Dataset has been downloaded and saved

Import required Libraries

```
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
```

Read the Dataset

```
df = pd.read_csv('/content/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
```

Preprocessing the Dataset

```
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=0.25)

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)
```

Create Model and Add Layers

```
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
_________________________________________________________________
```

Create Model

```
model = RNN()
```

##Compiling the Model

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

Training 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
27/27 [==============================] - 7s 277ms/step - loss: 0.0092
- accuracy: 0.9982 - val_loss: 0.0804 - val_accuracy: 0.9821
Epoch 2/10
27/27 [==============================] - 8s 295ms/step - loss: 0.0069
- accuracy: 0.9982 - val_loss: 0.0843 - val_accuracy: 0.9821
```

```
<keras.callbacks.History at 0x7fcc62da8710>
```

Save the model

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

Test the model

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

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

```
44/44 [==============================] - 1s 23ms/step - loss: 0.0523 -
accuracy: 0.9892
```

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

```
Test set
  Loss: 0.052
  Accuracy: 0.989
```