# 1.Import requried library

import pandas as pd import numpy as np

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

import csv

import tensorflow as tf import pandas as pd import numpy as np

import matplotlib.pyplot as plt

from tensorflow.keras.preprocessing.text import Tokenizer

from tensorflow.keras.preprocessing.sequence import pad\_sequences import nltk

nltk.download('stopwords')

from nltk.corpus import stopwords

STOPWORDS = set(stopwords.words('english'))

[nltk\_data] Downloading package stopwords to /root/nltk\_data... [nltk\_data] Unzipping corpora/stopwords.zip.

# 2.Read dataset and do preprocessing

import pandas as pd import numpy as np import seaborn as sns

import matplotlib.pyplot as plt

%matplotlib inline

df = pd.read\_csv('/content/drive/MyDrive/spam.csv',delimiter=',',encoding=' latin-1')

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 |

1. ham U dun say so early hor... U c already then say... NaN
2. 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)

*#dropping unwanted columns*

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

*#Count of Spam and Ham values*

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

v1

ham 4825

spam 747

dtype: int64

*#Label Encoding target column*

X = df.v2 Y = df.v1

le = LabelEncoder()

Y = le.fit\_transform(Y) Y = Y.reshape(-1,1)

*# Test and train spilit*

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)

# 3.Create Model

# 4.Add layers (LSTM ,Dense-(Hidden Layers),Ouput)

*#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)

# Compile the model

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

# 6.Fit the model

model.fit(sequences\_matrix,Y\_train,batch\_size=128,epochs=10, validation\_split=0.2)

Epoch 1/10

30/30 [==============================] - 8s 263ms/step - loss: 0.0035

* accuracy: 0.9995 - val\_loss: 0.1122 - val\_accuracy: 0.9863 Epoch 2/10

30/30 [==============================] - 13s 454ms/step - loss: 0.0026

* accuracy: 0.9995 - val\_loss: 0.1018 - val\_accuracy: 0.9873 Epoch 3/10

30/30 [==============================] - 14s 468ms/step - loss: 0.0026

* accuracy: 0.9992 - val\_loss: 0.0911 - val\_accuracy: 0.9852 Epoch 4/10

30/30 [==============================] - 15s 493ms/step - loss: 0.0023

* accuracy: 0.9995 - val\_loss: 0.1240 - val\_accuracy: 0.9852 Epoch 5/10

30/30 [==============================] - 10s 349ms/step - loss: 0.0015

* accuracy: 0.9995 - val\_loss: 0.1336 - val\_accuracy: 0.9863 Epoch 6/10

30/30 [==============================] - 7s 249ms/step - loss: 0.0026

* accuracy: 0.9992 - val\_loss: 0.1339 - val\_accuracy: 0.9873 Epoch 7/10

30/30 [==============================] - 9s 289ms/step - loss:

3.0076e-04 - accuracy: 0.9997 - val\_loss: 0.1313 - val\_accuracy: 0.9873

Epoch 8/10

30/30 [==============================] - 8s 255ms/step - loss:

4.5712e-04 - accuracy: 0.9997 - val\_loss: 0.1547 - val\_accuracy: 0.9873

Epoch 9/10

30/30 [==============================] - 8s 253ms/step - loss:

1.8049e-04 - accuracy: 1.0000 - val\_loss: 0.1490 - val\_accuracy: 0.9863

Epoch 10/10

30/30 [==============================] - 11s 366ms/step - loss:

4.6702e-05 - accuracy: 1.0000 - val\_loss: 0.1521 - val\_accuracy: 0.9873

<keras.callbacks.History at 0x7f284144c9d0>

# 7.Save the model

model.save("model\_1")

WARNING:absl:Function `\_wrapped\_model` contains input name(s) InputLayer with unsupported characters which will be renamed to inputlayer in the SavedModel.

WARNING:absl:Found untraced functions such as lstm\_cell\_layer\_call\_fn, lstm\_cell\_layer\_call\_and\_return\_conditional\_losses while saving (showing 2 of 2). These functions will not be directly callable after loading.

# 8.Test the Model

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 49ms/step - loss: 0.2340 -

accuracy: 0.9809

Accuracy: 0.981

y\_pred = model.predict(test\_sequences\_matrix) print(y\_pred[25:40].round(3))

27/27 [==============================] - 1s 23ms/step [[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]

[0.]]

print(Y\_test[25:40]) [[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]

[0]]