SMS SPAM Classification

1)Import required library

```
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
```

2) i) Read dataset

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

2) ii)Pre-processing

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
df # Drop the columns that are not requried for the neural network.
```

	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 × 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: P
  FutureWarning
```

```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
# Split into training and test data.
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
2000 |  |
```

```
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) # Padding the words to ge
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 (LSTM, Dense-(Hidden Layers), Output)

```
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=['accuracy'])
```

```
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
_________________________________________________________________
```

7)Fit the model on the training data.

```
X = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=5,validation_split=0.2)
X
```

```
Epoch 1/5
30/30 [==============================] - 45s 1s/step - loss: 0.4372 - accuracy: 0.8617
Epoch 2/5
30/30 [==============================] - 38s 1s/step - loss: 0.1539 - accuracy: 0.9490
Epoch 3/5
30/30 [==============================] - 32s 1s/step - loss: 0.0579 - accuracy: 0.9836
Epoch 4/5
30/30 [==============================] - 32s 1s/step - loss: 0.0374 - accuracy: 0.9894
Epoch 5/5
30/30 [==============================] - 33s 1s/step - loss: 0.0249 - accuracy: 0.9931
<keras.callbacks.History at 0x7f493d43cb50>
```

8)Save the model

```
model.save
```

```
<bound method Model.save of <keras.engine.sequential.Sequential object at
0x7f4941e06810>>
```

9)Evaluate the model on test set data.

```
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 85ms/step - loss: 0.0916 - accuracy: 0.9749
```

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

```
Test set
 Loss: 0.092
 Accuracy: 0.975
```