Import Required Library

In [1]:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount,
call drive.mount("/content/drive", force_remount=True).
```

In [6]:

```
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

In [7]:

```
df =
pd.read_csv('/content/drive/MyDrive/44/spam.csv',delimiter=',',encoding='la
tin-1')
df.head()
```

Out[7]:

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

Pre-processing The Dataset

In [8]:

```
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

In [9]:

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
```

Add Layers

In [11]:

```
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1.5)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

In [12]:

```
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
_________________________________________________________________
```

Compile the Model

In [13]:

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

Fit the Model

In [15]:

```
history =
model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,validation_split
=0.2)
```

```
Epoch 1/15
168/168 [==============================] - 33s 183ms/step - loss: 0.1781 -
accuracy: 0.9417 - val_loss: 0.0413 - val_accuracy: 0.9856
Epoch 2/15
168/168 [==============================] - 32s 189ms/step - loss: 0.0452 -
accuracy: 0.9862 - val_loss: 0.0387 - val_accuracy: 0.9892
```

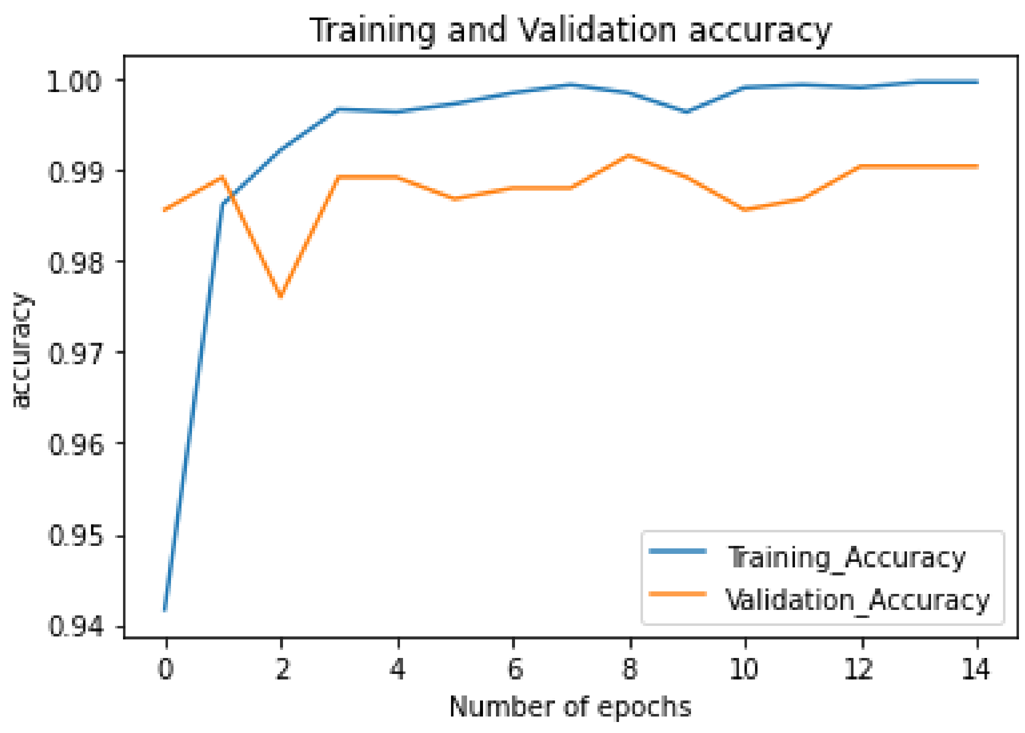
```
Epoch 3/15
168/168 [==============================] - 30s 180ms/step - loss: 0.0267 -
accuracy: 0.9922 - val_loss: 0.0623 - val_accuracy: 0.9761
Epoch 4/15
168/168 [==============================] - 30s 178ms/step - loss: 0.0134 -
accuracy: 0.9967 - val_loss: 0.0368 - val_accuracy: 0.9892
Epoch 5/15
168/168 [==============================] - 30s 178ms/step - loss: 0.0107 -
accuracy: 0.9964 - val_loss: 0.0406 - val_accuracy: 0.9892
Epoch 6/15
168/168 [==============================] - 30s 178ms/step - loss: 0.0080 -
accuracy: 0.9973 - val_loss: 0.0563 - val_accuracy: 0.9868
Epoch 7/15
168/168 [==============================] - 30s 181ms/step - loss: 0.0055 -
accuracy: 0.9985 - val_loss: 0.0411 - val_accuracy: 0.9880
Epoch 8/15
168/168 [==============================] - 30s 181ms/step - loss: 0.0041 -
accuracy: 0.9994 - val_loss: 0.0804 - val_accuracy: 0.9880
Epoch 9/15
168/168 [==============================] - 32s 193ms/step - loss: 0.0063 -
accuracy: 0.9985 - val_loss: 0.0532 - val_accuracy: 0.9916
Epoch 10/15
168/168 [==============================] - 30s 180ms/step - loss: 0.0095 -
accuracy: 0.9964 - val_loss: 0.0894 - val_accuracy: 0.9892
Epoch 11/15
168/168 [==============================] - 30s 181ms/step - loss: 0.0037 -
accuracy: 0.9991 - val_loss: 0.0969 - val_accuracy: 0.9856
Epoch 12/15
168/168 [==============================] - 31s 184ms/step - loss: 0.0029 -
accuracy: 0.9994 - val_loss: 0.0715 - val_accuracy: 0.9868
Epoch 13/15
168/168 [==============================] - 31s 187ms/step - loss: 0.0021 -
accuracy: 0.9991 - val_loss: 0.0665 - val_accuracy: 0.9904
Epoch 14/15
168/168 [==============================] - 30s 181ms/step - loss: 0.0023 -
accuracy: 0.9997 - val_loss: 0.0627 - val_accuracy: 0.9904
Epoch 15/15
168/168 [==============================] - 33s 196ms/step - loss: 0.0018 -
accuracy: 0.9997 - val_loss: 0.0682 - val_accuracy: 0.9904
```

In [19]:

```
metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy':
'Training_Accuracy', 'val_loss': 'Validation_Loss', 'val_accuracy':
'Validation_Accuracy'}, inplace = True)
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])
```

In [20]:

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



Save The Model

In [21]:

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

Test The Model

In [22]:

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

In [23]:

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

```
44/44 [==============================] - 3s 77ms/step - loss: 0.1322 -
accuracy: 0.9835
```

In [24]:

```
print(' Accuracy: {:0.5f}'.format(accuracy1[0],accuracy1[1]))
```

```
 Accuracy: 0.13224
```