# Assignment – 4

**Problem Statement :- SMS SPAM Classification**

|  |  |
| --- | --- |
| Assignment Date | 21 October 2022 |
| Student Name | Nishanthini S |
| Student Roll Number | 921319104136 |
| Maximum Marks | 2 Marks |

# 1. Import Required Libraries

**import** numpy **as** np

**import** pandas **as** pd

**import** matplotlib.pyplot **as** plt

**import** seaborn **as** sns

**import** keras

**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, pad\_sequences

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

# 2. Read dataset and pre processing

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**.**shape

(5572, 2)

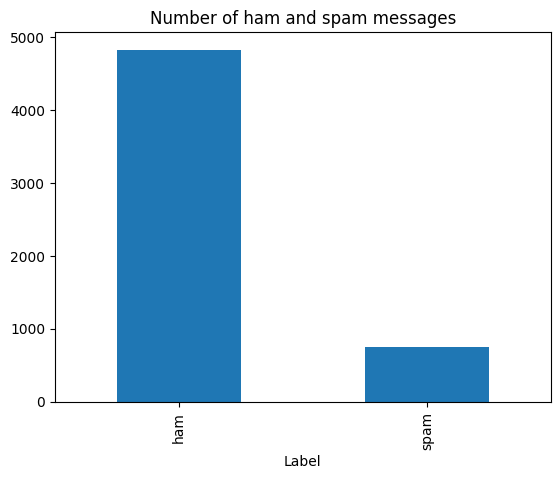
*#plot the ham and spam messages to understand the distribution*

df['v1']**.**value\_counts()**.**plot(kind**=**'bar')

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

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

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



X **=** df**.**v2

Y **=** df**.**v1

*#label encoding for Y*

le **=** LabelEncoder()

Y **=** le**.**fit\_transform(Y)

Y **=** Y**.**reshape(**-**1,1)

# 3. Train-test split

*#split into train and test sets*

X\_train,X\_test,Y\_train,Y\_test **=** train\_test\_split(X,Y,test\_size**=**0.20)

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 **=** keras**.**utils**.**pad\_sequences(sequences,maxlen**=**max\_len)

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

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)

# 5. Create Model

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

# 6. Compile the Model

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

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

# 7.Fit the Model

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

28/28 [==============================] - 4s 148ms/step - loss: 0.0415 - accuracy: 0.9879 - val\_loss: 0.0635 - val\_accuracy: 0.9809

Epoch 2/10

28/28 [==============================] - 4s 144ms/step - loss: 0.0254 - accuracy: 0.9927 - val\_loss: 0.0641 - val\_accuracy: 0.9843

# 8. Save the Model

model**.**save('spam\_lstm\_model.h5')

# 9.Test the Model

*#processing test data*

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

test\_sequences\_matrix **=** keras**.**utils**.**pad\_sequences(test\_sequences,maxlen**=**max\_len)

*#evaluation of our model*

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

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

35/35 [==============================] - 1s 16ms/step - loss: 0.0910 - accuracy: 0.9785

Test set

Loss: 0.091

Accuracy: 0.978