```
import pandas as pd
import nltk
import re


data = pd.read_csv("/content/spam.csv",encoding = "ISO-8859-1")


data.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

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

	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...

TEXT PROCESSING

```
nltk.download('stopwords')
nltk.download('all')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
[nltk_data] Downloading collection 'all'
[nltk_data]    |
[nltk_data]    | Downloading package abc to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/abc.zip.
[nltk_data]    | Downloading package alpino to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/alpino.zip.
[nltk_data]    | Downloading package averaged_perceptron_tagger to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping taggers/averaged_perceptron_tagger.zip.
```

```
[nltk_data]    | Downloading package averaged_perceptron_tagger_ru to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping
[nltk_data]    |       taggers/averaged_perceptron_tagger_ru.zip.
[nltk_data]    | Downloading package basque_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/basque_grammars.zip.
[nltk_data]    | Downloading package biocreative_ppi to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/biocreative_ppi.zip.
[nltk_data]    | Downloading package bllip_wsj_no_aux to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping models/bllip_wsj_no_aux.zip.
[nltk_data]    | Downloading package book_grammars to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping grammars/book_grammars.zip.
[nltk_data]    | Downloading package brown to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown.zip.
[nltk_data]    | Downloading package brown_tei to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/brown_tei.zip.
[nltk_data]    | Downloading package cess_cat to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_cat.zip.
[nltk_data]    | Downloading package cess_esp to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cess_esp.zip.
[nltk_data]    | Downloading package chat80 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/chat80.zip.
[nltk_data]    | Downloading package city_database to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/city_database.zip.
[nltk_data]    | Downloading package cmudict to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/cmudict.zip.
[nltk_data]    | Downloading package comparative_sentences to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/comparative_sentences.zip.
[nltk_data]    | Downloading package comtrans to /root/nltk_data...
[nltk_data]    | Downloading package conll2000 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2000.zip.
[nltk_data]    | Downloading package conll2002 to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/conll2002.zip.
[nltk_data]    | Downloading package conll2007 to /root/nltk_data...
[nltk_data]    | Downloading package crubadan to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/crubadan.zip.
[nltk_data]    | Downloading package dependency_treebank to
[nltk_data]    |     /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dependency_treebank.zip.
[nltk_data]    | Downloading package dolch to /root/nltk_data...
[nltk_data]    |   Unzipping corpora/dolch.zip.
```

```
from nltk.corpus import stopwords
from nltk.stem.porter import PorterStemmer


ps = PorterStemmer()
input = []


data.shape
```

```
(5572, 2)
```

```
from nltk.translate.ribes_score import word_rank_alignment
from numpy.lib.shape_base import split

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)


input
```

```
['go jurong point crazi avail bugi n great world la e buffet cine got amor wat',
 'ok lar joke wif u oni',
 'free entri wkli comp win fa cup final tkt st may text fa receiv entri question
std txt rate c appli',
 'u dun say earli hor u c alreadi say',
 'nah think goe usf live around though',
 'freemsg hey darl week word back like fun still tb ok xxx std chg send rcv',
 'even brother like speak treat like aid patent',
 'per request mell mell oru minnaminungint nurungu vettam set callertun caller
press copi friend callertun',
 'winner valu network custom select receivea prize reward claim call claim code kl
valid hour',
 'mobil month u r entitl updat latest colour mobil camera free call mobil updat co
free',
 'gonna home soon want talk stuff anymor tonight k cri enough today',
 'six chanc win cash pound txt csh send cost p day day tsandc appli repli hl
info',
 'urgent week free membership prize jackpot txt word claim c www dbuk net lccltd
pobox ldnw rw',
 'search right word thank breather promis wont take help grant fulfil promis
wonder bless time',
 'date sunday',
 'xxxmobilemovieclub use credit click wap link next txt messag click http wap
xxxmobilemovieclub com n qjkgighjjgcbl',
 'oh k watch',
 'eh u rememb spell name ye v naughti make v wet',
 'fine way u feel way gota b',
 'england v macedonia dont miss goal team news txt ur nation team eg england tri
wale scotland txt poboxox w wq',
 'serious spell name',
 'go tri month ha ha joke',
 'pay first lar da stock comin',
```

```
 'aft finish lunch go str lor ard smth lor u finish ur lunch alreadi',
 'fffffffff alright way meet sooner',
 'forc eat slice realli hungri tho suck mark get worri know sick turn pizza lol',
 'lol alway convinc',
 'catch bu fri egg make tea eat mom left dinner feel love',
 'back amp pack car let know room',
 'ahhh work vagu rememb feel like lol',
 'wait still clear sure sarcast x want live us',
 'yeah got v apologet n fallen actin like spoilt child got caught till go badli
cheer',
 'k tell anyth',
 'fear faint housework quick cuppa',
 'thank subscript rington uk mobil charg month pleas confirm repli ye repli
charg',
 'yup ok go home look time msg xuhui go learn nd may lesson',
 'oop let know roommat done',
 'see letter b car',
 'anyth lor u decid',
 'hello saturday go text see decid anyth tomo tri invit anyth',
 'pl go ahead watt want sure great weekend abiola',
 'forget tell want need crave love sweet arabian steed mmmmmm yummi',
 'rodger burn msg tri call repli sm free nokia mobil free camcord pleas call
deliveri tomorrow',
 'see',
 'great hope like man well endow lt gt inch',
```

```
#creating document term matrix

from sklearn.feature_extraction.text import CountVectorizer
cv = CountVectorizer(max_features=2000)


cv

     CountVectorizer(max_features=2000)


x = cv.fit_transform(input).toarray()


x
x.shape

     (5572, 2000)


from sklearn import preprocessing
le = preprocessing.LabelEncoder()

data['v1'] = le.fit_transform(data['v1'])
data['v1'].unique()

     array([0, 1])


y = data['v1'].values


y
```

```
     array([0, 0, 1, ..., 0, 0, 0])
```

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

```
#y = data.iloc[:,1:2].values
#y
```

```
from sklearn.model_selection import train_test_split
```

```
x_train,x_test,y_train,y_test = train_test_split(x,y,test_size = 0.4)
```

Model -ANN

```
from keras.layers import LSTM,Dense,Dropout,Input,Embedding,Activation,Flatten
from keras.models import Model
```

```
from tensorflow.keras.models import Sequential
```

```
model = Sequential()
```

```
#model.add(Embedding(5000,64,input_length = 2000))
#model.add(LSTM(100))
#model.add(Dense(1565,activation = "relu"))
#model.add(Dense(3000,activation = "relu"))
#model.add(Dense(1,activation = "sigmoid"))
```

```
model.add(Dense(1565,activation = "relu"))
model.add(Dense(3000,activation = "relu"))
model.add(Dense(1,activation = "sigmoid"))
model.add(Flatten())
#model.summary()
```

```
#ip = Input(shape = [2000])
#layer = Embedding(1000,50,input_length = 150)(ip)
#layer = Dense(1000)(ip)
#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 = ip,outputs = layer)
```

```
#model = Sequential()
#model.add(LSTM(100))
#model.add(Dense(1465,activation = "relu"))
#model.add(Dense(3000,activation = "relu"))
#model.add(Dense(1,activation = "sigmoid"))


model.compile(optimizer = "adam",loss = "binary_crossentropy", metrics = ["accuracy"])


model.fit(x_train,y_train,epochs = 15)
```

```
Epoch 1/15
105/105 [==============================] - 9s 78ms/step - loss: 0.1483 - accuracy: 0
Epoch 2/15
105/105 [==============================] - 8s 76ms/step - loss: 0.0123 - accuracy: 0
Epoch 3/15
105/105 [==============================] - 8s 74ms/step - loss: 0.0055 - accuracy: 0
Epoch 4/15
105/105 [==============================] - 8s 74ms/step - loss: 0.0028 - accuracy: 0
Epoch 5/15
105/105 [==============================] - 8s 77ms/step - loss: 0.0028 - accuracy: 0
Epoch 6/15
105/105 [==============================] - 8s 76ms/step - loss: 0.0028 - accuracy: 0
Epoch 7/15
105/105 [==============================] - 8s 75ms/step - loss: 0.0026 - accuracy: 0
Epoch 8/15
105/105 [==============================] - 8s 75ms/step - loss: 0.0026 - accuracy: 0
Epoch 9/15
105/105 [==============================] - 8s 75ms/step - loss: 0.0025 - accuracy: 0
Epoch 10/15
105/105 [==============================] - 8s 74ms/step - loss: 0.0025 - accuracy: 0
Epoch 11/15
105/105 [==============================] - 8s 73ms/step - loss: 0.0025 - accuracy: 0
Epoch 12/15
105/105 [==============================] - 8s 75ms/step - loss: 0.0023 - accuracy: 0
Epoch 13/15
105/105 [==============================] - 8s 74ms/step - loss: 0.0024 - accuracy: 0
Epoch 14/15
105/105 [==============================] - 8s 76ms/step - loss: 0.0023 - accuracy: 0
Epoch 15/15
105/105 [==============================] - 8s 74ms/step - loss: 0.0022 - accuracy: 0
<keras.callbacks.History at 0x7f01af652410>
```

```
model.save("SMS_Spam_Classifier.h5")


ham = "im donee. come pick me up"
spam = "WINNER$$$$ SMS REPLY 'WIN'"
message = re.sub('[^a-zA-Z]',' ',spam)
message
```

```
'WINNER     SMS REPLY  WIN '
```

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

```
message = ' '.join(message)
```

```
message
```

```
'winner sm repli win'
```

```
cv.transform([message])
```

```
<1x2000 sparse matrix of type '<class 'numpy.int64'>'
        with 4 stored elements in Compressed Sparse Row format>
```

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

```
<1x2000 sparse matrix of type '<class 'numpy.int64'>'
        with 4 stored elements in Compressed Sparse Row format>
```

```
import numpy as np
pred = model.predict(message1.astype(float))
```

```
1/1 [==============================] - 0s 166ms/step
```

```
pred
```

```
array([[0.23785314]], dtype=float32)
```

```
pred > 0.5
```

```
array([[False]])
```

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

```
'im donee  come pick me up'
```

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

```
msg
```

```
'im done come pick'
```

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

```
<1x2000 sparse matrix of type '<class 'numpy.int64'>'
        with 4 stored elements in Compressed Sparse Row format>
```

```
pred1 = model.predict(cv.transform([msg]))
```

```
1/1 [==============================] - 0s 146ms/step
```

```
pred1
```

```
array([[2.672558e-06]], dtype=float32)
```

```
pred1 > 0.5
```

```
array([[False]])
```