```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt

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 Adam
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping


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

	v1	v2	Unnamed: 2	Unnamed: 3	Unnamed
0	ham	Go until jurong point, crazy.. Available only ...	NaN	NaN	
1	ham	Ok lar... Joking wif u oni...	NaN	NaN	
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...	NaN	NaN	
3	ham	U dun say so early hor... U c already then say...	NaN	NaN	
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	

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


from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator


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=0.2)


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

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)

model.summary()
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 150)]             0

 embedding (Embedding)       (None, 150, 50)           50000

 lstm (LSTM)                 (None, 128)               91648

 dense (Dense)               (None, 128)               16512

 activation (Activation)     (None, 128)               0

 dropout (Dropout)           (None, 128)               0

 dense_1 (Dense)             (None, 1)                 129

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 158,289
Trainable params: 158,289
Non-trainable params: 0
_________________________________________________________________
```

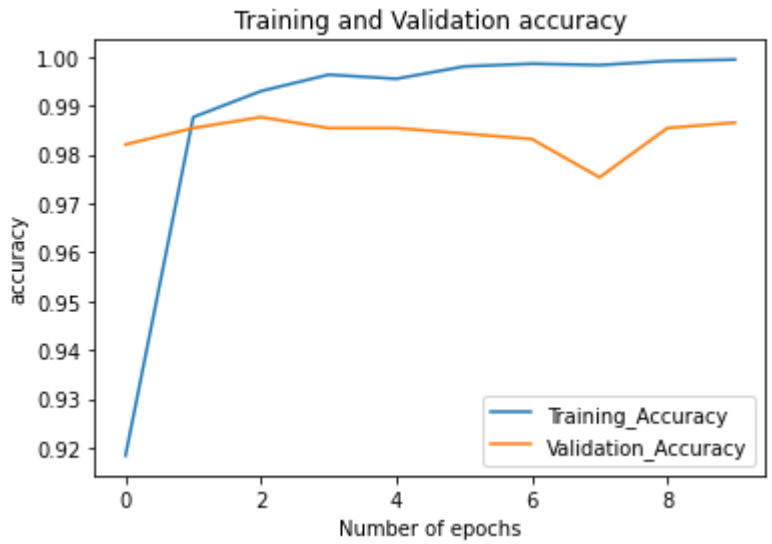
```
model.compile(loss='binary_crossentropy',optimizer=Adam(),metrics=['accuracy'])


history = model.fit(sequences_matrix,Y_train,batch_size=0,epochs=10,
          validation_split=0.2)
```

```
Epoch 1/10
112/112 [==============================] - 32s 249ms/step - loss: 0.2429 - ac
Epoch 2/10
112/112 [==============================] - 27s 241ms/step - loss: 0.0429 - ac
Epoch 3/10
112/112 [==============================] - 27s 242ms/step - loss: 0.0211 - ac
Epoch 4/10
112/112 [==============================] - 27s 242ms/step - loss: 0.0136 - ac
Epoch 5/10
112/112 [==============================] - 36s 321ms/step - loss: 0.0121 - ac
Epoch 6/10
112/112 [==============================] - 27s 242ms/step - loss: 0.0073 - ac
```

```
Epoch 7/10
112/112 [==============================] - 27s 241ms/step - loss: 0.0044 - ac
Epoch 8/10
112/112 [==============================] - 27s 245ms/step - loss: 0.0102 - ac
Epoch 9/10
112/112 [==============================] - 27s 244ms/step - loss: 0.0028 - ac
Epoch 10/10
112/112 [==============================] - 27s 242ms/step - loss: 0.0023 - ac
```

```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy',
def plot_graphs1(var1, var2, string):
    metrics[[var1, var2]].plot()
    plt.title('Training and Validation ' + string)
    plt.xlabel ('Number of epochs')
    plt.ylabel(string)
    plt.legend([var1, var2])
```

```
plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```




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

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

```
accuracy1 = model.evaluate(test_sequences_matrix,Y_test)
```

```
35/35 [==============================] - 4s 104ms/step - loss: 0.1160 - accur
```

```
print(' Accuracy: {:0.3f}'.format(accuracy1[0],accuracy1[1]))
```

```
 Accuracy: 0.116
```

Colab paid products - Cancel contracts here

✓ 0s completed at 9:38 PM