

**1.Download The Dataset** link using <https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification/data>

## 2. import required 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.models import load_model
%matplotlib inline
```

## 3.Read the Dataset and do pre-processing

Read the 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        |

pre-porocessing

```
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
```

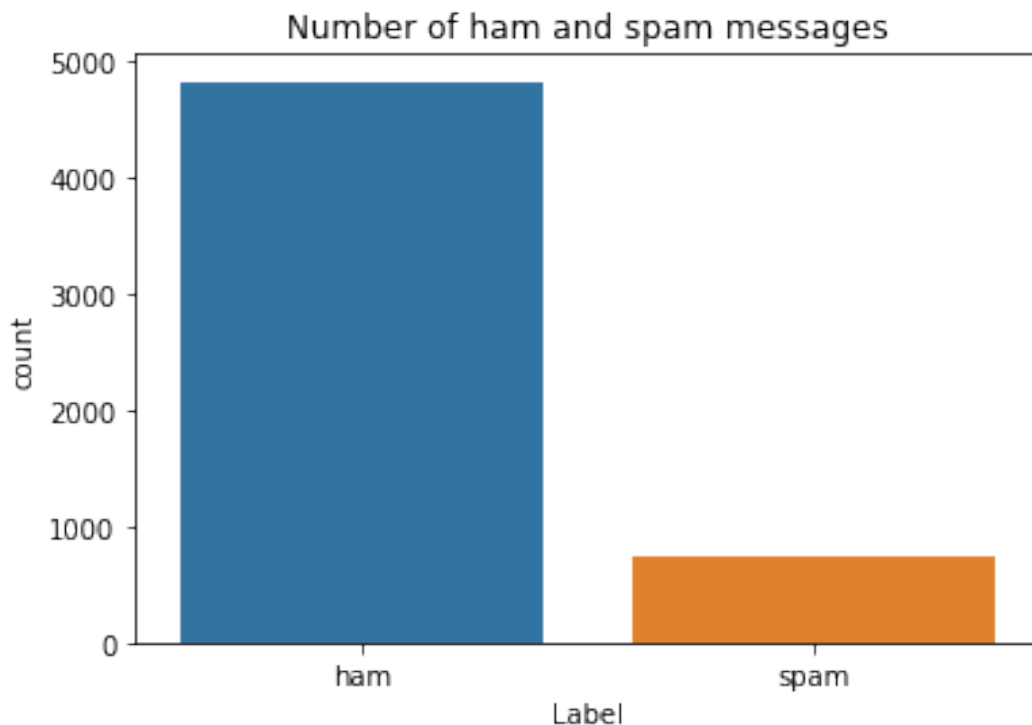
#### 4.Create Model

```
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')
```



```

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 = sequence.pad_sequences(sequences,maxlen=max_len)

```

### 5.Add Layer(LSTM,Dense-(Hidden Layers),Output)

```

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)

```

### 6.Compile the Model

```

model = Model(inputs=inputs,outputs=layer)
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) | 500000  |
| 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
=====
```

## 7. Fit the Model

```
model.fit(sequences_matrix, Y_train, batch_size=128, epochs=10,
          validation_split=0.2)
```

```
Epoch 1/10
30/30 [=====] - 8s 269ms/step - loss: 0.0476
- accuracy: 0.9852 - val_loss: 0.0552 - val_accuracy: 0.9863
Epoch 2/10
30/30 [=====] - 8s 266ms/step - loss: 0.0352
- accuracy: 0.9905 - val_loss: 0.0472 - val_accuracy: 0.9873
Epoch 3/10
30/30 [=====] - 8s 265ms/step - loss: 0.0288
- accuracy: 0.9910 - val_loss: 0.0525 - val_accuracy: 0.9852
Epoch 4/10
30/30 [=====] - 8s 266ms/step - loss: 0.0171
- accuracy: 0.9950 - val_loss: 0.0614 - val_accuracy: 0.9842
Epoch 5/10
30/30 [=====] - 11s 362ms/step - loss: 0.0132
- accuracy: 0.9958 - val_loss: 0.0714 - val_accuracy: 0.9831
Epoch 6/10
30/30 [=====] - 11s 378ms/step - loss: 0.0085
- accuracy: 0.9974 - val_loss: 0.0846 - val_accuracy: 0.9831
Epoch 7/10
30/30 [=====] - 9s 308ms/step - loss: 0.0072
- accuracy: 0.9984 - val_loss: 0.0928 - val_accuracy: 0.9800
Epoch 8/10
30/30 [=====] - 13s 430ms/step - loss: 0.0058
- accuracy: 0.9989 - val_loss: 0.0940 - val_accuracy: 0.9852
Epoch 9/10
30/30 [=====] - 14s 465ms/step - loss: 0.0050
- accuracy: 0.9984 - val_loss: 0.0996 - val_accuracy: 0.9852
Epoch 10/10
30/30 [=====] - 17s 556ms/step - loss: 0.0057
- accuracy: 0.9989 - val_loss: 0.1146 - val_accuracy: 0.9789
```

<keras.callbacks.History at 0x7f281c6e5d10>

## 8. Save the Model

```
model.save("model.h5")
```

## 9. Test the Model

```
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 [=====] - 1s 45ms/step - loss: 0.0812 -
accuracy: 0.9833

print('Test set\n Loss: {:.3f}\n Accuracy:
{:.3f}'.format(accr[0],accr[1]))

Test set
Loss: 0.081
Accuracy: 0.983
```