Assignment Date	30 October 2022
Student Name	A.R.Aarthi
Student Roll no	910619104001
Maximum Marks	2 Marks

Spam Classification

Import Libraries

In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
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 RMSprop
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
import tensorflow
from tensorflow.keras.preprocessing.sequence import pad_sequences
```

Read CSV File

In [3]:
```
df=pd.read_csv("spam.csv",encoding='ISO-8859-1')
```

In [4]:
```
df.head()
```

Out[4]:

	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

In [5]:
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'], axis=1, inplace=True)
df.head(10)
```

Out[5]:

	v1	v2
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...
5	spam	FreeMsg Hey there darling it's been 3 week's n...
6	ham	Even my brother is not like to speak with me. ...
7	ham	As per your request 'Melle Melle (Oru Minnamin...
8	spam	WINNER!! As a valued network customer you have...
9	spam	Had your mobile 11 months or more? U R entitle...

Model Creation

In [6]:
```
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

In [7]:
```
X_train, X_test, Y_train, Y_test = train_test_split(X, Y, test_size=0.30, random_state=7)
```

In [8]:
```
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)
```

Adding Layers

In [9]:
```
def RNN_model():
 inputs = Input(name='inputs', shape=(max_len))
 layer = Embedding(max_words, 50, input_length=max_len)(inputs)
 layer = LSTM(64)(layer)
 layer = Dense(256, name='FC1')(layer)
 layer = Activation('relu')(layer)
 layer = Dropout(0.5)(layer)
 layer = Dense(1, name='out_layer')(layer)
 layer = Activation('sigmoid')(layer)
 model = Model(inputs=inputs, outputs=layer)
 return model
```

Model Compilation

In [10]:
```
model = RNN_model()
model.compile(loss='binary_crossentropy', optimizer='adam', metrics=['accuracy'])
```

In [11]:
```
model.summary()
```

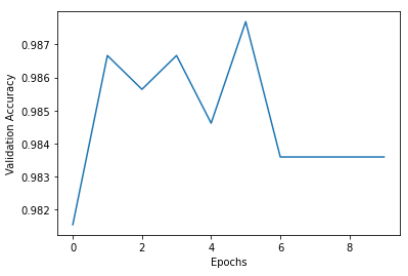
```
Model: "model"
 Layer (type)                Output Shape              Param #
=================================================================
 inputs (InputLayer)         [(None, 150)]             0

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

 lstm (LSTM)                 (None, 64)                29440

 FC1 (Dense)                 (None, 256)               16640

 activation (Activation)     (None, 256)               0

 dropout (Dropout)           (None, 256)               0

 out_layer (Dense)           (None, 1)                 257

 activation_1 (Activation)   (None, 1)                 0

=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
```

In [12]:
```
data = model.fit(sequences_matrix, Y_train, batch_size=16, epochs=10, validation_split=0.25)
```

```
Epoch 1/10
183/183 [==============================] - 17s 69ms/step - loss: 0.1828 - accuracy: 0.9395 - val_loss: 0.0687 - val_accuracy: 0.9815
Epoch 2/10
183/183 [==============================] - 11s 60ms/step - loss: 0.0322 - accuracy: 0.9911 - val_loss: 0.0539 - val_accuracy: 0.9867
Epoch 3/10
183/183 [==============================] - 11s 62ms/step - loss: 0.0145 - accuracy: 0.9969 - val_loss: 0.0602 - val_accuracy: 0.9856
Epoch 4/10
183/183 [==============================] - 11s 61ms/step - loss: 0.0045 - accuracy: 0.9990 - val_loss: 0.0876 - val_accuracy: 0.9867
Epoch 5/10
183/183 [==============================] - 11s 61ms/step - loss: 0.0015 - accuracy: 0.9997 - val_loss: 0.0857 - val_accuracy: 0.9846
Epoch 6/10
183/183 [==============================] - 11s 62ms/step - loss: 0.0070 - accuracy: 0.9983 - val_loss: 0.0934 - val_accuracy: 0.9877
Epoch 7/10
183/183 [==============================] - 11s 62ms/step - loss: 0.0021 - accuracy: 0.9997 - val_loss: 0.0818 - val_accuracy: 0.9836
Epoch 8/10
183/183 [==============================] - 11s 61ms/step - loss: 0.0015 - accuracy: 0.9997 - val_loss: 0.1094 - val_accuracy: 0.9836
Epoch 9/10
183/183 [==============================] - 11s 61ms/step - loss: 1.7969e-04 - accuracy: 1.0000 - val_loss: 0.1135 - val_accuracy: 0.9836
Epoch 10/10
183/183 [==============================] - 11s 62ms/step - loss: 7.7695e-05 - accuracy: 1.0000 - val_loss: 0.1191 - val_accuracy: 0.9836
```

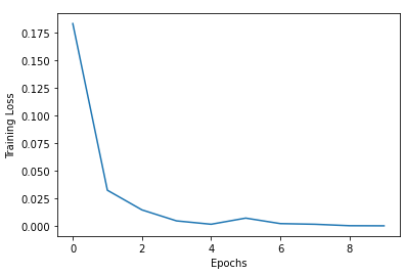
In [13]:
```
plt.figure()
plt.xlabel('Epochs')
plt.ylabel('Validation Accuracy')
plt.plot(data.epoch,data.history['val_accuracy'])
```

Out[13]:
```
[<matplotlib.lines.Line2D at 0x1f044ed7850>]
```



In [14]:
```
plt.figure()
plt.xlabel('Epochs')
plt.ylabel('Training Loss')
plt.plot(data.epoch, data.history['loss'])
```

Out[14]:
```
[<matplotlib.lines.Line2D at 0x1f045f88e20>]
```



Saving Model

In [15]:
```
model.save('Spam_Detector_model.h5')
```

Testing the Model

In [16]:
```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

In [17]:
```
test_accuracy = model.evaluate(test_sequences_matrix, Y_test)
```

```
53/53 [==============================] - 1s 26ms/step - loss: 0.1555 - accuracy: 0.9779
```

In [18]:
```
model.metrics_names
```

Out[18]:
```
['loss', 'accuracy']
```

In [19]:
```
print('Test Loss: {: 0.4f} and Test Accuracy: {: 0.2f}%'.format(test_accuracy[0], test_accuracy[1]*100))
```

```
Test Loss:  0.1555 and Test Accuracy:  97.79%
```