**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** to\_categorical

**from** keras.callbacks **import** EarlyStopping

**import** tensorflow

**%matplotlib** inline

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

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

sns**.**countplot(df**.**v1)

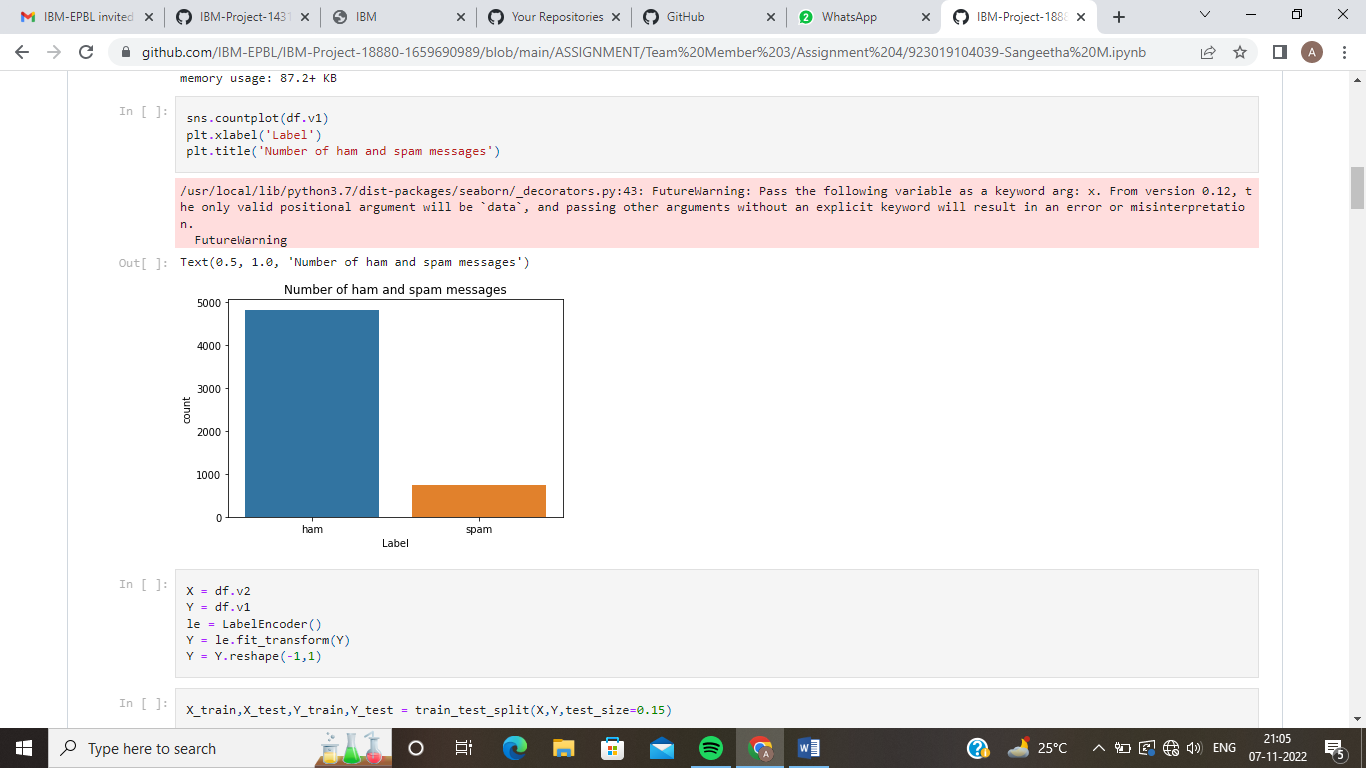
plt**.**xlabel('Label')

plt**.**title('Number of ham and spam messages')

/usr/local/lib/python3.7/dist-packages/seaborn/\_decorators.py:43: FutureWarning: Pass the following variable as a keyword arg: x. From version 0.12, the only valid positional argument will be `data`, and passing other arguments without an explicit keyword will result in an error or misinterpretation.

FutureWarning

Text(0.5, 1.0, 'Number of ham and spam messages')



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)

**def** RNN():

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)

**return** model

model **=** RNN()

model**.**summary()

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

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

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

df**.**columns

Index(['v1', 'v2'], dtype='object')

data**=**df**.**rename(

{

"v1":"Category",

"v2":"Message"

},

axis**=**1

)

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

data["Message Length"]**=**data["Message"]**.**apply(len)

fig**=**plt**.**figure(figsize**=**(12,8))

sns**.**histplot(

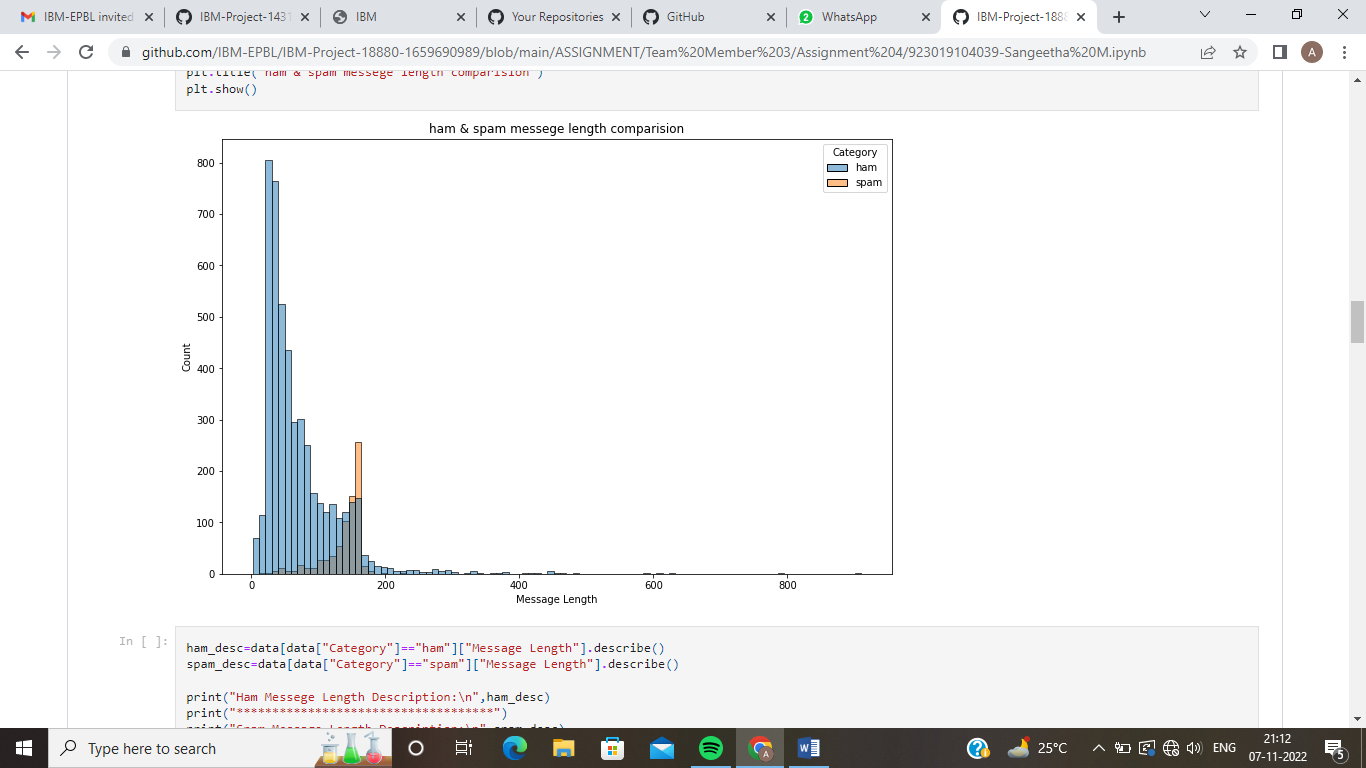
x**=**data["Message Length"],

hue**=**data["Category"]

)

plt**.**title("ham & spam messege length comparision")

plt**.**show()



ham\_desc**=**data[data["Category"]**==**"ham"]["Message Length"]**.**describe()

spam\_desc**=**data[data["Category"]**==**"spam"]["Message Length"]**.**describe()

print("Ham Messege Length Description:\n",ham\_desc)

print("\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

print("Spam Message Length Description:\n",spam\_desc)

data**.**describe(include**=**"all")

|  | **Category** | **Message** | **Message Length** |
| --- | --- | --- | --- |
| **count** | 5572 | 5572 | 5572.000000 |
| **unique** | 2 | 5169 | NaN |
| **top** | ham | Sorry, I'll call later | NaN |
| **freq** | 4825 | 30 | NaN |
| **mean** | NaN | NaN | 80.118808 |
| **std** | NaN | NaN | 59.690841 |
| **min** | NaN | NaN | 2.000000 |
| **25%** | NaN | NaN | 36.000000 |
| **50%** | NaN | NaN | 61.000000 |
| **75%** | NaN | NaN | 121.000000 |
| **max** | NaN | NaN | 910.000000 |

data["Category"]**.**value\_counts()

ham 4825

spam 747

Name: Category, dtype: int64

sns**.**countplot(

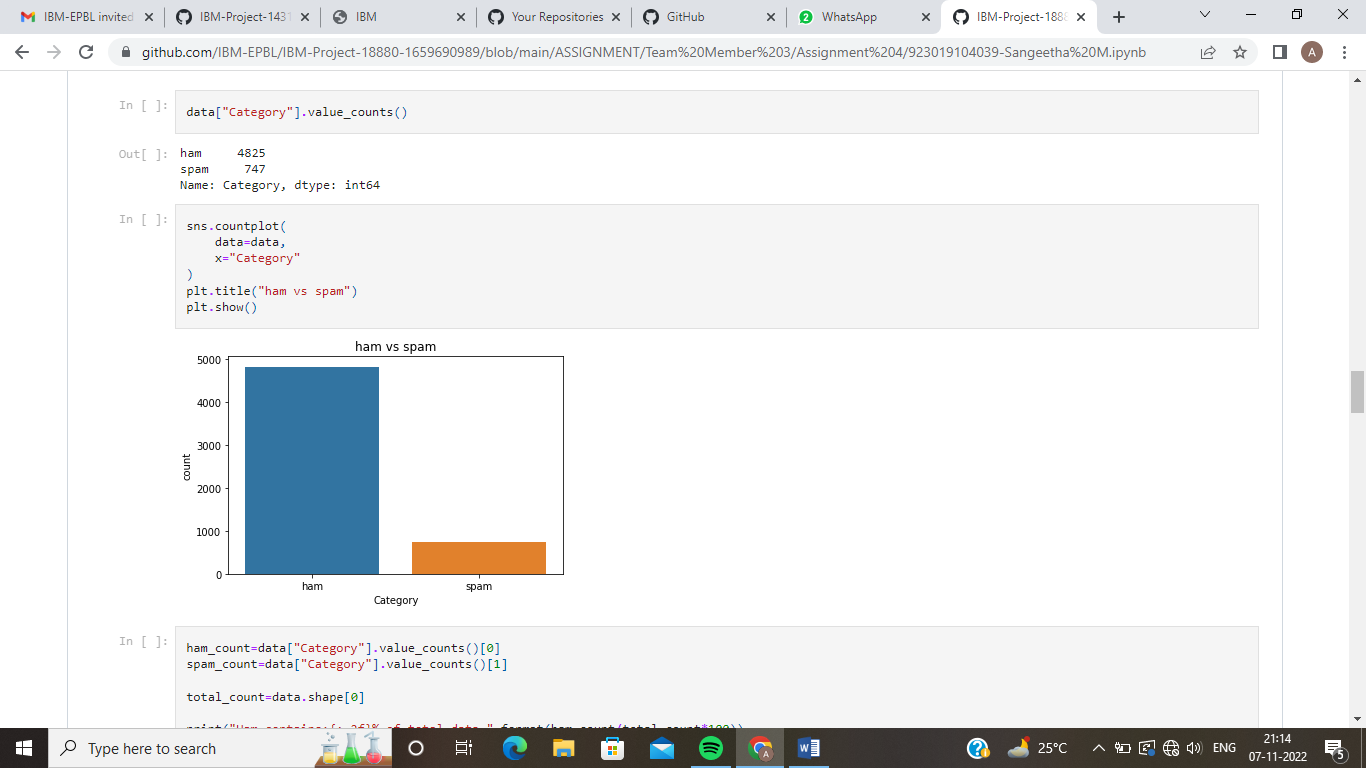
data**=**data,

x**=**"Category"

)

plt**.**title("ham vs spam")

plt**.**show()



ham\_count**=**data["Category"]**.**value\_counts()[0]

spam\_count**=**data["Category"]**.**value\_counts()[1]

total\_count**=**data**.**shape[0]

print("Ham contains:{:.2f}% of total data."**.**format(ham\_count**/**total\_count**\***100))

print("Spam contains:{:.2f}% of total data."**.**format(spam\_count**/**total\_count**\***100))

Ham contains:86.59% of total data.

Spam contains:13.41% of total data.

minority\_len**=**len(data[data["Category"]**==**"spam"])

majority\_len**=**len(data[data["Category"]**==**"ham"])

minority\_indices**=**data[data["Category"]**==**"spam"]**.**index

majority\_indices**=**data[data["Category"]**==**"ham"]**.**index

random\_majority\_indices**=**np**.**random**.**choice(

majority\_indices,

size**=**minority\_len,

replace**=False**

)

undersampled\_indices**=**np**.**concatenate([minority\_indices,random\_majority\_indices])

df**=**data**.**loc[undersampled\_indices]

df**=**df**.**sample(frac**=**1)

df**=**df**.**reset\_index()

df**=**df**.**drop(

columns**=**["index"],

)

df**.**shape

(1494, 3)

df["Category"]**.**value\_counts()

spam 747

ham 747

Name: Category, dtype: int64

sns**.**countplot(

data**=**df,

x**=**"Category"

)

plt**.**title("ham vs spam")

plt**.**show()



df**.**head()

|  | **Category** | **Message** | **Message Length** |
| --- | --- | --- | --- |
| **0** | spam | Dear Voucher Holder, To claim this weeks offer... | 152 |
| **1** | ham | I cant pick the phone right now. Pls send a me... | 51 |
| **2** | spam | TheMob>Hit the link to get a premium Pink Pant... | 149 |
| **3** | spam | URGENT This is our 2nd attempt to contact U. Y... | 136 |
| **4** | spam | 449050000301 You have won a å£2,000 price! To ... | 70 |

df["Label"]**=**df["Category"]**.**map(

{

"ham":0,

"spam":1

}

)

df**.**head()

|  | **Category** | **Message** | **Message Length** | **Label** |
| --- | --- | --- | --- | --- |
| **0** | spam | Dear Voucher Holder, To claim this weeks offer... | 152 | 1 |
| **1** | ham | I cant pick the phone right now. Pls send a me... | 51 | 0 |
| **2** | spam | TheMob>Hit the link to get a premium Pink Pant... | 149 | 1 |
| **3** | spam | URGENT This is our 2nd attempt to contact U. Y... | 136 | 1 |
| **4** | spam | 449050000301 You have won a å£2,000 price! To ... | 70 | 1 |

**import** re

**import** nltk

**from** nltk.corpus **import** stopwords

**from** nltk.stem **import** PorterStemmer

stemmer**=**PorterStemmer()

corpus**=**[]

**for** message **in** df["Message"]:

message**=**re**.**sub("[^a-zA-Z]"," ",message)

message**=**message**.**lower()

message**=**message**.**split()

**from** tensorflow.keras.preprocessing.text **import** one\_hot

vocab\_size**=**10000

oneHot\_doc**=**[one\_hot(words,n**=**vocab\_size)

**for** words **in** corpus

]

df["Message Length"]**.**describe()

count 1494.000000

mean 105.296519

std 57.379967

min 2.000000

25% 49.000000

50% 120.000000

75% 153.000000

max 450.000000

Name: Message Length, dtype: float64

fig**=**plt**.**figure(figsize**=**(12,8))

sns**.**kdeplot(

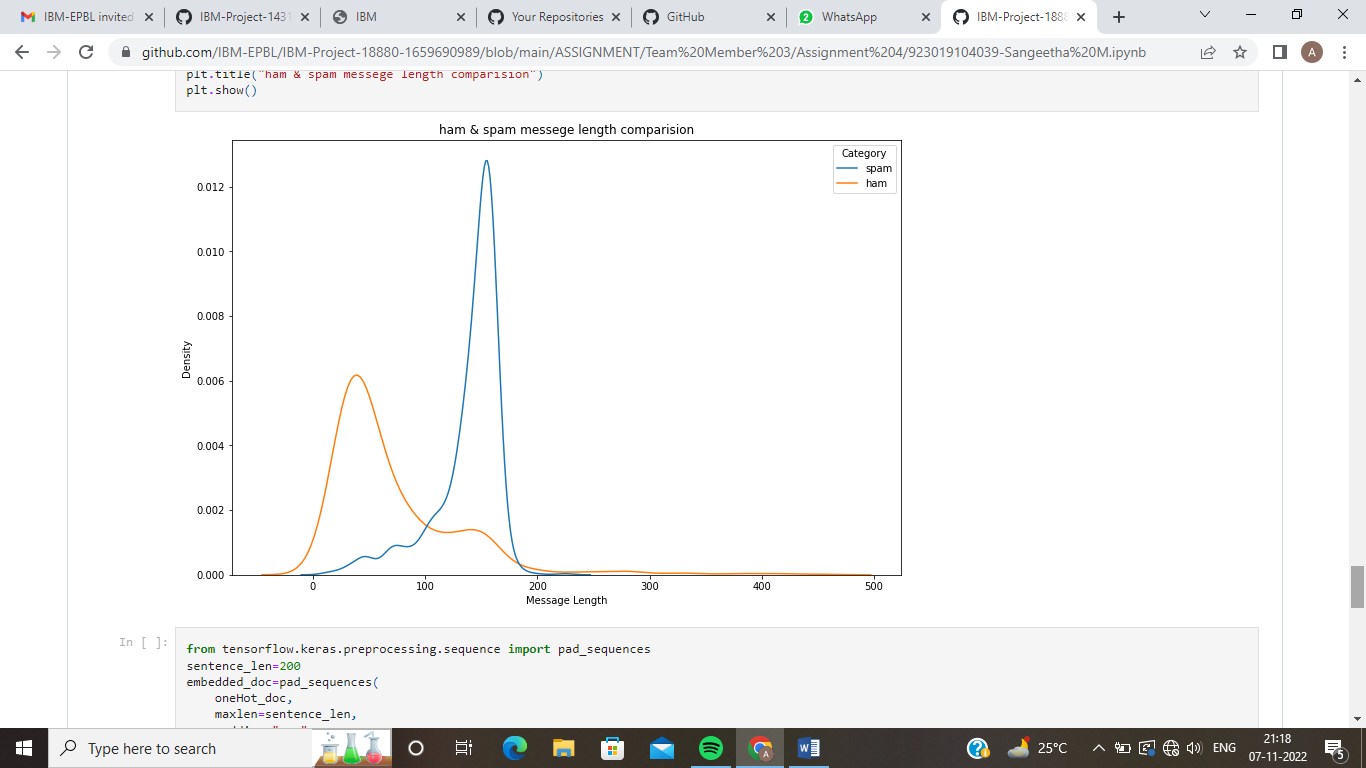
x**=**df["Message Length"],

hue**=**df["Category"]

)

plt**.**title("ham & spam messege length comparision")

plt**.**show()



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

sentence\_len**=**200

embedded\_doc**=**pad\_sequences(

oneHot\_doc,

maxlen**=**sentence\_len,

padding**=**"pre"

)

extract\_features**=**pd**.**DataFrame(

data**=**embedded\_doc

)

target**=**df["Label"]

df\_final**=**pd**.**concat([extract\_features,target],axis**=**1)

df\_final**.**head()

|  | **0** | **1** | **2** | **3** | **4** | **5** | **6** | **7** | **8** | **9** | **...** | **191** | **192** | **193** | **194** | **195** | **196** | **197** | **198** | **199** | **Label** |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| **0** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 |
| **1** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 0 |
| **2** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 |
| **3** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 |
| **4** | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | ... | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | NaN | 1 |

5 rows × 201 columns

X**=**df\_final**.**drop("Label",axis**=**1)

y**=**df\_final["Label"]

**from** sklearn.model\_selection **import** train\_test\_split

X,

y,

random\_state**=**37,

test\_size**=**0.2

)

X\_train,X\_val,y\_train,y\_val**=**train\_test\_split(

X\_trainval,

y\_trainval,

random\_state**=**37,

test\_size**=**0.2

)

model **=** RNN()

model**.**summary()

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

Model: "model\_1"

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

Layer (type) Output Shape Param #

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

inputs (InputLayer) [(None, 150)] 0

embedding\_1 (Embedding) (None, 150, 50) 50000

lstm\_1 (LSTM) (None, 64) 29440

FC1 (Dense) (None, 256) 16640

activation\_2 (Activation) (None, 256) 0

dropout\_1 (Dropout) (None, 256) 0

out\_layer (Dense) (None, 1) 257

activation\_3 (Activation) (None, 1) 0

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

Total params: 96,337

Trainable params: 96,337

Non-trainable params: 0

\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_