/21 [==============================] - 0s 11ms/step - loss: 0.5894 - acc
uracy: 0.7134 - val_loss: 0.6702 - val_accuracy: 0.6280
Epoch 11/20
21/21 [==============================] - 0s 13ms/step - loss: 0.5848 - acc
uracy: 0.7073 - val_loss: 0.6567 - val_accuracy: 0.6402
Epoch 12/20
21/21 [==============================] - 0s 18ms/step - loss: 0.5726 - acc
uracy: 0.7119 - val_loss: 0.6549 - val_accuracy: 0.6585
Epoch 13/20
21/21 [==============================] - 0s 11ms/step - loss: 0.5685 - acc
uracy: 0.7165 - val_loss: 0.6417 - val_accuracy: 0.6280
Epoch 14/20
21/21 [==============================] - 0s 15ms/step - loss: 0.5608 - acc
uracy: 0.7195 - val_loss: 0.6283 - val_accuracy: 0.6402
Epoch 15/20
21/21 [==============================] - 0s 12ms/step - loss: 0.5589 - acc
uracy: 0.7088 - val_loss: 0.6176 - val_accuracy: 0.6524
Epoch 16/20
21/21 [==============================] - 0s 13ms/step - loss: 0.5463 - acc
uracy: 0.7256 - val_loss: 0.6260 - val_accuracy: 0.6707
Epoch 17/20
21/21 [==============================] - 0s 16ms/step - loss: 0.5515 - acc
uracy: 0.7317 - val_loss: 0.6032 - val_accuracy: 0.6768
Epoch 18/20
21/21 [==============================] - 0s 11ms/step - loss: 0.5299 - acc
uracy: 0.7271 - val_loss: 0.5844 - val_accuracy: 0.6585
Epoch 19/20
21/21 [==============================] - 0s 10ms/step - loss: 0.5262 - acc
uracy: 0.7393 - val_loss: 0.5889 - val_accuracy: 0.6707
Epoch 20/20
21/21 [==============================] - 0s 10ms/step - loss: 0.5180 - acc
uracy: 0.7332 - val_loss: 0.5672 - val_accuracy: 0.6768
```

Out[12]:
```
<keras.src.callbacks.History at 0x25b49c5eca0>
```

In [13]:
```python
test_loss, test_acc = model.evaluate(x_test, y_test)
print('Test accuracy:', test_acc)
```

```
7/7 [==============================] - 0s 8ms/step - loss: 0.5244 - accura
cy: 0.7220
Test accuracy: 0.7219512462615967
```

In [ ]:
```python

```

In [ ]:
```python

```

In [14]:
```python
from nltk import ngrams
from nltk.tokenize import word_tokenize
text = "The greatest glory in Living lies not in never falling but in r
words = word_tokenize(text)
def generate_ngrams(tokens, n):
    n_grams = ngrams(tokens, n)
    return [' '.join(gram) for gram in n_grams]
bi_grams = generate_ngrams(words, 2)
print("Bi-grams:", bi_grams)
tri_grams = generate_ngrams(words, 3)
print("Tri-grams:", tri_grams)

```

```
Bi-grams: ['The greatest', 'greatest glory', 'glory in', 'in Living', 'Liv
ing lies', 'lies not', 'not in', 'in never', 'never falling', 'falling bu
t', 'but in', 'in raising', 'raising every', 'every Lies']
Tri-grams: ['The greatest glory', 'greatest glory in', 'glory in Living',
'in Living lies', 'Living lies not', 'lies not in', 'not in never', 'in ne
ver falling', 'never falling but', 'falling but in', 'but in raising', 'in
raising every', 'raising every Lies']
```

In [ ]:
```python

```