

```
import pandas as pd

spam=
pd.read_csv('C:/Users/Admin/Desktop/Nalaiyathiran/assign/spam.csv',del
imiter=',',encoding='latin-1')
```

```
spam.head()
```

```

      v1                                                    v2 Unnamed: 2
\
0  ham  Go until jurong point, crazy.. Available only ...      NaN
1  ham                                     Ok lar... Joking wif u oni...      NaN
2  spam  Free entry in 2 a wkly comp to win FA Cup fina...      NaN
3  ham  U dun say so early hor... U c already then say...      NaN
4  ham  Nah I don't think he goes to usf, he lives aro...      NaN
```

```

      Unnamed: 3 Unnamed: 4
0      NaN      NaN
1      NaN      NaN
2      NaN      NaN
3      NaN      NaN
4      NaN      NaN
```

```
spam.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed:
4'],axis=1,inplace=True)
spam.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
#   Column  Non-Null Count  Dtype
---  -
0    v1      5572 non-null     object
1    v2      5572 non-null     object
dtypes: object(2)
memory usage: 87.2+ KB
```

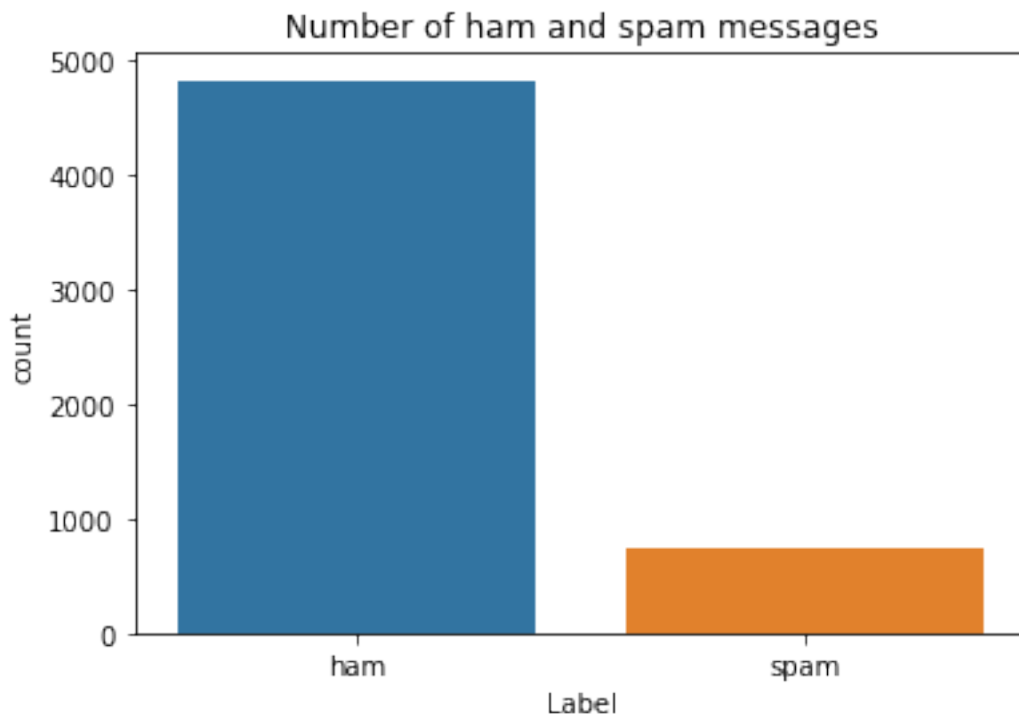
```
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.preprocessing import LabelEncoder

sns.countplot(spam.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
X = spam.v2
```

```
Y = spam.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

C:\Users\Admin\anaconda3\lib\site-packages\seaborn_decorators.py:36: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

```
warnings.warn(
```



```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from keras.models import Model
from keras.layers import LSTM, Activation, Dense, Dropout, Input,
Embedding
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.callbacks import EarlyStopping
%matplotlib inline

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)

from keras.preprocessing.sequence import pad_sequences

max_words = 1000
max_len = 150
tok = Tokenizer(num_words=max_words)
```

```

tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = pad_sequences(sequences,maxlen=max_len)

```

```

def RNN():
    inputs = Input(name='inputs',shape=[max_len])
    layer = Embedding(max_words,50,input_length=max_len)(inputs)
    layer = LSTM(64)(layer)
    layer = Dense(256,name='FC1')(layer)
    layer = Activation('relu')(layer)
    layer = Dropout(0.5)(layer)
    layer = Dense(1,name='out_layer')(layer)
    layer = Activation('sigmoid')(layer)
    model = Model(inputs=inputs,outputs=layer)
    return model

```

```

model = RNN()
model.summary()

```

Model: "model"

Layer (type)	Output Shape	Param #
inputs (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 64)	29440
FC1 (Dense)	(None, 256)	16640
activation (Activation)	(None, 256)	0
dropout (Dropout)	(None, 256)	0
out_layer (Dense)	(None, 1)	257
activation_1 (Activation)	(None, 1)	0
Total params: 96,337		
Trainable params: 96,337		
Non-trainable params: 0		

```

from tensorflow.keras.optimizers import RMSprop

```

```

model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=[
'accuracy'])

```

```

model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
validation_split=0.2,callbacks=[EarlyStopping(monitor='val_loss',min_d
elta=0.0001)])

Epoch 1/10
30/30 [=====] - 6s 144ms/step - loss: 0.3212
- accuracy: 0.8743 - val_loss: 0.1448 - val_accuracy: 0.9473
Epoch 2/10
30/30 [=====] - 4s 129ms/step - loss: 0.0865
- accuracy: 0.9810 - val_loss: 0.0528 - val_accuracy: 0.9873

<keras.callbacks.History at 0x207c54e3a90>

model.save('Spam.h5')

test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
test_sequences_matrix

array([[ 0,  0,  0, ..., 386, 696, 100],
       [ 0,  0,  0, ...,  82, 259,  2],
       [ 0,  0,  0, ..., 296,  27, 338],
       ...,
       [ 0,  0,  0, ..., 621, 377, 190],
       [ 0,  0,  0, ...,  93, 143, 11],
       [ 0,  0,  0, ..., 408, 744, 480]])

accr = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy:',accr[1])
print('Loss:',accr[0])

27/27 [=====] - 0s 13ms/step - loss: 0.0951 -
accuracy: 0.9749
Accuracy: 0.9748803973197937
Loss: 0.09510093927383423

```