Assignment -IV STM for Text Classification

Assignment Date	03 November 2022
Student Name	R.Jeyapriya
Student Roll Number	9517201906018
Maximum Marks	2 Marks

#Import necessary libraries

import numpy as np

import pandas as pd

import matplotlib.pyplot as plt

import seaborn as sns

%matplotlib inline

from sklearn.model_selection import train_test_split

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

from sklearn.preprocessing import LabelEncoder

from keras.preprocessing.text import Tokenizer

from keras.models import Sequential

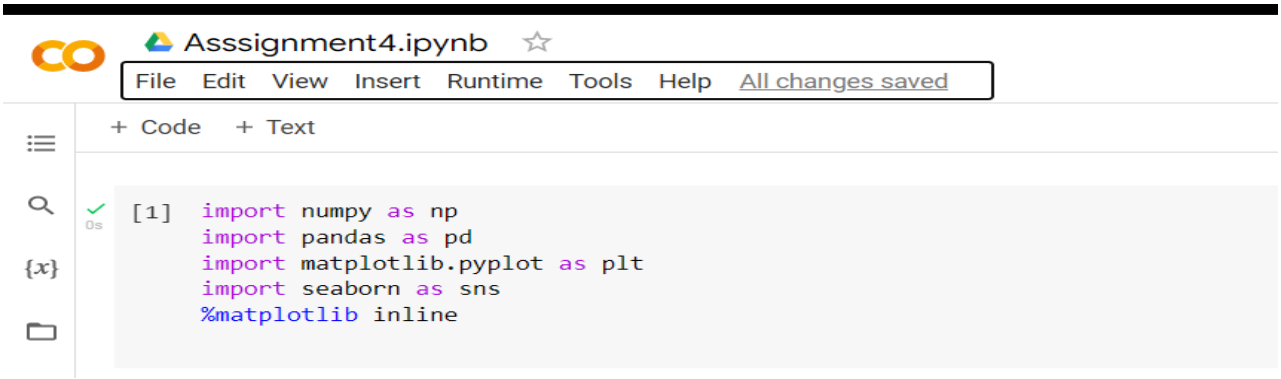
from tensorflow.keras.preprocessing import sequence

from tensorflow.keras.utils import to_categorical

from keras.callbacks import EarlyStopping

from tensorflow.keras.optimizers import RMSprop

from keras_preprocessing.sequence import pad_sequences



```
from sklearn.model_selection import train_test_split
from keras.layers import Dense , LSTM , Embedding , Dropout , Activation , Flatten
from sklearn.preprocessing import LabelEncoder
from keras.preprocessing.text import Tokenizer
from keras.models import Sequential
from tensorflow.keras.preprocessing import sequence
from tensorflow.keras.utils import to_categorical
from keras.callbacks import EarlyStopping
from tensorflow.keras.optimizers import RMSprop
from keras_preprocessing.sequence import pad_sequences
```

#Read dataset and do pre-processing

data = pd.read_csv('/content/spam.csv',delimiter=',',encoding='latin-1')

data

#Information about dataset

data.describe().T

data.shape

#Check if there is any missing values

data.isnull().sum()

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

#Visualize the dataset

sns.countplot(data.v1)

#Preprocess using Label Encoding

X = data.v2

Y = data.v1

le = LabelEncoder()

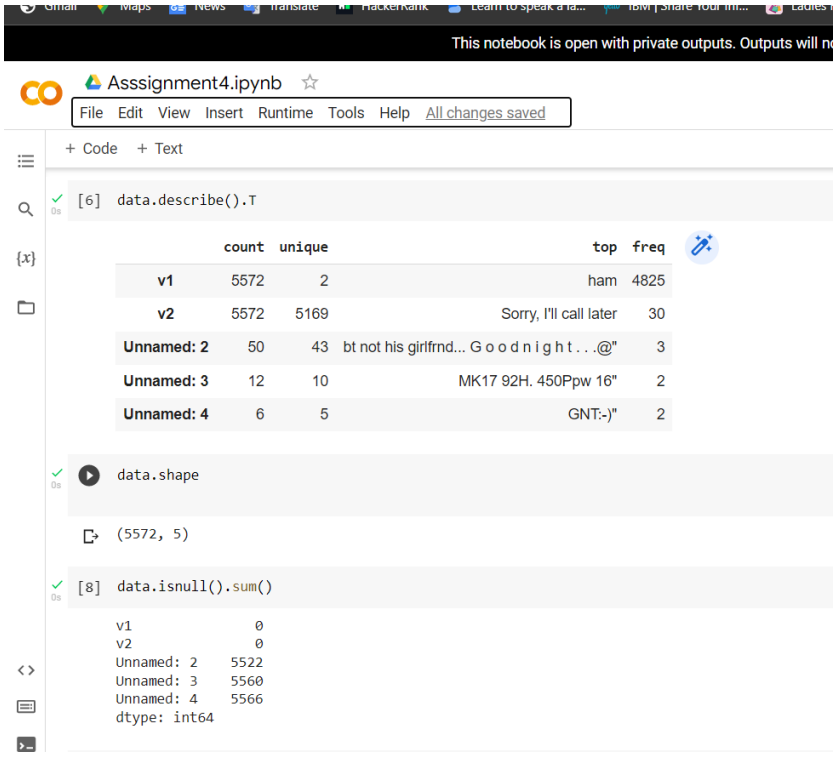
Y = le.fit_transform(Y)

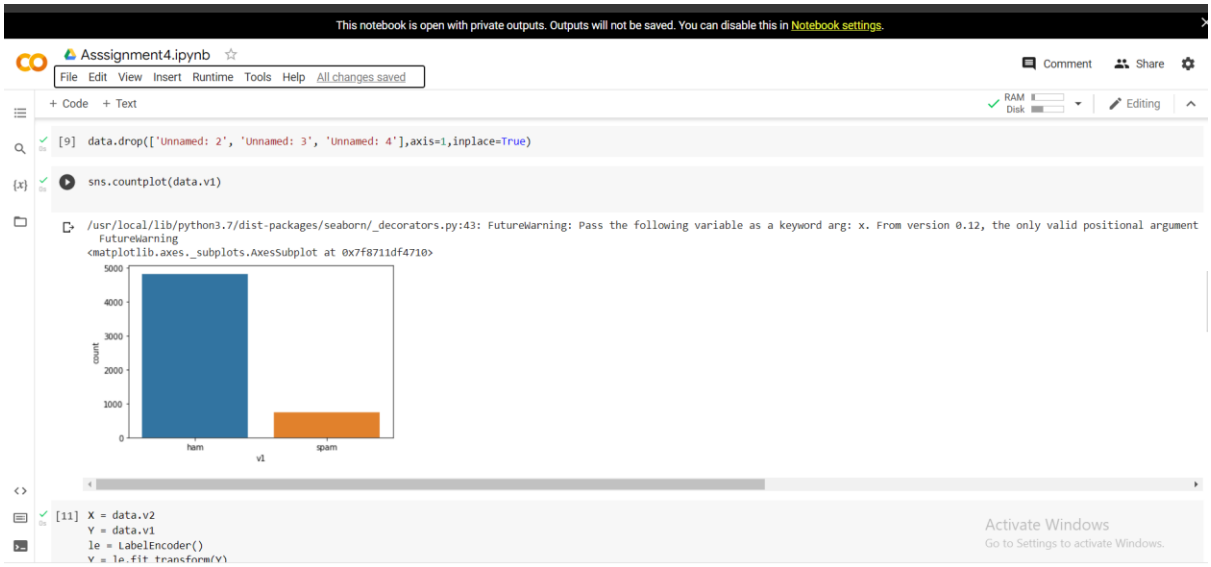
Y = Y.reshape(-1,1)

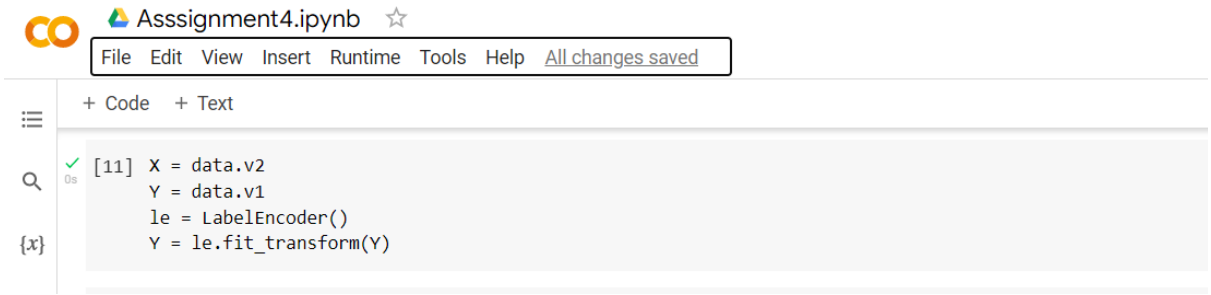
```
[5] data = pd.read_csv('/content/drive/MyDrive/spam.csv',delimiter=',',encoding='latin-1')
    data
```

	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

✓ 0s completed at 7:14 PM







#Create Model and Add Layers (LSTM, Dense-(Hidden Layers), Output)

#Splitting into training and testing data

X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)

max_word = 1000

max_len = 250

token = Tokenizer(num_words = max_word)

token.fit_on_texts(X_train)

sequences = token.texts_to_sequences(X_train)

seq_matrix = sequence.pad_sequences(sequences , maxlen = max_len)

#Creating the model

model = Sequential()

model.add(Embedding(max_word , 32 , input_length = max_len))

model.add(LSTM(64))

model.add(Flatten())

model.add(Dense(250, activation='relu'))

model.add(Dropout(0.5))

model.add(Dense(120, activation='relu'))

model.add(Dense(1, activation='sigmoid'))

```
[13] X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size = 0.2)
```

```
max_word = 1000
max_len = 250
token = Tokenizer(num_words = max_word)
token.fit_on_texts(X_train)
sequences = token.texts_to_sequences(X_train)
seq_matrix = sequence.pad_sequences(sequences , maxlen = max_len)
```

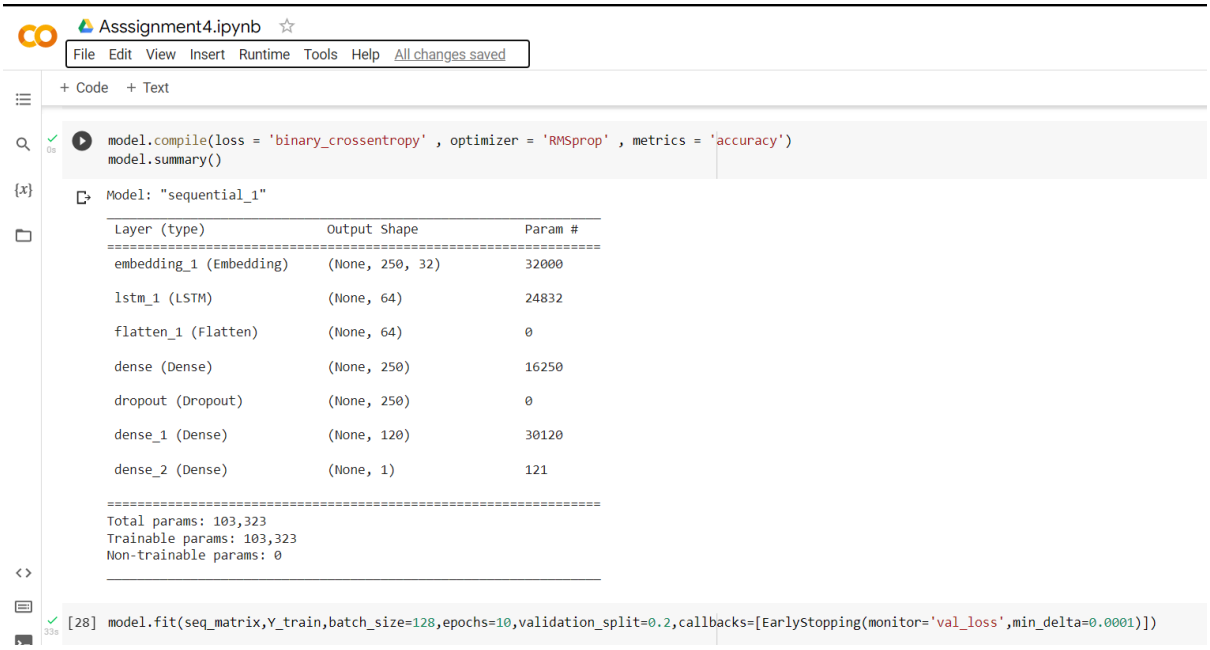
```
[26] model = Sequential()
     model.add(Embedding(max_word , 32 , input_length = max_len))
     model.add(LSTM(64))
     model.add(Flatten())
     model.add(Dense(250, activation='relu'))
     model.add(Dropout(0.5))
     model.add(Dense(120, activation='relu'))
     model.add(Dense(1, activation='sigmoid'))
```

0s completed at 7:14 PM

#compile the model

model.compile(loss = 'binary_crossentropy' , optimizer = 'RMSprop' , metrics = 'accuracy')

model.summary()



#Fit the model

model.fit(seq_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2,callbacks=[EarlySt

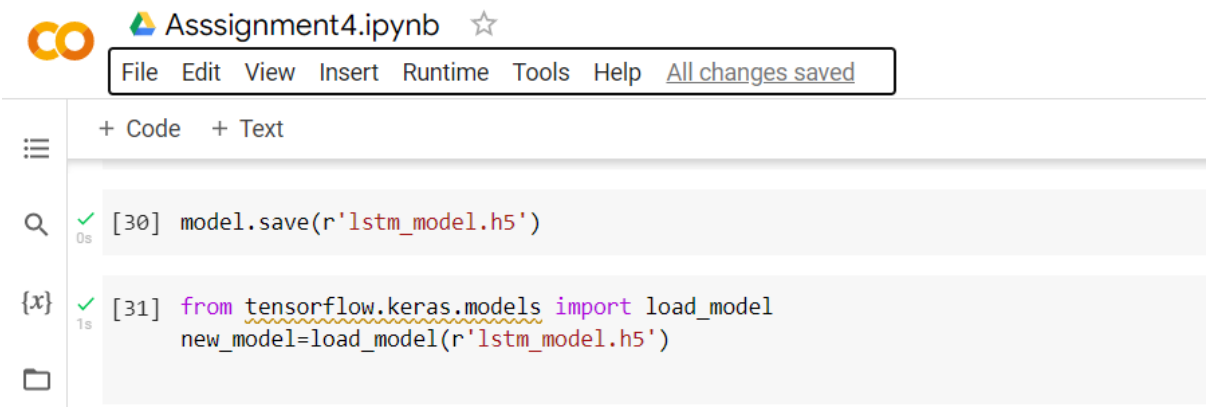
opping(monitor='val_loss',min_delta=0.0001)])

test_seq = token.texts_to_sequences(X_test)

test_seq_matrix = sequence.pad_sequences(test_seq,maxlen=max_len)

#Save the model

model.save(r'lstm_model.h5')



#Test the model:

from tensorflow.keras.models import load_model

new_model=load_model(r'lstm_model.h5')

new_model.evaluate(test_seq_matrix,Y_test)

scores = model.evaluate(test_seq_matrix, Y_test, verbose=0)

scores

print("Accuracy: %.2f%%" % (scores[1]*100))