# Assignment 4

# Tasks

Perform the Below Tasks to complete the assignment:-

* Download the Dataset:- [Dataset](https://storage.googleapis.com/kagglesdsdata/datasets/483/982/spam.csv?X-Goog-Algorithm=GOOG4-RSA-SHA256&X-Goog-Credential=gcp-kaggle-com%40kaggle-161607.iam.gserviceaccount.com%2F20221027%2Fauto%2Fstorage%2Fgoog4_request&X-Goog-Date=20221027T040626Z&X-Goog-Expires=259200&X-Goog-SignedHeaders=host&X-Goog-Signature=)
* Import required library
* Read dataset and do pre-processing
* Create Model
* Add Layers (LSTM, Dense-(Hidden Layers), Output)
* Compile the Model
* Fit the Model
* Save The Model
* Test The Model

In [1]:

**from** google.colab **import** drive

drive**.**mount('/content/drive')

Mounted at /content/drive

# 1. Load the dataset

In [5]:

dataset\_location **=** "/content/drive/MyDrive/IBM/spam.csv"

# 2. Import the library

In [20]:

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

**import** nltk

**import** re

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

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

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

**from** nltk.translate.ribes\_score **import** word\_rank\_alignment

**from** numpy.lib.shape\_base **import** split

**from** sklearn **import** preprocessing

**from** sklearn.feature\_extraction.text **import** CountVectorizer

**from** tensorflow.keras.models **import** Sequential

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

**from** keras.layers **import** LSTM,Dense,Dropout,Input,Embedding,Activation,Flatten

**from** keras.models **import** Model

**import** nltk

# 3. Read dataset and do preprocessing

In [7]:

data **=** pd**.**read\_csv(dataset\_location,encoding **=** "ISO-8859-1")

In [9]:

data**.**drop(["Unnamed: 2","Unnamed: 3","Unnamed: 4"],axis **=** 1,inplace **=** **True**)

data**.**head()

Out[9]:

|  | **v1** | **v2** |
| --- | --- | --- |
| **0** | ham | Go until jurong point, crazy.. Available only ... |
| **1** | ham | Ok lar... Joking wif u oni... |
| **2** | spam | Free entry in 2 a wkly comp to win FA Cup fina... |
| **3** | ham | U dun say so early hor... U c already then say... |
| **4** | ham | Nah I don't think he goes to usf, he lives aro... |

In [12]:

nltk**.**download('stopwords',quiet**=True**)

nltk**.**download('all',quiet**=True**)

Out[12]:

True

In [13]:

ps **=** PorterStemmer()

input **=** []

In [15]:

**for** i **in** range(0,5572):

v2 **=** data['v2'][i]

*#removing punctuation*

v2 **=** re**.**sub('[^a-zA-Z]',' ',v2)

*#converting to lower case*

v2 **=** v2**.**lower()

*#splitting the sentence*

v2 **=** v2**.**split()

*#removing the stopwords and stemming*

v2 **=** [ps**.**stem(word) **for** word **in** v2 **if** **not** word **in** set(stopwords**.**words('english'))]

v2 **=** ' '**.**join(v2)

input**.**append(v2)

In [21]:

*#creating document term matrix*

cv **=** CountVectorizer(max\_features**=**2000)

x **=** cv**.**fit\_transform(input)**.**toarray()

x**.**shape

Out[21]:

(5572, 2000)

In [22]:

le **=** preprocessing**.**LabelEncoder()

data['v1'] **=** le**.**fit\_transform(data['v1'])

data['v1']**.**unique()

Out[22]:

array([0, 1])

In [23]:

y **=** data['v1']**.**values

In [24]:

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

In [25]:

x\_train,x\_test,y\_train,y\_test **=** train\_test\_split(x,y,test\_size **=** 0.4)

# 4. Model building - Adding layers, Compiling model and saving model

In [16]:

model **=** Sequential()

In [26]:

model**.**add(Dense(1565,activation **=** "relu"))

model**.**add(Dense(3000,activation **=** "relu"))

model**.**add(Dense(1,activation **=** "sigmoid"))

model**.**add(Flatten())

In [27]:

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

In [ ]:

model**.**fit(x\_train,y\_train,epochs **=** 15)

In [30]:

model**.**save("spam-message-classifier.h5")

# 5. Testing the model

In [32]:

ham **=** "im donee. come pick me up"

spam **=** "WINNER$$$$ SMS REPLY 'WIN'"

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

message

Out[32]:

'WINNER SMS REPLY WIN '

# Testing with spam message

In [33]:

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

message **=** [ps**.**stem(word) **for** word **in** message **if** **not** word **in** set(stopwords**.**words('english')) ]

message **=** ' '**.**join(message)

In [37]:

message1 **=** cv**.**transform([message])

message1

Out[37]:

<1x2000 sparse matrix of type ''

with 4 stored elements in Compressed Sparse Row format>

In [48]:

TruePredction **=** model**.**predict(message1**.**astype(float))

1/1 [==============================] - 0s 13ms/step

In [50]:

TruePredction **>** 0.5

Out[50]:

array([[ True]])

# Testing with normal message

In [41]:

msg **=** re**.**sub('[^a-zA-Z]',' ',ham)

msg

Out[41]:

'im donee come pick me up'

In [42]:

msg **=** msg**.**split()

msg **=** [ps**.**stem(word) **for** word **in** msg **if** **not** word **in** set(stopwords**.**words('english'))]

msg **=** ' '**.**join(msg)

In [43]:

msg

Out[43]:

'im done come pick'

In [44]:

cv**.**transform([msg])

Out[44]:

<1x2000 sparse matrix of type ''

with 4 stored elements in Compressed Sparse Row format>

In [45]:

FalsePredection **=** model**.**predict(cv**.**transform([msg]))

1/1 [==============================] - 0s 110ms/step

In [47]:

FalsePredection **>** 0.5

Out[47]:

array([[False]])