

Import Required Library

In [33]:

```
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 [34]:

```
df = pd.read_csv('//SPAM(2).csv', delimiter=',', encoding='latin-1')
df.head()
```

Out[34]:

	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 [35]:

```
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 [36]:

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

Add Layers

Add Layers

In [37]:

```
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)
```

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 [38]:

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

Fit the Model

In [40]:

```
history = model.fit(sequences_matrix, Y_train, batch_size=15, epochs=10, validation_split=0.1)
```

Epoch 1/10

251/251 [=====] - 42s 166ms/step - loss: 0.0740 - accuracy: 0.9803 - val_loss: 0.0386 - val_accuracy: 0.9856

Epoch 2/10

251/251 [=====] - 40s 161ms/step - loss: 0.0338 - accuracy: 0.9910 - val_loss: 0.0224 - val_accuracy: 0.9952

Epoch 3/10

251/251 [=====] - 42s 166ms/step - loss: 0.0213 - accuracy: 0.9944 - val_loss: 0.0341 - val_accuracy: 0.9833

Epoch 4/10

251/251 [=====] - 41s 161ms/step - loss: 0.0113 - accuracy: 0.9968 - val_loss: 0.0404 - val_accuracy: 0.9880

Epoch 5/10

251/251 [=====] - 42s 166ms/step - loss: 0.0131 - accuracy: 0.9960 - val_loss: 0.0445 - val_accuracy: 0.9833

Epoch 6/10

```

251/251 [=====] - 41s 163ms/step - loss: 0.0054 - accuracy: 0.99
87 - val_loss: 0.0398 - val_accuracy: 0.9856
Epoch 7/10
251/251 [=====] - 41s 162ms/step - loss: 0.0055 - accuracy: 0.99
87 - val_loss: 0.0687 - val_accuracy: 0.9809
Epoch 8/10
251/251 [=====] - 41s 165ms/step - loss: 0.0075 - accuracy: 0.99
81 - val_loss: 0.0489 - val_accuracy: 0.9833
Epoch 9/10
251/251 [=====] - 41s 163ms/step - loss: 0.0039 - accuracy: 0.99
89 - val_loss: 0.0644 - val_accuracy: 0.9809
Epoch 10/10
251/251 [=====] - 41s 163ms/step - loss: 0.0122 - accuracy: 0.99
57 - val_loss: 0.1195 - val_accuracy: 0.9689

```

In [41]:

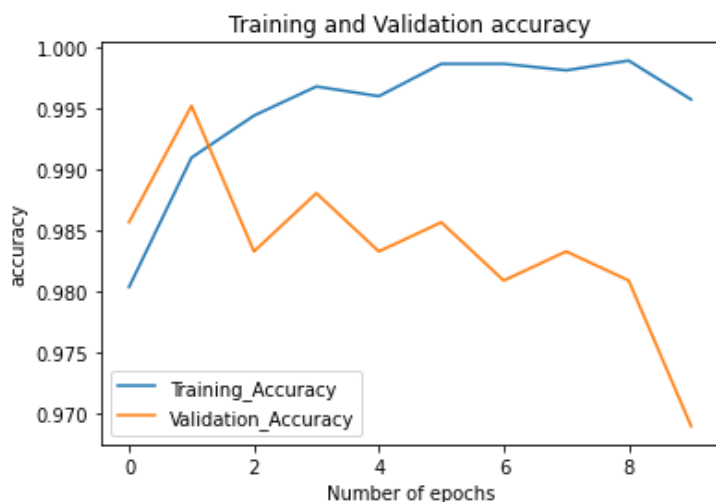
```

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 [42]:

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



Save The Model

In [43]:

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

Test The Model

In [44]:

```

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

```

In [45]:

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

```
44/44 [=====] - 3s 79ms/step - loss: 0.1191 - accuracy: 0.9727
```

In [46]:

```
print(' Accuracy: {:.4f}'.format(accuracy1[0],accuracy1[1]))
```

Accuracy: 0.1191