```
import numpy as np
import pandas as pd
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/spam.csv',delimiter=',',encoding='latin-1')

df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 5 columns):
 #   Column      Non-Null Count  Dtype
---  ------      --------------  -----
 0   v1          5572 non-null   object
 1   v2          5572 non-null   object
 2   Unnamed: 2  50 non-null     object
 3   Unnamed: 3  12 non-null     object
 4   Unnamed: 4  6 non-null      object
dtypes: object(5)
memory usage: 217.8+ KB

df.head()

     v1                                                 v2 Unnamed: 2
\
0   ham  Go until jurong point, crazy.. Available only ...        NaN

1   ham                      Ok lar... Joking wif u oni...        NaN

2  spam  Free entry in 2 a wkly comp to win FA Cup fina...        NaN

3   ham  U dun say so early hor... U c already then say...        NaN

4   ham  Nah I don't think he goes to usf, he lives aro...        NaN


  Unnamed: 3 Unnamed: 4
0        NaN        NaN
1        NaN        NaN
2        NaN        NaN
3        NaN        NaN
4        NaN        NaN
```

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

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)

# Tokenisation function
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)

#creating LSTM model
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 [==============================] - 12s 295ms/step - loss: 0.3334
- accuracy: 0.8749 - val_loss: 0.1872 - val_accuracy: 0.9293
Epoch 2/10
30/30 [==============================] - 8s 272ms/step - loss: 0.0840
- accuracy: 0.9784 - val_loss: 0.1138 - val_accuracy: 0.9684
Epoch 3/10
30/30 [==============================] - 10s 343ms/step - loss: 0.0359
- accuracy: 0.9905 - val_loss: 0.0903 - val_accuracy: 0.9705
Epoch 4/10
30/30 [==============================] - 9s 314ms/step - loss: 0.0305
- accuracy: 0.9908 - val_loss: 0.0834 - val_accuracy: 0.9768
Epoch 5/10
30/30 [==============================] - 8s 272ms/step - loss: 0.0224
- accuracy: 0.9934 - val_loss: 0.0980 - val_accuracy: 0.9778
Epoch 6/10
30/30 [==============================] - 8s 275ms/step - loss: 0.0208
- accuracy: 0.9942 - val_loss: 0.0839 - val_accuracy: 0.9778
Epoch 7/10
30/30 [==============================] - 12s 399ms/step - loss: 0.0142
- accuracy: 0.9958 - val_loss: 0.0849 - val_accuracy: 0.9778
Epoch 8/10
30/30 [==============================] - 8s 270ms/step - loss: 0.0127
- accuracy: 0.9968 - val_loss: 0.1274 - val_accuracy: 0.9747
Epoch 9/10
30/30 [==============================] - 8s 275ms/step - loss: 0.0088
- accuracy: 0.9982 - val_loss: 0.1120 - val_accuracy: 0.9778
Epoch 10/10
30/30 [==============================] - 8s 269ms/step - loss: 0.0072
- accuracy: 0.9984 - val_loss: 0.1315 - val_accuracy: 0.9778

<keras.callbacks.History at 0x7fb58dbfad10>
```

```
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 [==============================] - 2s 54ms/step - loss: 0.0611 -
accuracy: 0.9916
Accuracy: 0.992
```