Import Required Library

```
In [2]:
```

```
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 [3]:
```

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

```
Out[3]:
```

	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 [5]:
```

```
df.drop(['Unnamed: 4', 'Unnamed: 2', 'Unnamed: 3'],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 [6]:
```

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

Add Layers

Add Layers

In [7]:

```
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1.2)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
```

In [8]:

```
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 [9]:

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

Fit the Model

In [10]:

```
history = model.fit(sequences_matrix,Y_train,batch_size=25,epochs=16,validation_split=0.
1)
```

```
Epoch 1/16
151/151 [==============================] - 37s 224ms/step - loss: 0.1788 - accuracy: 0.94
20 - val_loss: 0.0949 - val_accuracy: 0.9665
Epoch 2/16
151/151 [==============================] - 34s 225ms/step - loss: 0.0379 - accuracy: 0.98
94 - val_loss: 0.1058 - val_accuracy: 0.9713
Epoch 3/16
151/151 [==============================] - 33s 216ms/step - loss: 0.0175 - accuracy: 0.99
60 - val_loss: 0.1092 - val_accuracy: 0.9737
Epoch 4/16
151/151 [==============================] - 32s 213ms/step - loss: 0.0097 - accuracy: 0.99
76 - val_loss: 0.1404 - val_accuracy: 0.9737
Epoch 5/16
151/151 [==============================] - 32s 211ms/step - loss: 0.0037 - accuracy: 0.99
89 - val_loss: 0.1475 - val_accuracy: 0.9785
Epoch 6/16
```

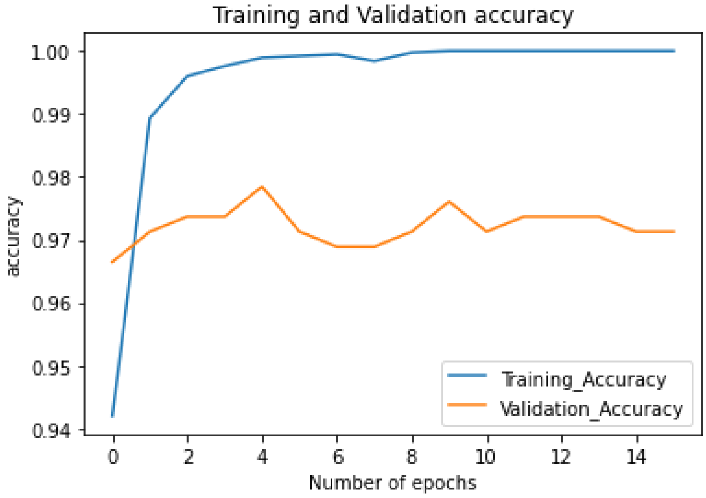
```
151/151 [==============================] - 33s 221ms/step - loss: 0.0031 - accuracy: 0.99
92 - val_loss: 0.1584 - val_accuracy: 0.9713
Epoch 7/16
151/151 [==============================] - 32s 212ms/step - loss: 0.0049 - accuracy: 0.99
95 - val_loss: 0.1769 - val_accuracy: 0.9689
Epoch 8/16
151/151 [==============================] - 32s 214ms/step - loss: 0.0046 - accuracy: 0.99
84 - val_loss: 0.1971 - val_accuracy: 0.9689
Epoch 9/16
151/151 [==============================] - 33s 218ms/step - loss: 7.8598e-04 - accuracy:
0.9997 - val_loss: 0.2396 - val_accuracy: 0.9713
Epoch 10/16
151/151 [==============================] - 34s 222ms/step - loss: 3.5395e-04 - accuracy:
1.0000 - val_loss: 0.2378 - val_accuracy: 0.9761
Epoch 11/16
151/151 [==============================] - 33s 219ms/step - loss: 6.6110e-05 - accuracy:
1.0000 - val_loss: 0.2559 - val_accuracy: 0.9713
Epoch 12/16
151/151 [==============================] - 33s 220ms/step - loss: 4.4396e-05 - accuracy:
1.0000 - val_loss: 0.2653 - val_accuracy: 0.9737
Epoch 13/16
151/151 [==============================] - 34s 227ms/step - loss: 4.1752e-05 - accuracy:
1.0000 - val_loss: 0.2683 - val_accuracy: 0.9737
Epoch 14/16
151/151 [==============================] - 33s 220ms/step - loss: 3.2523e-05 - accuracy:
1.0000 - val_loss: 0.2762 - val_accuracy: 0.9737
Epoch 15/16
151/151 [==============================] - 32s 212ms/step - loss: 2.0555e-05 - accuracy:
1.0000 - val_loss: 0.2821 - val_accuracy: 0.9713
Epoch 16/16
151/151 [==============================] - 33s 219ms/step - loss: 2.3620e-05 - accuracy:
1.0000 - val_loss: 0.2888 - val_accuracy: 0.9713
```

In [11]:

```
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 [12]:

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



Save The Model

In [13]:

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

Test The Model

In [14]:

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

In [15]:

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

```
44/44 [==============================] - 4s 86ms/step - loss: 0.1831 - accuracy: 0.9799
```

In [17]:

```
print(' Accuracy: {:0.4f}'.format(accuracy1[0],accuracy1[1]))
```

```
 Accuracy: 0.1831
```