1) Import the necessary libraries

```
import pandas as pd
import numpy as np
from keras import utils
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
%matplotlib inline
```

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

```
        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
...    ...                                                ...        .
..
5567  spam  This is the 2nd time we have tried 2 contact u...
NaN
5568   ham              Will Ì_ b going to esplanade fr home?
NaN
5569   ham  Pity, * was in mood for that. So...any other s...
NaN
5570   ham  The guy did some bitching but I acted like i'd...
NaN
5571   ham                        Rofl. Its true to its name
NaN

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

```
3           NaN        NaN
4           NaN        NaN
...         ...        ...
5567        NaN        NaN
5568        NaN        NaN
5569        NaN        NaN
5570        NaN        NaN
5571        NaN        NaN

[5572 rows x 5 columns]
```

2)Preprocessing

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

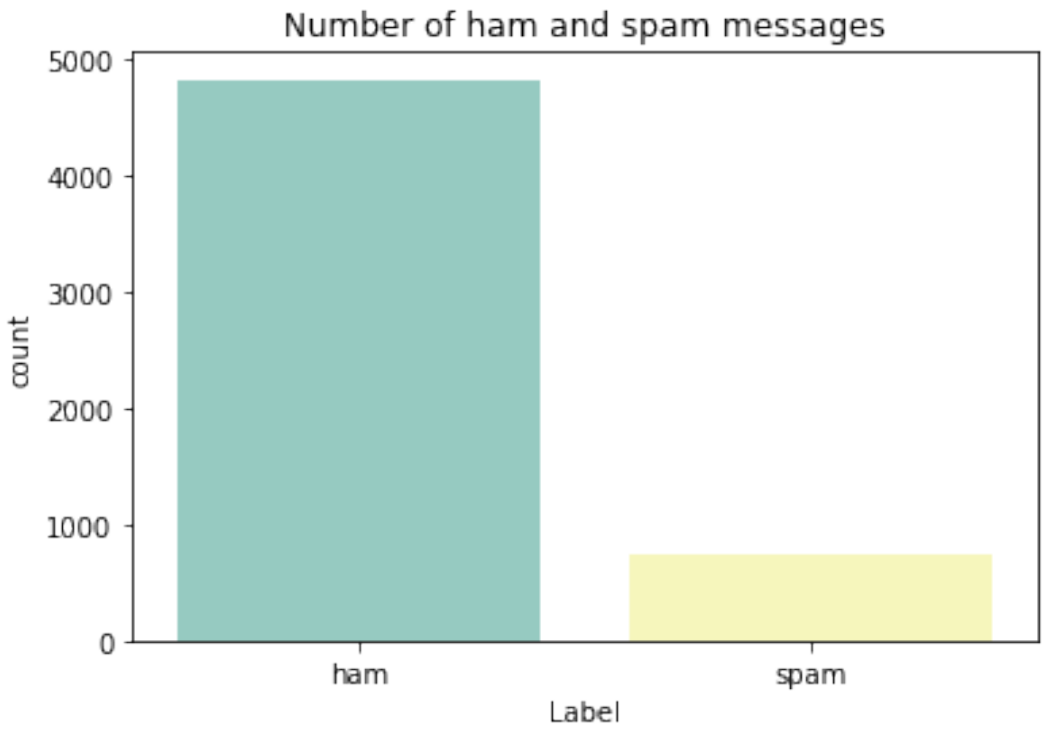
```
        v1                                                  v2
0      ham  Go until jurong point, crazy.. Available only ...
1      ham                      Ok lar... Joking wif u oni...
2     spam  Free entry in 2 a wkly comp to win FA Cup fina...
3      ham  U dun say so early hor... U c already then say...
4      ham  Nah I don't think he goes to usf, he lives aro...
...    ...                                                 ...
5567  spam  This is the 2nd time we have tried 2 contact u...
5568   ham              Will Ì_ b going to esplanade fr home?
5569   ham  Pity, * was in mood for that. So...any other s...
5570   ham  The guy did some bitching but I acted like i'd...
5571   ham                         Rofl. Its true to its name

[5572 rows x 2 columns]
```

```
sns.countplot(df.v1,palette='Set3')
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)
```

3)Split into training and test data.

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

sequences_matrix.shape

(4736, 150)

sequences_matrix.ndim

2

sequences_matrix = np.reshape(sequences_matrix,(4736,150,1))

sequences_matrix.ndim #3d shape verification to proceed to RNN LSTM

3
```

4)Create model for RNN

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding

model = Sequential()
```

5)Add Layers

```
model.add(Embedding(max_words,50,input_length=max_len))

model.add(LSTM(units=64,input_shape =
(sequences_matrix.shape[1],1),return_sequences=True))

model.add(LSTM(units=64,return_sequences=True))

model.add(LSTM(units=64,return_sequences=True))

model.add(LSTM(units=64))

model.add(Dense(units = 256,activation = 'relu'))

model.add(Dense(units = 1,activation = 'sigmoid'))
```

6)Compile the model

```
model.summary()
model.compile(loss='binary_crossentropy',optimizer='adam',metrics=['ac
curacy'])
```

```
Model: "sequential"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 150, 64)           29440

 lstm_1 (LSTM)               (None, 150, 64)           33024

 lstm_2 (LSTM)               (None, 150, 64)           33024

 lstm_3 (LSTM)               (None, 64)                33024

 dense (Dense)               (None, 256)               16640

 dense_1 (Dense)             (None, 1)                 257

=================================================================
Total params: 195,409
Trainable params: 195,409
```

```
Non-trainable params: 0
_________________________________________________________________
```

```
M =
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_
split=0.2)
M
```

```
Epoch 1/5
30/30 [==============================] - 46s 1s/step - loss: 0.4470 -
accuracy: 0.8408 - val_loss: 0.4003 - val_accuracy: 0.8576
Epoch 2/5
30/30 [==============================] - 38s 1s/step - loss: 0.2757 -
accuracy: 0.8999 - val_loss: 0.1091 - val_accuracy: 0.9673
Epoch 3/5
30/30 [==============================] - 34s 1s/step - loss: 0.0715 -
accuracy: 0.9791 - val_loss: 0.0733 - val_accuracy: 0.9736
Epoch 4/5
30/30 [==============================] - 31s 1s/step - loss: 0.0481 -
accuracy: 0.9876 - val_loss: 0.0606 - val_accuracy: 0.9800
Epoch 5/5
30/30 [==============================] - 37s 1s/step - loss: 0.0358 -
accuracy: 0.9894 - val_loss: 0.0616 - val_accuracy: 0.9821

<keras.callbacks.History at 0x7f127f4aa990>
```

8)Save the model

```
model.save
```

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

accr = model.evaluate(test_sequences_matrix,Y_test)
```

```
27/27 [==============================] - 4s 83ms/step - loss: 0.0741 -
accuracy: 0.9773
```

```
l = accr[0]
a =accr[1]
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(l,a))
```

```
Test set
  Loss: 0.074
  Accuracy: 0.977
```

Accuracy and Loss Graph

```
results = pd.DataFrame({"Train Loss": M.history['loss'], "Validation
Loss": M.history['val_loss'],
              "Train Accuracy": M.history['accuracy'], "Validation
Accuracy": M.history['val_accuracy']
```

```
            })
fig, ax = plt.subplots(nrows=2, figsize=(16, 9))
results[["Train Loss", "Validation Loss"]].plot(ax=ax[0])
results[["Train Accuracy", "Validation Accuracy"]].plot(ax=ax[1])
ax[0].set_xlabel("Epoch")
ax[1].set_xlabel("Epoch")
plt.show()
```