Assignment - 4

Python Programming

Assignment Date	27 October 2022
Student Name	Ms.D.Janani
Student Roll Number	510119106004
Maximum Marks	2 Marks

QUESTION: SMS Spam Classification

```
import pandas as pd
import numpy as np
import sklearn as sk
import tensorflow
```

```
data=pd.read_csv("/content/drive/MyDrive/LSTM/simple-lstm-for-text-cla
ssification.ipynb")
```

```
data.head()
```

```
Empty DataFrame
Columns:
[{"cells":[{"metadata":{"_cell_guid":"79c7e3d0-c299-4dcb-8224-4455121e
e9b0", _uuid:"d629ff2d2480ee46fbb7e2d37f6b5fab8052498a",
collapsed:true}, cell_type:"markdown", source:"# Import the necessary
libraries", outputs:[], execution_count:null},
{"metadata":{"trusted":true,
_uuid:"d6fb32fd69316596e236eab5fb8cf77c848508c3"}, cell_type:"code",
source:"import pandas as pd\nimport numpy as np\nimport
matplotlib.pyplot as plt\nimport seaborn as sns\nfrom
sklearn.model_selection import train_test_split\nfrom
sklearn.preprocessing import LabelEncoder\nfrom keras.models import
Model\nfrom keras.layers import LSTM,  Activation,  Dense,  Dropout,
Input,  Embedding\nfrom keras.optimizers import RMSprop\nfrom
keras.preprocessing.text import Tokenizer\nfrom keras.preprocessing
import sequence\nfrom keras.utils import to_categorical\nfrom
keras.callbacks import EarlyStopping\n%matplotlib inline",
execution_count:null, outputs:[]},
{"metadata":{"_uuid":"f674695f1742479cefdeec0e81ab469f7b6ec90f"},
cell_type:"markdown".1, source:"### Load the data into Pandas
dataframe", outputs:[].1, execution_count:null}.1,
{"metadata":{"trusted":true.1,
_uuid:"aca2f1d9da3f35d104763166fe4d25448410d8f2"}, cell_type:"code".1,
source:"df = pd.read_csv('../input/spam.csv', delimiter=', ',
encoding='latin-1')\ndf.head()", execution_count:null.1,
outputs:[]}.1,
{"metadata":{"_uuid":"53083ccecf39523cff290495a6cc768061ba9b46"},
cell_type:"markdown".2, source:"Drop the columns that are not required
for the neural network.", outputs:[].2, execution_count:null}.2,
{"metadata":{"scrolled":true, trusted:true,
_uuid:"95a8b5d6f19cf42d4f55c6d2842faf1d0d55c1d0"}, cell_type:"code".2,
source:"df.drop(['Unnamed: 2',  'Unnamed: 3',  'Unnamed: 4'], axis=1,
inplace=True)\ndf.info()", execution_count:null.2, outputs:[]}.2,
{"metadata":{"_uuid":"3c7060084470000f39a2dcc15b656586dcd6e9fd"},
cell_type:"markdown".3, source:"Understand the distribution better.",
outputs:[].3, execution_count:null}.3, {"metadata":{"trusted":true.2,
_uuid:"a12002f521dd8eaeb0f69a932cbf23815ffd09d7"}, cell_type:"code".3,
source:"sns.countplot(df.v1)\nplt.xlabel('Label')\nplt.title('Number
of ham and spam messages')", execution_count:null.3, outputs:[]}.3,
{"metadata":{"_uuid":"353a8191f86c3a22843a729b5d4a5acefbf94be8"},
cell_type:"markdown".4, source:"* Create input and output vectors.\n*
Process the labels.", outputs:[].4, execution_count:null}.4,
{"metadata":{"collapsed":true, trusted:true.1,
_uuid:"a1a345c1683e2fcc7173ecae867a5da87f2dde24"}, cell_type:"code".4,
source:"X = df.v2\nY = df.v1\nle = LabelEncoder()\nY =
le.fit_transform(Y)\nY = Y.reshape(-1, 1)", execution_count:null.4,
outputs:[]}.4,
{"metadata":{"_uuid":"150e244a39b814d8a41bbe0e419bc5f28e457dd6"},
cell_type:"markdown".5, source:"Split into training and test data.",
outputs:[].5, execution_count:null}.5, {"metadata":{"scrolled":true.1,
trusted:true.2, collapsed:true,
_uuid:"aa3386af09469682c66cc53a1830a4e42f0e70b6"}, cell_type:"code".5,
source:"X_train, X_test, Y_train, Y_test = train_test_split(X, Y,
test_size=0.15)", execution_count:null.5, outputs:[]}.5,
{"metadata":{"_uuid":"c5378d55c271e01480c1ac07f94ff99a80f900d6"},
cell_type:"markdown".6, source:"### Process the data\n* Tokenize the
data and convert the text to sequences.\n* Add padding to ensure that
all the sequences have the same shape.\n* There are many ways of
taking the *max_len* and here an arbitrary length of 150 is chosen.",
outputs:[].6, execution_count:null}.6, {"metadata":{"trusted":true.3,
collapsed:true.1, _uuid:"bdca14f2b8cd7bd7cb5ee66fd40ea522217c03c6"},
cell_type:"code".6, source:"max_words = 1000\nmax_len = 150\ntok =
Tokenizer(num_words=max_words)\ntok.fit_on_texts(X_train)\nsequences =
tok.texts_to_sequences(X_train)\nsequences_matrix =
sequence.pad_sequences(sequences, ...]
Index: []

[0 rows x 202 columns]
```

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

```
train_set
```

```
array([], shape=(0, 1), dtype=object)
```

```
len(train_set)
```

```
0
```

```
x_train = []
y_train = []
```

```
for i in range(60,0):
  x_train.append(train_set[i-60:i,0])
  y_train.append(train_set[i,0])
```

```
x_train
```

```
[array([], dtype=object)]
```

```
len(x_train)
```

```
1
```

```
x_train.ndim
```

```
2
```

```
x_train = np.reshape(x_train,(1190,60,1))
```

```
x_train.ndim
```

```
3
```

```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import LSTM,Dense
```

```
model = Sequential()
```

```
model.add(LSTM(units = 50,input_shape
=(x_train.shape[1],1),return_Sequential = True ))

model.add(LSTM(units = 50, return_sequences = True))

model.add(LSTM(units = 50, return_sequences =  True))

model.add(LSTM(units = 50))

model.add(LSTM(units = 1))

model.compile(optimizer = "adam", loss = "mse")

model.fit(x_train, y_train, epouchs = 5, branch_size = 64))

                                   ^

pred = model.predict(x_train)

y_train

[]
```

```
#RMSE
error = y_train - pred

SE = error*error

MSE = SE.mean()

MSE

model.add(Dense(units = 1565, activation "relu"))
model.add(Dense(units = 2000, activation "relu"))

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

model.compile(optimizer = 'adam'), loss =
'binary_crossentrophy',metrics = ['accuracy'])

model.fit(x_train,y_train,epouchs = 10)


text1 = "I hate this food and it was very bad"
text2 = "I love this food and it is very tasty"


text1.split()

['I', 'hate', 'this', 'food', 'and', 'it', 'was', 'very', 'bad']


def preprocessing(text1)
  text = re.sub('[^a-zA-Z]'),' ',teext1)
  text = text.split()
  text = [ps.stem(word) for word in text if not word in set
(stopwords.words('english'))]
  text ' '.join(text)
  return text


preprocessing (text1)

model.save(“text classification”)

x_train()

[array([], dtype=object)]

y_train()

[]
```