Import the necessary libraries

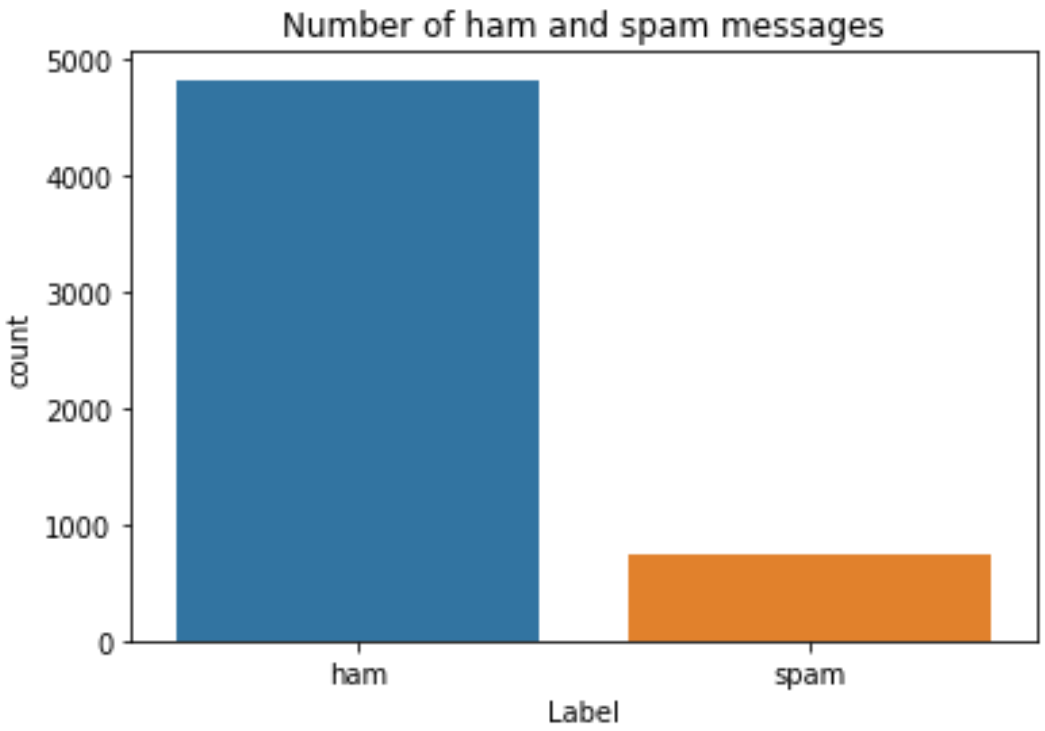
```
import pandas as pd import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
from sklearn.model_selection
import train_test_split from
sklearn.preprocessing import
LabelEncoder from keras.models
import Model
from keras.layers import LSTM,
Activation, Dense, Dropout,
Input, Embedding
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 %matplotlib
inline
Using TensorFlow backend.
Download Dataset and Load into
Dataframe
df =
pd.read_csv('../input/spam.csv'
,delimiter=',',encoding='latin-
1') df.head()
     v1
v2 Unnamed: 2
\
ham  Go until jurong point,
crazy.. Available only ...
NaN
ham                      Ok
lar... Joking wif u oni...
NaN
spam  Free entry in 2 a wkly
comp to win FA Cup fina...
NaN
ham  U dun say so early hor...
U c already then say...
NaN
ham  Nah I don't think he goes
to usf, he lives aro...
NaN
```

```
  Unnamed: 3 Unnamed: 4  0
NaN        NaN
NaN        NaN
NaN        NaN
NaN        NaN
NaN        NaN
```

Data Analysis

Drop the columns that are not required for the neural network.

```
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):
v1    5572 non-null object v2
5572 non-null object dtypes:
object(2) memory usage: 87.1+
KB
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and
spam messages')
Text(0.5,1,'Number of ham and
spam messages')
```




```
= df.v2 Y = df.v1 le =
LabelEncoder() Y =
le.fit_transform(Y)
= Y.reshape(-1,1)
X_train,X_test,Y_train,Y_test =
train_test_split(X,Y,test_size=
0.15)
Pre-Processing
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(sequence
s,maxlen=max_len)
```

Create Model and add Layers `def`

```
RNN():     inputs =
Input(name='inputs',shape=[max_
len])
    layer =
Embedding(max_words,50,input_le
ngth=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=lay
er)     return model
```

Compile the Model

```
model = RNN()
model.summary()
model.compile(loss='binary_cros
sentropy',optimizer=RMSprop(),m
etrics=[ 'accuracy'])
```

```
_________________________________
_________________________________
___ Layer (type)
Output Shape              Param
#
=================================
=================================
=== inputs (InputLayer)
(None, 150)               0
_________________________________
_________________________________
___ embedding_1 (Embedding)
(None, 150, 50)           50000
_________________________________
_________________________________
___ lstm_1 (LSTM)
(None, 64)                29440
_________________________________
_________________________________
___ FC1 (Dense)
(None, 256)               16640
_________________________________
_________________________________
___ activation_1 (Activation)
(None, 256)               0
_________________________________
_________________________________
___ dropout_1 (Dropout)
(None, 256)               0
_________________________________
_________________________________
___ out_layer (Dense)
(None, 1)                 257
```

```
_________________________________
_________________________________
___ activation_2 (Activation)
(None, 1)                 0
=================================
=================================
===
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________
_________________________________
___
Fit the Model
model.fit(sequences_matrix,Y_tr
ain,batch_size=128,epochs=10,
validation_split=0.2,callbacks=
[EarlyStopping(monitor='val_los
s',min_d elta=0.0001)])
Train on 3788 samples, validate
on 948 samples
Epoch 1/10
3788/3788
[==============================
] - 8s 2ms/step - loss:
0.3312 - acc: 0.8746 -
val_loss: 0.1460 - val_acc:
0.9504
Epoch 2/10
3788/3788
[==============================
] - 8s 2ms/step - loss:
0.0860 - acc: 0.9789 -
val_loss: 0.0666 - val_acc:
0.9768
Epoch 3/10
3788/3788
[==============================
] - 8s 2ms/step - loss:
0.0447 - acc: 0.9873 -
val_loss: 0.0465 - val_acc:
0.9895
Epoch 4/10
3788/3788
[==============================
] - 9s 2ms/step - loss:
```

```
0.0353 - acc: 0.9892 -
val_loss: 0.0459 - val_acc:
0.9863
Epoch 5/10
3788/3788
[==============================
] - 8s 2ms/step - loss:
0.0258 - acc: 0.9918 -
val_loss: 0.0437 - val_acc:
0.9884
Epoch 6/10
3788/3788
[==============================
] - 8s 2ms/step - loss:
0.0196 - acc: 0.9947 -
val_loss: 0.0468 - val_acc:
0.9905
<keras.callbacks.History at
0x7f780f71ad68>
Test the Model
test_sequences =
tok.texts_to_sequences(X_test)
test_sequences_matrix =
sequence.pad_sequences(test_seq
uences,maxlen=max_len) accr =
model.evaluate(test_sequences_m
atrix,Y_test) 836/836
[==============================
] - 1s 821us/step
print('Test set\n  Loss:
{:0.3f}\n  Accuracy:
{:0.3f}'.format(accr[0],accr[1]
))
Test set
  Loss: 0.057
  Accuracy: 0.986
```