Assignment - 4

Python Programming

Assignment Date	1 November 2022
Student Name	J. Flora
Student Roll Number	953719106015
Maximum Marks	2 Marks

Question 1:

Download the Dataset

Solution:

Data set has been download and stored.

Question 2:

Import required library

Solution:

```
import pandas as pd

import seaborn as sns

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 pad_sequences

from keras.utils import to_categorical
```

```
from keras.callbacks import EarlyStopping
```

Question 3:

Read dataset and do pre-processing

Solution:

```
df = pd.read_csv('../Datasets/spam.csv',delimiter = ','encoding = 'latin-1')
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis = 1,inplace = True)
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1, 1)
X_train, X_test, Y_train, Y_test = train_test_split(X,Y,test_size = 0.15)
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 = pad_sequences(sequences, maxlen = max_len)
```

Question 4:

Create Model

Solution:

```
model = RNN()
```

Question 5:

Add Layers (LSTM, Dense-(Hidden Layers), Output)

Solution:

```
def RNN():
    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)
    model = Model(inputs = inputs, outputs = layer)
    return model
```

Question 6:

Compile the Model

Solution:

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

Question 7:

Fit the Model

Solution:

```
model.fit(
    sequences_matrix,
```

```
    Y_train,
    batch_size = 128,
    epochs=10,
    validation_split = 0.2,
    callbacks=[EarlyStopping(monitor = 'val_loss', min_delta = 0.0001)])
```

Question 8:

Save the Model

Solution:

```
model.save('./spam.h5')
```

Question 9:

Test the Model

Solution:

```
test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences, maxlen = max_len)
accr = model.evaluate(test_sequences_matrix, Y_test)
print('Test set\n  Loss: {:0.3f}\n  Accuracy: {:0.3f}'.format(accr[0],accr[1]))
```