

#1.Spam dataset downloaded

from:- <https://www.kaggle.com/datasets/uciml/sms-spam-collection-dataset?resource=download>

#2.Required libraries are imported

```
import numpy as np
import pandas as pd
import keras

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.preprocessing.sequence import pad_sequences
%matplotlib inline
```

#3.Read dataset and pre processing

```
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

drop the unnecessary columns with Nan values

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

```
df.shape
```

```
(5572, 2)
```

#plot the ham and spam messages to understand the distribution

```
sns.countplot(df.v1)
```

```
plt.xlabel('Label')
```

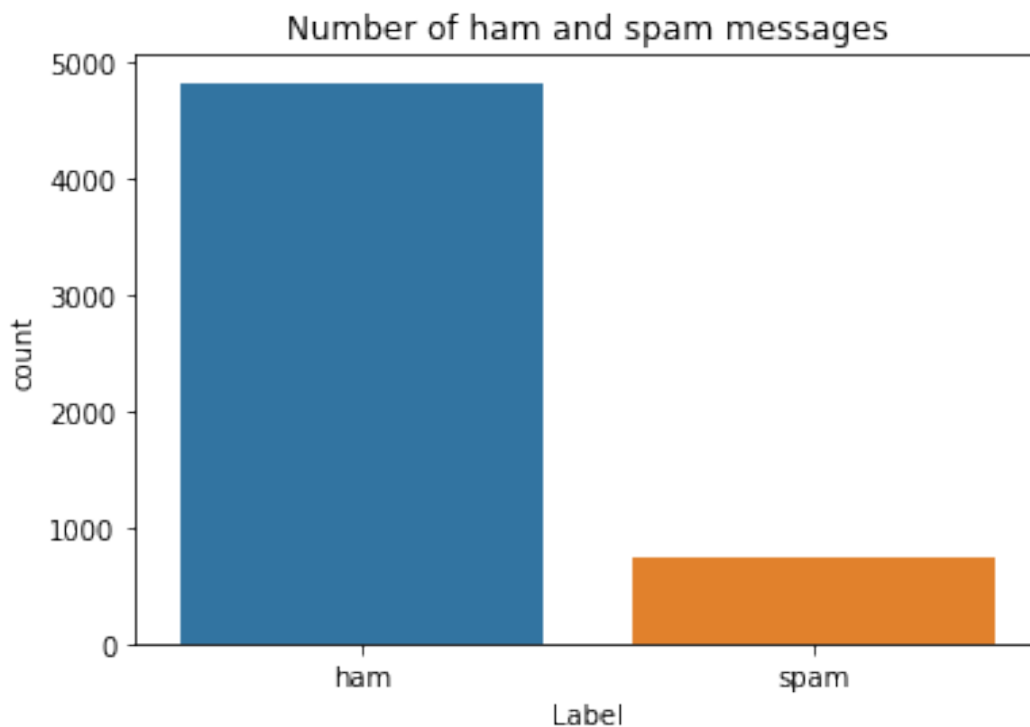
```
plt.title('Number of ham and spam messages')
```

```
/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
```

#label encoding for Y

```
le = LabelEncoder()
```

```
Y = le.fit_transform(Y)
```

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

Train-test split

#split into train and test sets

```
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.20)
```

```
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 = keras.utils.pad_sequences(sequences,maxlen=max_len)
```

#4.Create LSTM model,

#5.Add layers

```
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)
```

#6.compile the model

```
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
=====
```

#7.fit the model

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

Epoch 1/10

```
28/28 [=====] - 9s 246ms/step - loss: 0.3549
- accuracy: 0.8626 - val_loss: 0.1654 - val_accuracy: 0.9742
```

Epoch 2/10

```
28/28 [=====] - 4s 153ms/step - loss: 0.0957
- accuracy: 0.9767 - val_loss: 0.0468 - val_accuracy: 0.9821
```

<keras.callbacks.History at 0x7fec903e8390>

#8. Save the model

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

#9.test the model

#processing test data

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

#evaluation of our model

```
accr = model.evaluate(test_sequences_matrix,Y_test)
print('Test set\n Loss: {:.3f}\n Accuracy:
{:.3f}'.format(accr[0],accr[1]))
```

```
35/35 [=====] - 0s 14ms/step - loss: 0.0816 -
accuracy: 0.9776
Test set
Loss: 0.082
Accuracy: 0.978
```