

```

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 pad_sequences
from keras.utils import to_categorical
from keras.callbacks import EarlyStopping
READING DATASET

df = pd.read_csv('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
4      ham      Nah I don't think he goes to usf, he lives aro...      NaN      NaN      NaN
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
df.info()
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
df.groupby(['v1']).size()
v1
ham      4825
spam     747
dtype: int64
df.groupby(['v2']).size()
v2
<#> in mca. But not conform.

```

<#> mins but i had to stop somewhere first.

1

<DECIMAL> m but its not a common car here so its better to buy from china or asia. Or if i find it less expensive. I.ll holla 1

and picking them up from various points

1

came to look at the flat, seems ok, in his 50s? * Is away alot wiv work. Got woman coming at 6.30 too. 1

İİ still got lessons? İİ in sch? .. 1

İİ takin linear algebra today?

1

İİ thk of wat to eat tonight. 1

İİ v ma fan... 1

İİ wait 4 me in sch i finish ard 5..

1

Length: 5169, 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 = pad_sequences(sequences,maxlen=max_len)

CREATE MODEL AND ADD LAYERS

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)

COMPILE AND FIT THE MODEL

model.summary()

```

model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,
        validation_split=0.2)
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

```

Epoch 1/10
30/30 [=====] - 11s 286ms/step - loss: 0.3204 - accuracy:
0.8820 - val_loss: 0.1487 - val_accuracy: 0.9726
Epoch 2/10
30/30 [=====] - 8s 260ms/step - loss: 0.0889 - accuracy:
0.9791 - val_loss: 0.0641 - val_accuracy: 0.9831
Epoch 3/10
30/30 [=====] - 8s 263ms/step - loss: 0.0482 - accuracy:
0.9863 - val_loss: 0.0461 - val_accuracy: 0.9895
Epoch 4/10
30/30 [=====] - 8s 261ms/step - loss: 0.0361 - accuracy:
0.9894 - val_loss: 0.0363 - val_accuracy: 0.9895
Epoch 5/10
30/30 [=====] - 8s 258ms/step - loss: 0.0312 - accuracy:
0.9897 - val_loss: 0.0365 - val_accuracy: 0.9895
Epoch 6/10

```

30/30 [=====] - 10s 328ms/step - loss: 0.0223 - accuracy: 0.9923 - val_loss: 0.0418 - val_accuracy: 0.9863

Epoch 7/10

30/30 [=====] - 9s 284ms/step - loss: 0.0179 - accuracy: 0.9945 - val_loss: 0.0473 - val_accuracy: 0.9852

Epoch 8/10

30/30 [=====] - 10s 344ms/step - loss: 0.0123 - accuracy: 0.9950 - val_loss: 0.0599 - val_accuracy: 0.9895

Epoch 9/10

30/30 [=====] - 8s 271ms/step - loss: 0.0087 - accuracy: 0.9974 - val_loss: 0.0592 - val_accuracy: 0.9905

Epoch 10/10

30/30 [=====] - 8s 262ms/step - loss: 0.0082 - accuracy: 0.9971 - val_loss: 0.0490 - val_accuracy: 0.9884

SAVING THE MODEL

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

TEST THE MODEL

```
test_sequences = tok.texts_to_sequences(X_test)
```

```
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)
```

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

27/27 [=====] - 1s 24ms/step - loss: 0.0512 - accuracy: 0.9856

```
print('Test set\n Loss: {:.3f}\n Accuracy: {:.3f}'.format(accr[0],accr[1]))
```

Test set

Loss: 0.051

Accuracy: 0.986

TEAM MEMBERS:

Arul Murugan S

Nishanth R

Rithika P

Sunmathi M