EMERGING METHODS FOR EARLY DETECTION OF FOREST FIRES

ASSIGNMENT 4

Date	29 October 2022
Team ID	PNT2022TMID07051
Project Name	Emerging Methods for Early Detection of Forest Fires

Import Libraries:

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
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 Adam
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
```

Read the dataset:

```
df = pd.read_csv('/content/sample_data/spam.csv', delimiter=',',encoding='latin-1')
df.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
4	ham	Nah I don't think he goes to usf, he lives aro...	NaN	NaN	NaN

Pre-processing The Dataset:

```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
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=0.25)
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)
```

Create Model:

```
inputs = Input(shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
```

Add Layers:

```
layer = LSTM(128)(layer)
layer = Dense(128)(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1.5)(layer)
layer = Activation('sigmoid')(layer)
model = Model(inputs=inputs,outputs=layer)
model.summary()
```

Model: "model"

Layer (type)	Output Shape	Param #
input_1 (InputLayer)	[(None, 150)]	0
embedding (Embedding)	(None, 150, 50)	50000
lstm (LSTM)	(None, 128)	91648
dense (Dense)	(None, 128)	16512
activation (Activation)	(None, 128)	0
dropout (Dropout)	(None, 128)	0
dense_1 (Dense)	(None, 1)	129
activation_1 (Activation)	(None, 1)	0

Total params: 158,289
Trainable params: 158,289
Non-trainable params: 0

Compile the Model:

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

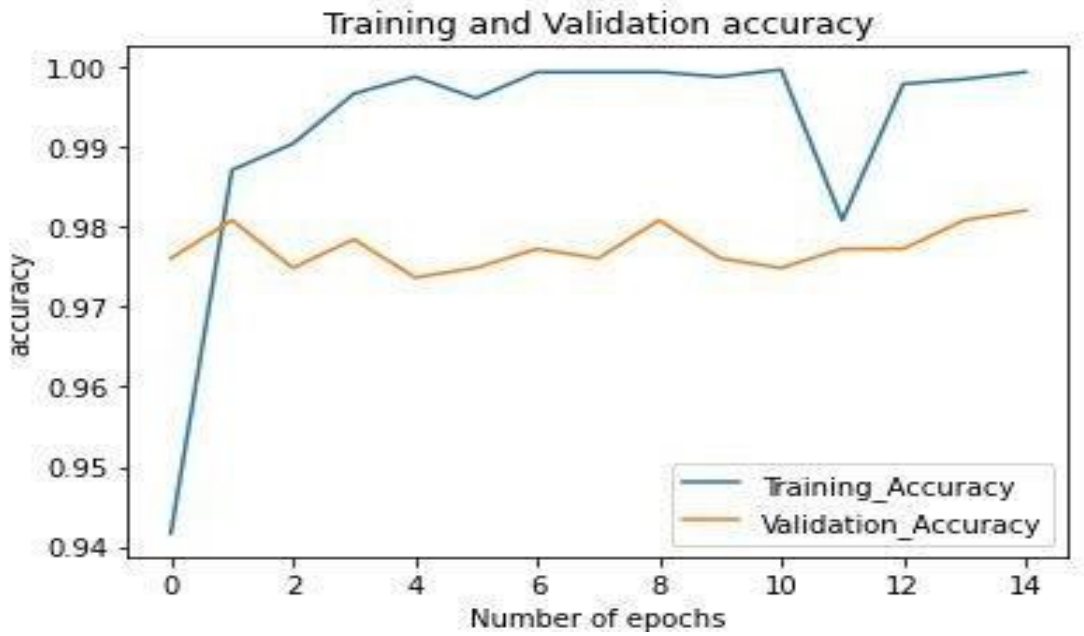
Fit the model:

```
history =
model.fit(sequences_matrix,Y_train,batch_size=20,epochs=15,validation_split=0.2
)
Epoch 1/15
168/168 [==============================] - 47s 264ms/step - loss: 0.187
1 - accuracy: 0.9417 - val_loss: 0.0686 - val_accuracy: 0.9761
Epoch 2/15
168/168 [==============================] - 44s 263ms/step - loss: 0.042
1 - accuracy: 0.9871 - val_loss: 0.0647 - val_accuracy: 0.9809
Epoch 3/15
168/168 [==============================] - 31s 182ms/step - loss: 0.027
3 - accuracy: 0.9904 - val_loss: 0.0740 - val_accuracy: 0.9749
Epoch 4/15
168/168 [==============================] - 38s 226ms/step - loss: 0.013
2 - accuracy: 0.9967 - val_loss: 0.0766 - val_accuracy: 0.9785
Epoch 5/15
168/168 [==============================] - 30s 182ms/step - loss: 0.006
4 - accuracy: 0.9988 - val_loss: 0.1017 - val_accuracy: 0.9737
Epoch 6/15
168/168 [==============================] - 30s 177ms/step - loss: 0.010
4 - accuracy: 0.9961 - val_loss: 0.1308 - val_accuracy: 0.9749
Epoch 7/15
168/168 [==============================] - 31s 186ms/step - loss: 0.003
2 - accuracy: 0.9994 - val_loss: 0.1227 - val_accuracy: 0.9773
Epoch 8/15
168/168 [==============================] - 30s 177ms/step - loss: 0.003
1 - accuracy: 0.9994 - val_loss: 0.1322 - val_accuracy: 0.9761
```

```
Epoch 9/15
168/168 [==============================] - 30s 181ms/step - loss: 0.002
3 - accuracy: 0.9994 - val_loss: 0.1311 - val_accuracy: 0.9809
Epoch 10/15
168/168 [==============================] - 30s 176ms/step - loss: 0.004
9 - accuracy: 0.9988 - val_loss: 0.1548 - val_accuracy: 0.9761
Epoch 11/15
168/168 [==============================] - 31s 185ms/step - loss: 0.002
0 - accuracy: 0.9997 - val_loss: 0.1519 - val_accuracy: 0.9749
Epoch 12/15
168/168 [==============================] - 30s 178ms/step - loss: 0.321
3 - accuracy: 0.9809 - val_loss: 0.0775 - val_accuracy: 0.9773
Epoch 13/15
168/168 [==============================] - 30s 178ms/step - loss: 0.010
9 - accuracy: 0.9979 - val_loss: 0.0880 - val_accuracy: 0.9773
Epoch 14/15
168/168 [==============================] - 30s 178ms/step - loss: 0.004
6 - accuracy: 0.9985 - val_loss: 0.1085 - val_accuracy: 0.9809
Epoch 15/15
168/168 [==============================] - 30s 178ms/step - loss: 0.002
1 - accuracy: 0.9994 - val_loss: 0.1110 - val_accuracy: 0.9821

metrics = pd.DataFrame(history.history)
metrics.rename(columns = {'loss': 'Training_Loss', 'accuracy': 'Training_Accuracy',
'val_loss': 'Validation_Loss', 'val_accuracy': 'Validation_Accuracy'}, inplace =
True)
def plot_graphs1(var1, var2, string):
  metrics[[var1, var2]].plot()
  plt.title('Training and Validation ' + string)
  plt.xlabel ('Number of epochs')
  plt.ylabel(string)
  plt.legend([var1, var2])

plot_graphs1('Training_Accuracy', 'Validation_Accuracy', 'accuracy')
```



Save the model:

model.save('Spam_sms_classifier.h5')

Test the model:

test_sequences = tok.texts_to_sequences(X_test)
test_sequences_matrix = pad_sequences(test_sequences,maxlen=max_len)

accuracy1 = model.evaluate(test_sequences_matrix,Y_test)

44/44 [==============================] - 3s 75ms/step - loss: 0.0431 - a
ccuracy: 0.9914

print('Accuracy:{:0.5f}'.format(accuracy1[0],accuracy1[1]))

Accuracy: 0.04312