Import Library

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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.optimizers import RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from keras.utils import pad_sequences
```

Read dataset

```
df = pd.read_csv('spam.csv',delimiter=',',encoding='latin-1')
df.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
```

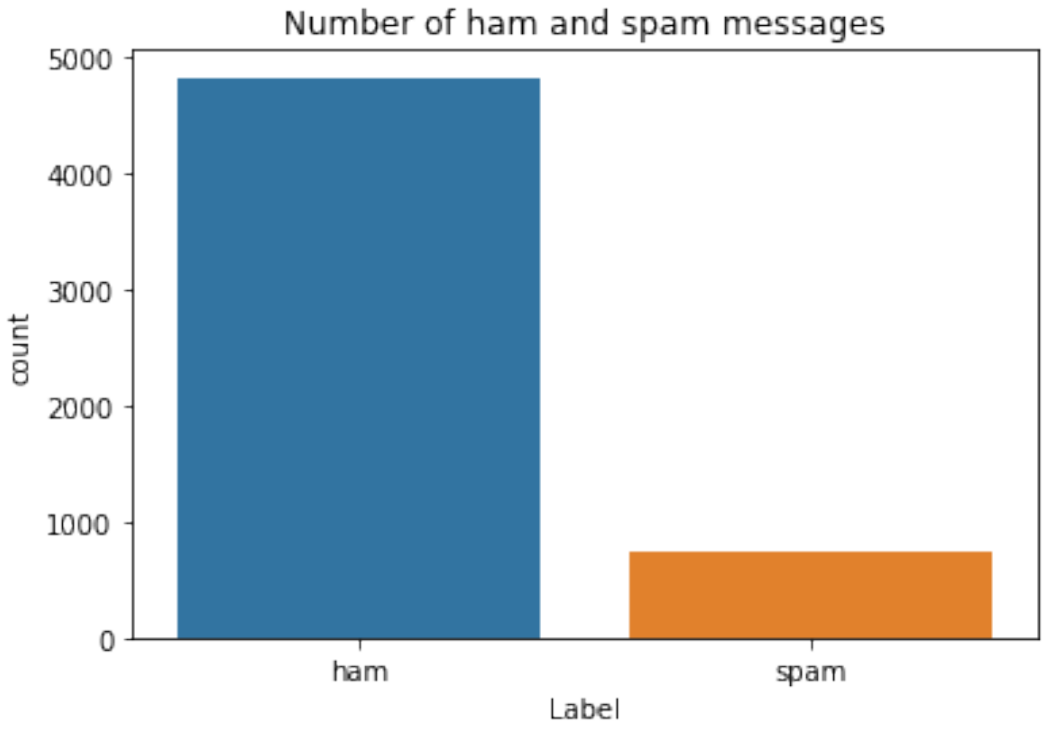
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed:
4'],axis=1,inplace=True)
df.info()
```

```
<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
```

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

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43:
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.
  FutureWarning
```

```
Text(0.5, 1.0, 'Number of ham and spam messages')
```



Create Model

```
X = df.v2
Y = df.v1
le = LabelEncoder()
```

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

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

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)
```

Add layers

```
def LSTM():
    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
```

Compile the Model

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

```
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
_________________________________________________________________
```

Fit the model

```
model.fit(sequences_matrix,Y_train,batch_size=140,epochs=8,
          validation_split=0.4)
```

```
Epoch 1/8
21/21 [==============================] - 1s 36ms/step - loss: 1.0791e-
05 - val_loss: 0.0157
Epoch 2/8
21/21 [==============================] - 0s 21ms/step - loss: 3.1017e-
06 - val_loss: 0.0152
Epoch 3/8
21/21 [==============================] - 0s 16ms/step - loss: 1.6997e-
06 - val_loss: 0.0152
Epoch 4/8
21/21 [==============================] - 0s 16ms/step - loss: 1.5786e-
06 - val_loss: 0.0152
Epoch 5/8
21/21 [==============================] - 0s 15ms/step - loss: 2.4668e-
06 - val_loss: 0.0153
Epoch 6/8
21/21 [==============================] - 0s 16ms/step - loss: 1.3341e-
06 - val_loss: 0.0152
Epoch 7/8
21/21 [==============================] - 0s 15ms/step - loss: 1.4029e-
06 - val_loss: 0.0153
Epoch 8/8
21/21 [==============================] - 0s 15ms/step - loss: 1.3283e-
06 - val_loss: 0.0153
```

```
<keras.callbacks.History at 0x7f7084819790>
```

Save Model

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

Testing

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 0s 6ms/step - loss: 0.0110
```