```
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 RMSprop
from keras.preprocessing.text import Tokenizer
from keras_preprocessing import sequence
from keras.utils import to_categorical
from keras.models import load_model
```

```
df = pd.read_csv("/content/sample_data/spam.csv",encoding='ISO-8859-1')
```

```
df
```

	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
...	...	...	...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...	NaN	NaN	NaN
5568	ham	Will Ì_ b going to esplanade fr home?	NaN	NaN	NaN
5569	ham	Pity, * was in mood for that. So...any other s...	NaN	NaN	NaN
5570	ham	The guy did some bitching but I acted like i'd...	NaN	NaN	NaN
5571	ham	Rofl. Its true to its name	NaN	NaN	NaN

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

	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...
...	...	...
5567	spam	This is the 2nd time we have tried 2 contact u...
5568	ham	Will Ì_ b going to esplanade fr home?
5569	ham	Pity, * was in mood for that. So...any other s...

```
df.groupby(['v1']).size()
```

```
v1
ham     4825
spam     747
dtype: int64
```

```
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=0.15)
```

```
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 = sequence.pad_sequences(sequences,maxlen=max_len)
```

```
sequences_matrix
```

```
array([[  0,   0,   0, ...,   2,   3, 102],
       [  0,   0,   0, ..., 737, 788,  53],
       [  0,   0,   0, ..., 312, 898, 104],
       ...,
       [  0,   0,   0, ..., 227,  42,  41],
       [  0,   0,   0, ...,   8,  35, 769],
       [  0,   0,   0, ..., 141, 104,  56]], dtype=int32)
```

```
inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)


model = Model(inputs=inputs,outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 InputLayer (InputLayer)     [(None, 150)]             0         
                                                                 
 embedding (Embedding)       (None, 150, 50)           50000     
                                                                 
 lstm (LSTM)                 (None, 64)                29440     
                                                                 
 FullyConnectedLayer1 (Dense  (None, 256)              16640     
 )                                                               
                                                                 
 activation (Activation)     (None, 256)               0         
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 OutputLayer (Dense)         (None, 1)                 257       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
```

```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
30/30 [==============================] - 8s 164ms/step - loss: 0.3430 - accuracy: 0.867
Epoch 2/10
30/30 [==============================] - 5s 151ms/step - loss: 0.0958 - accuracy: 0.974
```

```
Epoch 3/10
30/30 [==============================] - 5s 152ms/step - loss: 0.0490 - accuracy: 0.986
Epoch 4/10
30/30 [==============================] - 5s 151ms/step - loss: 0.0341 - accuracy: 0.988
Epoch 5/10
30/30 [==============================] - 4s 149ms/step - loss: 0.0244 - accuracy: 0.992
Epoch 6/10
30/30 [==============================] - 4s 150ms/step - loss: 0.0206 - accuracy: 0.994
Epoch 7/10
30/30 [==============================] - 4s 150ms/step - loss: 0.0145 - accuracy: 0.995
Epoch 8/10
30/30 [==============================] - 5s 151ms/step - loss: 0.0126 - accuracy: 0.996
Epoch 9/10
30/30 [==============================] - 5s 152ms/step - loss: 0.0103 - accuracy: 0.997
Epoch 10/10
30/30 [==============================] - 4s 150ms/step - loss: 0.0077 - accuracy: 0.997
<keras.callbacks.History at 0x7f2b8019ffd0>
```

```
model.save('spam_model')
```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) InputLayer with unsupport
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_
```

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix  = sequence.pad_sequences(test_sequences,maxlen=max_len)
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:0.3f}'.format(accuracy[1]))
```

```
27/27 [==============================] - 1s 21ms/step - loss: 0.0986 - accuracy: 0.9833
Accuracy: 0.983
```

```
y_pred = model.predict(test_sequences_matrix)
pred = y_pred[25:40].round(3)
test =Y_test[25:40]
```

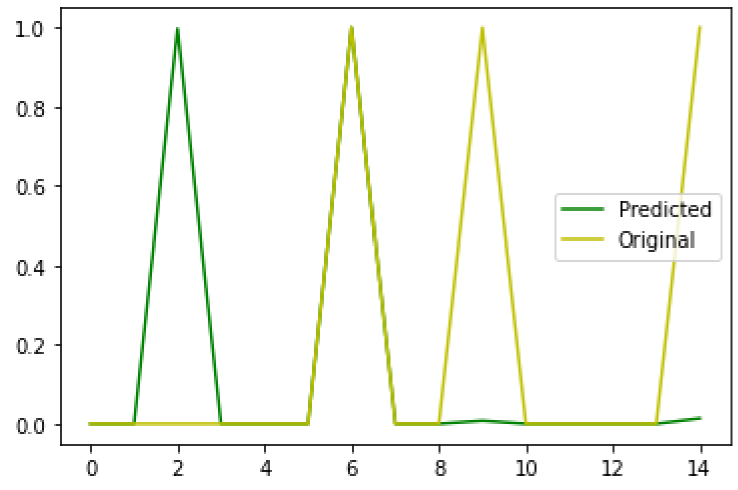
```
27/27 [==============================] - 1s 19ms/step
```

```
print(pred)
print(test)
```

```
[[0.   ]
 [0.   ]
 [0.997]
 [0.   ]
 [0.   ]
 [0.   ]
```

```
         [1.    ]
         [0.    ]
         [0.    ]
         [0.008]
         [0.    ]
         [0.    ]
         [0.    ]
         [0.    ]
         [0.014]]
        [[0]
         [0]
         [0]
         [0]
         [0]
         [0]
         [1]
         [0]
         [0]
         [1]
         [0]
         [0]
         [0]
         [0]
         [1]]
```

```
plt.plot(pred[:],color='g', label='Predicted')
plt.plot(test[:],color='y', label='Original')
plt.legend()
plt.show()
```


Predicted
Original