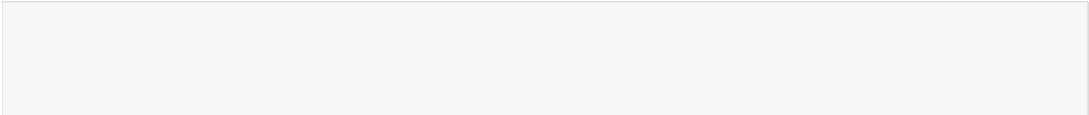
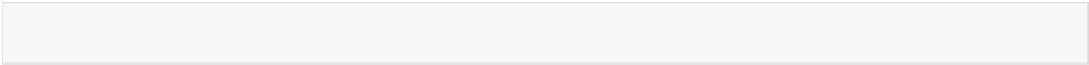
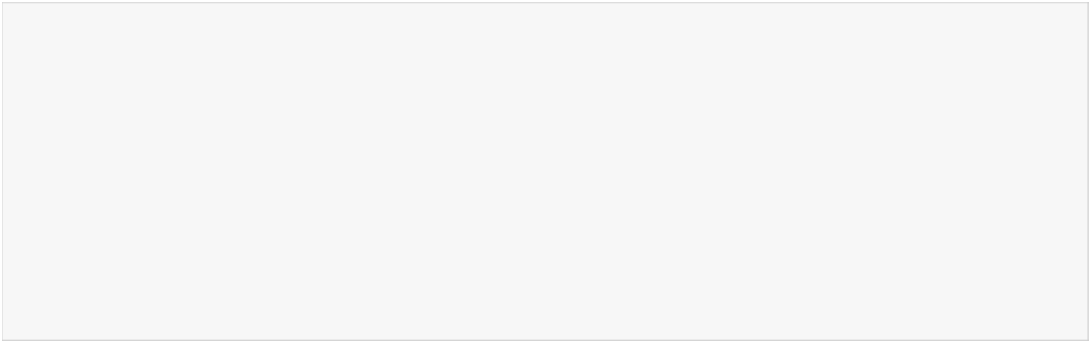
PNT2022TMID36340



# Dataset has been downloaded and saved

## Import required Libraries

In [ ]:

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 Adam

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

# Read the Dataset

In [ ]:

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

Out[ ]:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | **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 |

# Preprocessing the Dataset

In [ ]:

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

In [ ]:

from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator

In [ ]:

X = df.v2 Y = df.v1

le = LabelEncoder()

Y = le.fit\_transform(Y)





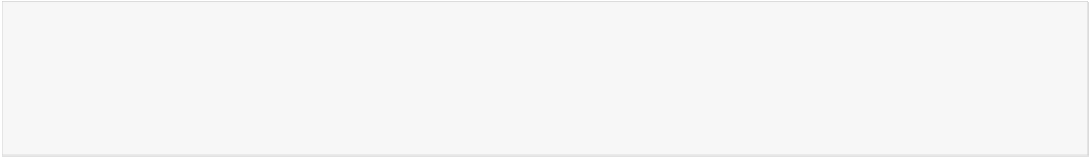
Y = Y.reshape(-1, 1)

In [ ]:



X\_train,X\_test,Y\_train,Y\_test = train\_test\_split(X,Y,test\_size=0.25)

In [ ]:



max\_words = 1000

max\_len = 150

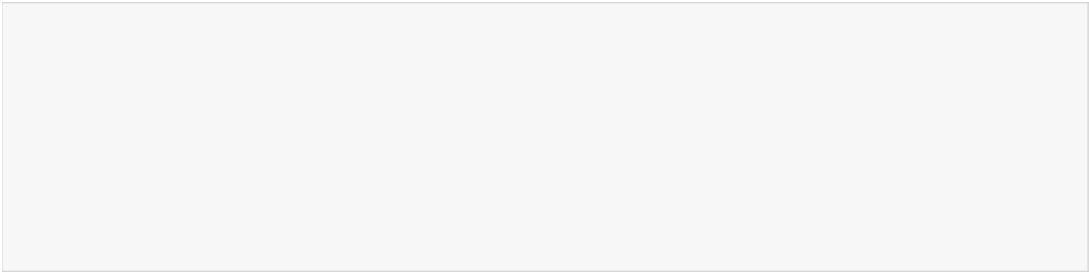
tok = Tokenizer(num\_words = max\_words) tok.fit\_on\_texts(X\_train)

sequences = tok.texts\_to\_sequences(X\_train)

sequences\_matrix = pad\_sequences(sequences, maxlen = max\_len)

## Create Model and Add Layers

In [ ]:



inputs = Input(shape=[max\_len])

layer = Embedding(max\_words,50,input\_length=max\_len)(inputs) layer = LSTM(128)(layer)

layer = Dense(128)(layer)

layer = Activation('relu')(layer) layer = Dropout(0.5)(layer)

layer = Dense(1)(layer)

layer = Activation('sigmoid')(layer)

model = Model(inputs=inputs,outputs=layer)

model.summary()

Model: "model"

Layer (type) Output Shape Param #

=================================================================

input\_1 (InputLayer) [(None, 150)] 0

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| embedding (Embedding) | (None, | 150, | 50) | 50000 |
| lstm (LSTM) | (None, | 128) |  | 91648 |
| dense (Dense) | (None, | 128) |  | 16512 |
| activation (Activation) | (None, | 128) |  | 0 |
| dropout (Dropout) | (None, | 128) |  | 0 |
| dense\_1 (Dense) | (None, | 1) |  | 129 |
| activation\_1 (Activation) | (None, | 1) |  | 0 |

=================================================================

Total params: 158,289

Trainable params: 158,289

Non-trainable params: 0

# Create Model

In [ ]:

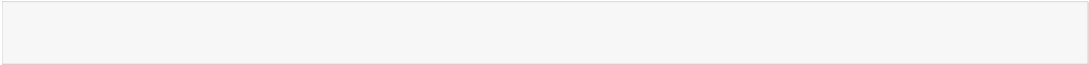
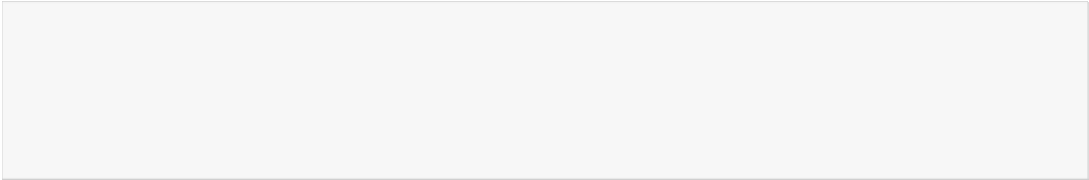
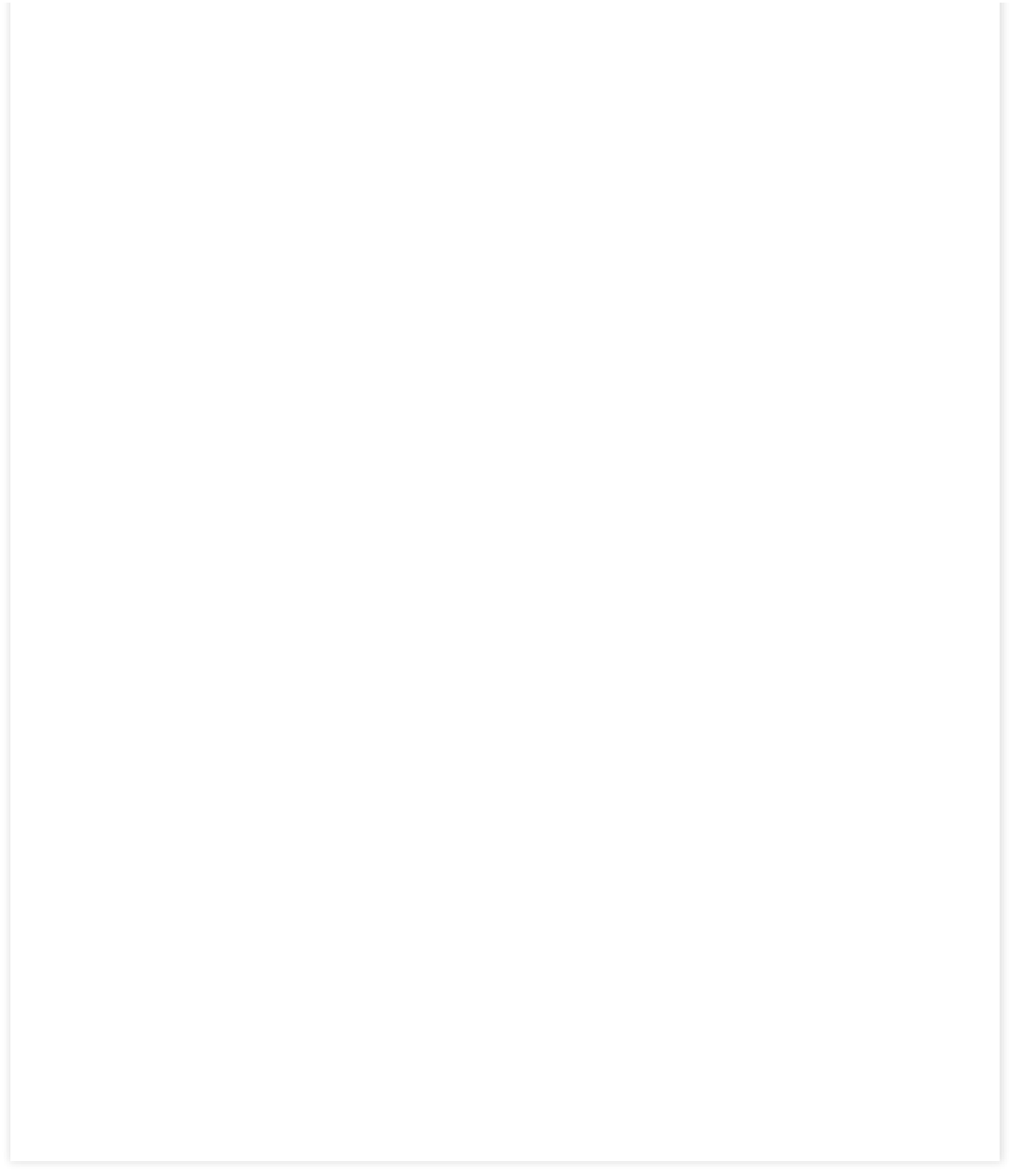


model = RNN()

## Compiling the Model

In [ ]:

model.compile(loss='binary\_crossentropy',optimizer=Adam(),metrics=['accuracy'])



## Training the Model

In [ ]:

model.fit(

sequences\_matrix, Y\_train, batch\_size = 128, epochs=10,

validation\_split = 0.2,

callbacks=[EarlyStopping(monitor = 'val\_loss', min\_delta = 0.0001)])

Epoch 1/10

27/27 [==============================] - 7s 277ms/step - loss: 0.0092 - accuracy: 0.9982

* val\_loss: 0.0804 - val\_accuracy: 0.9821 Epoch 2/10

27/27 [==============================] - 8s 295ms/step - loss: 0.0069 - accuracy: 0.9982

* val\_loss: 0.0843 - val\_accuracy: 0.9821 Out[ ]:

<keras.callbacks.History at 0x7fcc62da8710>

## Save the model

In [ ]:

model.save('Spam\_sms\_classifier.h5')

## Test the model

In [ ]:

test\_sequences = tok.texts\_to\_sequences(X\_test)

test\_sequences\_matrix = pad\_sequences(test\_sequences, maxlen = max\_len)

In [ ]:

accr = model.evaluate(test\_sequences\_matrix, Y\_test)

44/44 [==============================] - 1s 23ms/step - loss: 0.0523 - accuracy: 0.9892

In [ ]:

print('Test set\n Loss: {:0.3f}\n Accuracy: {:0.3f}'.format(accr[0],accr[1])) Test set

Loss: 0.052

Accuracy: 0.989