Import all require modules

In [1]:

```
import pandas as pd
import numpy as np
import warnings
warnings.filterwarnings('ignore')
```

In [2]:

```
import nltk
nltk.download('punkt') #for word tokenization
nltk.download('stopwords') #for removing or getting list of stopwords
nltk.download('wordnet') #for lemmatization
```

```
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\swapn\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\swapn\AppData\Roaming\nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\swapn\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
```

Out[2]:

```
True
```

In [3]:

```
from nltk.tokenize import word_tokenize
from nltk.corpus import  stopwords
from nltk.stem import WordNetLemmatizer
import matplotlib.pyplot as plt
from wordcloud import WordCloud
```

In [4]:

```
df = pd.read_csv("google.csv")
```

In [5]:

```
df
```

Out[5]:

	Names	feedback	review-snippet	review-full-text
0	Komit Bagate	Positive:	NaN	NaN
1	omkar thorat	Positive:	I enrolled for data science course. I got to l...	I enrolled for data science course. I got to l...
2	Apoorva Shelar	Positive:	All the teaching staff and non-teaching staff ...	All the teaching staff and non-teaching staff ...
3	Shailesh Jadhav	NaN	IT vedant is very excellent institution, Becau...	IT vedant is very excellent institution, Becau...
4	Siddhi Lale	Positive:	NaN	NaN
...	...	...	...	...
281	HersheyOP	NaN	NaN	NaN
282	Yogesh Bhurawane	Critical:	NaN	NaN
283	Sunil Bhave	NaN	NaN	NaN
284	swapnil gondkar	NaN	NaN	NaN
285	shelar vaibhav	NaN	NaN	NaN

286 rows × 4 columns

In [6]:

```
df.drop(["review-snippet"],axis=1,inplace=True)
df.drop(["Names"],axis=1,inplace=True)
```

In [7]:

```
df
```

Out[7]:

	feedback	review-full-text
0	Positive:	NaN
1	Positive:	I enrolled for data science course. I got to l...
2	Positive:	All the teaching staff and non-teaching staff ...
3	NaN	IT vedant is very excellent institution, Becau...
4	Positive:	NaN
...	...	...
281	NaN	NaN
282	Critical:	NaN
283	NaN	NaN
284	NaN	NaN
285	NaN	NaN

286 rows × 2 columns

missing value treatment

In [8]:

```
#measure the nan(null) values in df
df.isnull().sum()
```

Out[8]:

```
feedback            186
review-full-text    213
dtype: int64
```

In [9]:

```
#percentage of nan(null) values in df
nullper=(df.isnull().sum()/len(df))*100
print(nullper)
```

```
feedback            65.034965
review-full-text    74.475524
dtype: float64
```

In [10]:

```
# for each column, get value counts in decreasing order and take the index (value) of most
#df_most_common_imputed = df.apply(lambda x: x.fillna(x.value_counts().index[0]))
#df_most_common_imputed
```

In [11]:

```
#missing value treatment in categorical values using mode
df['feedback'] = df['feedback'].fillna(df['feedback'].mode()[0])
df['review-full-text'] = df['review-full-text'].fillna(df['review-full-text'].mode()[0])
```

In [12]:

```
df
```

Out[12]:

	feedback	review-full-text
0	Positive:	All the teaching staff and non-teaching staff ...
1	Positive:	I enrolled for data science course. I got to l...
2	Positive:	All the teaching staff and non-teaching staff ...
3	Positive:	IT vedant is very excellent institution, Becau...
4	Positive:	All the teaching staff and non-teaching staff ...
...	...	...
281	Positive:	All the teaching staff and non-teaching staff ...
282	Critical:	All the teaching staff and non-teaching staff ...
283	Positive:	All the teaching staff and non-teaching staff ...
284	Positive:	All the teaching staff and non-teaching staff ...
285	Positive:	All the teaching staff and non-teaching staff ...

286 rows × 2 columns

In [13]:

```
from sklearn.preprocessing import LabelEncoder
le=LabelEncoder()
for col in df:
    df['feedback']=le.fit_transform(df['feedback'])
```

In [14]:

```
df
```

Out[14]:

	feedback	review-full-text
0	1	All the teaching staff and non-teaching staff ...
1	1	I enrolled for data science course. I got to l...
2	1	All the teaching staff and non-teaching staff ...
3	1	IT vedant is very excellent institution, Becau...
4	1	All the teaching staff and non-teaching staff ...
...	...	...
281	1	All the teaching staff and non-teaching staff ...
282	0	All the teaching staff and non-teaching staff ...
283	1	All the teaching staff and non-teaching staff ...
284	1	All the teaching staff and non-teaching staff ...
285	1	All the teaching staff and non-teaching staff ...

286 rows × 2 columns

In [15]:

```
wc = WordCloud(width=800, height=800, background_color="white", min_font_size=10)
wc.generate("".join(df[df['feedback']==0]['review-full-text']))

plt.figure(figsize=(6,6))
plt.imshow(wc)
plt.axis("off")
plt.show()
```

In [16]:

```
wc = WordCloud(width=800, height=800, background_color="white", min_font_size=10)
wc.generate("".join(df[df['feedback']==1]['review-full-text']))

plt.figure(figsize=(6,6))
plt.imshow(wc)
plt.axis("off")
plt.show()
```

In [17]:

```
stop = stopwords.words("english")
def clean_text(text):
  tokens = word_tokenize(text.lower())
  # Filter only alphabets
  word_tokens = [t for t in tokens if t.isalpha()]
  clean_tokens = [t for t in word_tokens if t not in stop]
  lemma = WordNetLemmatizer()
  lemma_tokens = [lemma.lemmatize(t) for t in clean_tokens]
  return " ".join(lemma_tokens)
```

In [18]:

```
df['review-full-text'] = df['review-full-text'].apply(clean_text)
```

In [19]:

```
df['review-full-text'].head()
```

Out[19]:

```
0    teaching staff staff professional available as...
1    enrolled data science course got learn many th...
2    teaching staff staff professional available as...
3    vedant excellent institution method teaching g...
4    teaching staff staff professional available as...
Name: review-full-text, dtype: object
```

In [20]:

```
x= df['review-full-text']
y= df['feedback']
```

In [21]:

```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(x, y, test_size=0.3)
```

In [22]:

```
sent_len = []
for t in df['review-full-text']:
  sent_len.append(len(word_tokenize(t)))
df['sent_len'] = sent_len
df.head()
```

Out[22]:

	feedback	review-full-text	sent_len
0	1	teaching staff staff professional available as...	32
1	1	enrolled data science course got learn many th...	39
2	1	teaching staff staff professional available as...	32
3	1	vedant excellent institution method teaching g...	40
4	1	teaching staff staff professional available as...	32

In [23]:

```
max(sent_len)
```

Out[23]:

```
96
```

In [24]:

```
np.quantile(sent_len, 0.95)
```

Out[24]:

```
49.75
```

In [25]:

```
max_len = 49
```

In [26]:

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.preprocessing.text import Tokenizer
# Creates dictionary and every unique word is given number key
from tensorflow.keras.preprocessing import sequence
# To perform the padding of the documents with zero's to make the length of the
# document common
from tensorflow.keras.layers import (LSTM, Dropout, Embedding, SimpleRNN, GRU)
# All the index numbers are converted to vectors using Embedding
# SimpleRNN allows to implement the RNN architecture - activation function -tanh
# Dropout - manage overfitting of model
```

In [27]:

```
# Tokenization
tok = Tokenizer(char_level=False, split=" ")
tok.fit_on_texts(x_train)
```

In [28]:

```
tok.index_word
```

```
65: 'really',
66: 'learning',
67: 'career',
68: 'thing',
69: 'industry',
70: 'understand',
71: 'skill',
72: 'session',
73: 'namrata',
74: 'company',
75: 'much',
76: 'always',
77: 'support',
78: 'field',
79: 'friendly',
80: 'enrolled',
81: 'highly',
82: 'thank',
83: 'development',
84: 'u',
```

In [29]:

```
vocab_len = len(tok.index_word)
vocab_len
```

Out[29]:

```
670
```

In [30]:

```
seq_train = tok.texts_to_sequences(x_train)
seq_train
```

```
  18,
  28,
  19,
  15,
  29,
  30,
  20,
  31,
  7,
  11,
  14],
 [2,
  1,
  1,
  13,
  16,
  21,
  10,
  9,
  22,
```

In [31]:

```
seq_padded_train = sequence.pad_sequences(seq_train, maxlen=max_len)
seq_padded_train
```

Out[31]:

```
array([[ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14],
       ...,
       [ 0,  0,  0, ..., 50, 42, 97],
       [ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14]])
```

In [32]:

```
model = Sequential()
# vectorization
model.add(Embedding(vocab_len+1,49, input_length=max_len, mask_zero=True))
# RNN layer
model.add(SimpleRNN(32, activation="tanh"))
# ANN's hidden layer
model.add(Dense(32, activation="relu"))
# To check on overfitting
model.add(Dropout(0.2))
# output layer
model.add(Dense(1, activation="sigmoid"))
```

In [33]:

```
model.summary()
```

```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 49, 49)            32879
_________________________________________________________________
simple_rnn (SimpleRNN)       (None, 32)                2624
_________________________________________________________________
dense (Dense)                (None, 32)                1056
_________________________________________________________________
dropout (Dropout)            (None, 32)                0
_________________________________________________________________
dense_1 (Dense)              (None, 1)                 33
=================================================================
Total params: 36,592
Trainable params: 36,592
Non-trainable params: 0
_________________________________________________________________
```

In [34]:

```
model.compile(loss="binary_crossentropy", optimizer="adam")
```

In [35]:

```
model.fit(seq_padded_train, y_train, batch_size=50, epochs=50)
```

```
Epoch 1/50
4/4 [==============================] - 1s 9ms/step - loss: 0.6346
Epoch 2/50
4/4 [==============================] - 0s 11ms/step - loss: 0.4515
Epoch 3/50
4/4 [==============================] - 0s 10ms/step - loss: 0.3437
Epoch 4/50
4/4 [==============================] - 0s 13ms/step - loss: 0.2682
Epoch 5/50
4/4 [==============================] - 0s 13ms/step - loss: 0.2040
Epoch 6/50
4/4 [==============================] - 0s 13ms/step - loss: 0.1742
Epoch 7/50
4/4 [==============================] - 0s 11ms/step - loss: 0.1205
Epoch 8/50
4/4 [==============================] - 0s 12ms/step - loss: 0.1132
Epoch 9/50
4/4 [==============================] - 0s 12ms/step - loss: 0.1012
Epoch 10/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0890
Epoch 11/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0654
Epoch 12/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0475
Epoch 13/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0868
Epoch 14/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0934
Epoch 15/50
4/4 [==============================] - 0s 9ms/step - loss: 0.0847
Epoch 16/50
4/4 [==============================] - 0s 9ms/step - loss: 0.0678
Epoch 17/50
4/4 [==============================] - 0s 14ms/step - loss: 0.0331
Epoch 18/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0792
Epoch 19/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0378
Epoch 20/50
4/4 [==============================] - 0s 10ms/step - loss: 0.1154
Epoch 21/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0474
Epoch 22/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0321
Epoch 23/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0710
Epoch 24/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0398
Epoch 25/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0700
Epoch 26/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0453
Epoch 27/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0448
Epoch 28/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0430
```

```
Epoch 29/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0369
Epoch 30/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0605
Epoch 31/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0856
Epoch 32/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0705
Epoch 33/50
4/4 [==============================] - 0s 9ms/step - loss: 0.0579
Epoch 34/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0835
Epoch 35/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0567
Epoch 36/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0854
Epoch 37/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0497
Epoch 38/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0394
Epoch 39/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0439
Epoch 40/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0324
Epoch 41/50
4/4 [==============================] - 0s 12ms/step - loss: 0.1091
Epoch 42/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0548
Epoch 43/50
4/4 [==============================] - 0s 9ms/step - loss: 0.0588
Epoch 44/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0664
Epoch 45/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0416
Epoch 46/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0884
Epoch 47/50
4/4 [==============================] - 0s 10ms/step - loss: 0.0497
Epoch 48/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0456
Epoch 49/50
4/4 [==============================] - 0s 14ms/step - loss: 0.0428
Epoch 50/50
4/4 [==============================] - 0s 12ms/step - loss: 0.0404
```

Out[35]:

```
<tensorflow.python.keras.callbacks.History at 0x205a8cf23d0>
```

In [36]:

```
seq_test = tok.texts_to_sequences(x_test)
seq_test
```

```
 [2,
  1,
  1,
  13,
  16,
  21,
  10,
  9,
  22,
  4,
  3,
  6,
  23,
  24,
  17,
  5,
  25,
  26,
  8,
  12
```

In [37]:

```
seq_padded_test = sequence.pad_sequences(seq_test, maxlen=max_len)
seq_padded_test
```

Out[37]:

```
array([[ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14],
       ...,
       [ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ...,  7, 11, 14],
       [ 0,  0,  0, ..., 41, 91, 76]])
```

In [38]:

```
y_hat = model.predict(seq_padded_test)
```

In [39]:

```
# y_hat contains probability
y_hat = np.where(y_hat>=0.5, 1, 0)
```

In [40]:

```
from sklearn.metrics import classification_report
print(classification_report( y_test, y_hat))
```

```
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         0
           1       1.00      0.87      0.93        86

    accuracy                           0.87        86
   macro avg       0.50      0.44      0.47        86
weighted avg       1.00      0.87      0.93        86
```

In [41]:

```
model = Sequential()
# vectorization
model.add(Embedding(vocab_len+1,49, input_length=max_len, mask_zero=True))
# RNN layer
# model.add(SimpleRNN(32, activation="tanh"))
model.add(LSTM(32, activation="tanh"))
# ANN's hidden layer
model.add(Dense(32, activation="relu"))
# To check on overfitting
model.add(Dropout(0.2))
# output layer
model.add(Dense(1, activation="sigmoid"))
model.compile(loss="binary_crossentropy", optimizer="adam")
model.fit(seq_padded_train, y_train, batch_size=50, epochs=50)
```

```
Epoch 1/50
4/4 [==============================] - 4s 13ms/step - loss: 0.6792
Epoch 2/50
4/4 [==============================] - 0s 16ms/step - loss: 0.6477
Epoch 3/50
4/4 [==============================] - 0s 16ms/step - loss: 0.6054
Epoch 4/50
4/4 [==============================] - 0s 13ms/step - loss: 0.5432
Epoch 5/50
4/4 [==============================] - 0s 16ms/step - loss: 0.4544
Epoch 6/50
4/4 [==============================] - 0s 16ms/step - loss: 0.3094
Epoch 7/50
4/4 [==============================] - 0s 13ms/step - loss: 0.1476
Epoch 8/50
4/4 [==============================] - 0s 19ms/step - loss: 0.0562
Epoch 9/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0615
Epoch 10/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0577
Epoch 11/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0567
Epoch 12/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0475
Epoch 13/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0411
Epoch 14/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0399
Epoch 15/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0539
Epoch 16/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0877
Epoch 17/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0753
Epoch 18/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0817
Epoch 19/50
4/4 [==============================] - 0s 16ms/step - loss: 0.1130
Epoch 20/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0719
Epoch 21/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0829
Epoch 22/50
```

```
4/4 [==============================] - 0s 13ms/step - loss: 0.0367
Epoch 23/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0453
Epoch 24/50
4/4 [==============================] - 0s 19ms/step - loss: 0.0918
Epoch 25/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0862
Epoch 26/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0406
Epoch 27/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0637
Epoch 28/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0651
Epoch 29/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0744
Epoch 30/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0577
Epoch 31/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0286
Epoch 32/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0460
Epoch 33/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0460
Epoch 34/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0399
Epoch 35/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0428
Epoch 36/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0380
Epoch 37/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0472
Epoch 38/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0620
Epoch 39/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0485
Epoch 40/50
4/4 [==============================] - 0s 19ms/step - loss: 0.0603
Epoch 41/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0393
Epoch 42/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0524
Epoch 43/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0713
Epoch 44/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0426
Epoch 45/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0667
Epoch 46/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0571
Epoch 47/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0434
Epoch 48/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0635
Epoch 49/50
4/4 [==============================] - 0s 16ms/step - loss: 0.1071
Epoch 50/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0782
```

Out[41]:

```
<tensorflow.python.keras.callbacks.History at 0x205ae1b2100>
```

In [42]:

```
y_hat = model.predict(seq_padded_test)
y_hat = np.where(y_hat>=0.5, 1, 0)
print(classification_report(y_test, y_hat))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        86

    accuracy                           1.00        86
   macro avg       1.00      1.00      1.00        86
weighted avg       1.00      1.00      1.00        86
```

In [43]:

```
model = Sequential()
# vectorization
model.add(Embedding(vocab_len+1,49, input_length=max_len, mask_zero=True))
# RNN layer
model.add(GRU(32, activation="tanh"))
# ANN's hidden layer
model.add(Dense(32, activation="relu"))
# To check on overfitting|
model.add(Dropout(0.2))
# output layer
model.add(Dense(1, activation="sigmoid"))
model.compile(loss="binary_crossentropy", optimizer="adam")
model.fit(seq_padded_train, y_train, batch_size=50, epochs=50)
```

```
Epoch 1/50
4/4 [==============================] - 3s 13ms/step - loss: 0.6942
Epoch 2/50
4/4 [==============================] - 0s 11ms/step - loss: 0.6574
Epoch 3/50
4/4 [==============================] - 0s 13ms/step - loss: 0.6250
Epoch 4/50
4/4 [==============================] - 0s 13ms/step - loss: 0.5851
Epoch 5/50
4/4 [==============================] - 0s 11ms/step - loss: 0.5381
Epoch 6/50
4/4 [==============================] - 0s 13ms/step - loss: 0.4736
Epoch 7/50
4/4 [==============================] - 0s 16ms/step - loss: 0.4055
Epoch 8/50
4/4 [==============================] - 0s 13ms/step - loss: 0.3015
Epoch 9/50
4/4 [==============================] - 0s 11ms/step - loss: 0.2234
Epoch 10/50
4/4 [==============================] - 0s 13ms/step - loss: 0.1027
Epoch 11/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0666
Epoch 12/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0533
Epoch 13/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0558
Epoch 14/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0380
Epoch 15/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0639
Epoch 16/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0620
Epoch 17/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0938
Epoch 18/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0358
Epoch 19/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0373
Epoch 20/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0370
Epoch 21/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0925
Epoch 22/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0567
```

```
Epoch 23/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0752
Epoch 24/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0490
Epoch 25/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0780
Epoch 26/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0581
Epoch 27/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0396
Epoch 28/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0903
Epoch 29/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0729
Epoch 30/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0432
Epoch 31/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0431
Epoch 32/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0934
Epoch 33/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0333
Epoch 34/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0342
Epoch 35/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0353
Epoch 36/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0516
Epoch 37/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0851
Epoch 38/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0419
Epoch 39/50
4/4 [==============================] - 0s 13ms/step - loss: 0.1099
Epoch 40/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0629
Epoch 41/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0304
Epoch 42/50
4/4 [==============================] - 0s 13ms/step - loss: 0.1024
Epoch 43/50
4/4 [==============================] - 0s 11ms/step - loss: 0.0778
Epoch 44/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0336
Epoch 45/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0668
Epoch 46/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0402
Epoch 47/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0558
Epoch 48/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0567
Epoch 49/50
4/4 [==============================] - 0s 16ms/step - loss: 0.0672
Epoch 50/50
4/4 [==============================] - 0s 13ms/step - loss: 0.0271
```

Out[43]:

```
<tensorflow.python.keras.callbacks.History at 0x205b53657c0>
```

In [44]:

```
y_hat = model.predict(seq_padded_test)
y_hat = np.where(y_hat>=0.5, 1, 0)
print(classification_report(y_test, y_hat))
```

```
              precision    recall  f1-score   support

           1       1.00      1.00      1.00        86

    accuracy                           1.00        86
   macro avg       1.00      1.00      1.00        86
weighted avg       1.00      1.00      1.00        86
```

In [ ]:

```

```