```
import pandas as pd
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM, Dense, Dropout, Embedding
from tensorflow.keras.optimizers import RMSprop
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing import sequence
```

Data Pre-Processing

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.
```

```
df = pd.read_csv('/content/drive/MyDrive/spam.csv', delimiter=',', encoding='latin-1')
df.head()
```

	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

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

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
X = df.v2
Y = df.v1
encoder = LabelEncoder()
Y = encoder.fit_transform(Y)
Y = Y.reshape(-1,1)
```

```
tokenizer = Tokenizer(num_words=2000, lower=True)
tokenizer.fit_on_texts(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
X_train = sequence.pad_sequences(sequences, maxlen=200)
```

Create Model

```
model  = Sequential()
```

Add layers

```
model.add(Embedding(2000, 50, input_length=200))
model.add(LSTM(64))
model.add(Dense(256, activation="relu"))
model.add(Dropout(0.5))
model.add(Dense(1,activation="sigmoid"))
```

```
model.summary()
```

```
Model: "sequential_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_1 (Embedding)     (None, 200, 50)           100000

 lstm_1 (LSTM)               (None, 64)                29440

 dense_2 (Dense)             (None, 256)               16640

 dropout_1 (Dropout)         (None, 256)               0

 dense_3 (Dense)             (None, 1)                 257

=================================================================
Total params: 146,337
Trainable params: 146,337
Non-trainable params: 0
_________________________________________________________________
```

Compile the Model

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

Fit the Model

```
Epoch 1/10
28/28 [==============================] - 9s 326ms/step - loss: 0.0025 - accuracy: 0.
Epoch 2/10
28/28 [==============================] - 13s 484ms/step - loss: 0.0043 - accuracy: 0
Epoch 3/10
28/28 [==============================] - 11s 382ms/step - loss: 0.0020 - accuracy: 0
Epoch 4/10
28/28 [==============================] - 9s 314ms/step - loss: 0.0027 - accuracy: 0.
Epoch 5/10
28/28 [==============================] - 10s 370ms/step - loss: 0.0024 - accuracy: 0
Epoch 6/10
28/28 [==============================] - 9s 315ms/step - loss: 0.0022 - accuracy: 0.
Epoch 7/10
28/28 [==============================] - 9s 313ms/step - loss: 0.0029 - accuracy: 0.
Epoch 8/10
28/28 [==============================] - 9s 316ms/step - loss: 0.0019 - accuracy: 0.
Epoch 9/10
28/28 [==============================] - 9s 316ms/step - loss: 0.0015 - accuracy: 0.
Epoch 10/10
28/28 [==============================] - 9s 316ms/step - loss: 0.0018 - accuracy: 0.
<keras.callbacks.History at 0x7f6b210b7c50>
```

Save the Model

```
model.save("model.h5")
```

```
WARNING:tensorflow:Compiled the loaded model, but the compiled metrics have yet to b
```

Test the Model

```
test_sequences = tokenizer.texts_to_sequences(X_test)
X_test = sequence.pad_sequences(test_sequences, maxlen=200)
acc = model.evaluate(X_test, y_test)
```

```
def predict(message):
    txt = tokenizer.texts_to_sequences(message)
    txt = sequence.pad_sequences(txt, maxlen=200)
    preds = model.predict(txt)
    if preds > 0.5:
        print("Spam")
    else:
        print("Not Spam")
```

```
predict(["Sorry, I'll call later."])
```

```
1/1 [==============================] - 0s 473ms/step
Not Spam
```

ed & Free entry

```
1/1 [==============================] - 0s 24ms/step
Spam
```