Assignment -4

Python Programming

Assignment Date	19 September 2022
Student Name	Pradeepa J
Maximum Marks	2 Marks

Problem Statement:

Over recent years, as the popularity of mobile phone devices has increased, Short Message Service (SMS) has grown into a multi-billion-dollar industry. At the same time, reduction in the cost of messaging services has resulted in growth in unsolicited commercial advertisements (spams) being sent to mobile phones. Due to Spam SMS, Mobile service providers suffer from some sort of financial problems as well as it reduces calling time for users. Unfortunately, if the user accesses such Spam SMS they may face the problem of virus or malware. When SMS arrives at mobile it will disturb mobile user privacy and concentration. It may lead to frustration for the user. So Spam SMS is one of the major issues in the wireless communication world and it grows day by day.

Question-1:

Download the dataset: Dataset

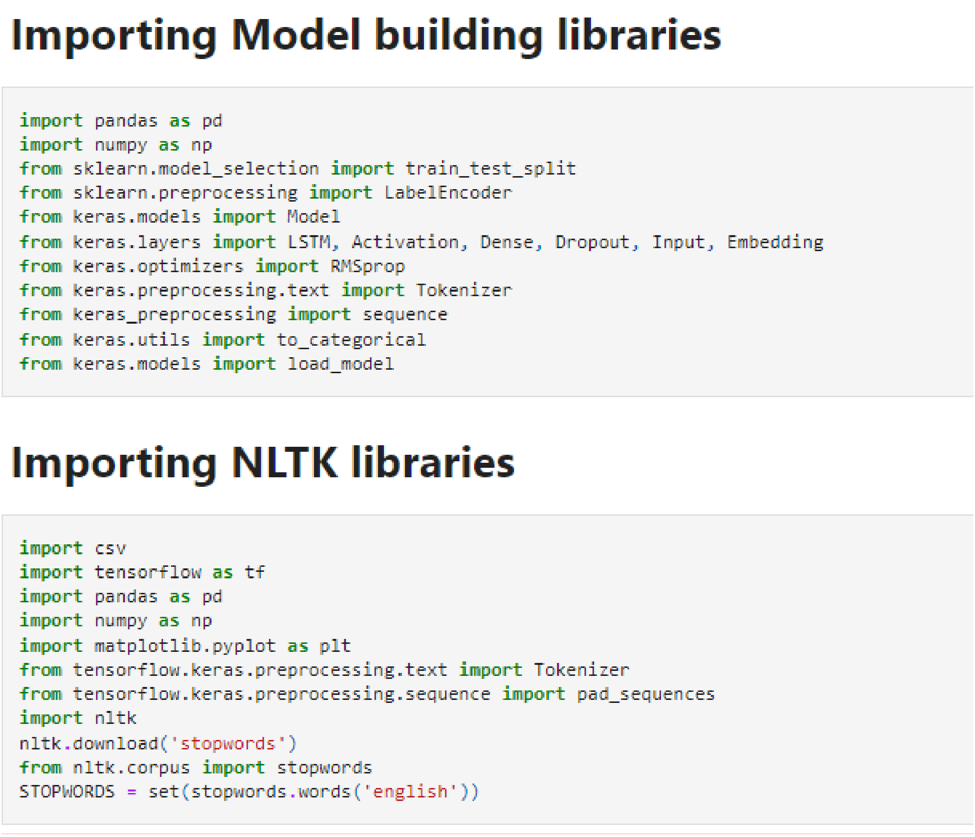
Solution:

https://www.kaggle.com/code/kredy10/simple-lstm-for-text-classification/data

Question-2:

Import required library

Solution:



Importing Model building libraries

```
import pandas as pd
import numpy as np
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 to_categorical
from keras.models import load_model
```

Importing NLTK libraries

```
import csv
import tensorflow as tf
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
import nltk
nltk.download('stopwords')
from nltk.corpus import stopwords
STOPWORDS = set(stopwords.words('english'))
```

Question-3:

Read dataset and do pre-processing

Solution:

```
from google.colab import drive
drive.mount('/content/drive')
```

```
Drive already mounted at /content/drive; to attempt to forcibly remount, call drive.mount("/content/drive", force_remount=True).
```

```
cd/content/drive/MyDrive/Colab Notebooks
```

```
/content/drive/MyDrive/Colab Notebooks
```

```
df = pd.read_csv('/content/drive/MyDrive/AI_IBM/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

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

```
RangeIndex: 5572 entries, 0 to 5571
Data columns (total 2 columns):
 #   Column  Non-Null Count  Dtype 
---  ------  --------------  ----- 
 0   v1      5572 non-null   object
 1   v2      5572 non-null   object
dtypes: object(2)
memory usage: 87.2+ KB
```

```
df.groupby(['v1']).size()
```

```
v1
ham     4825
spam     747
dtype: int64
```

```
#Label Encoding Required Column
X = df.v2
Y = df.v1
le = LabelEncoder()
Y = le.fit_transform(Y)
Y = Y.reshape(-1,1)
```

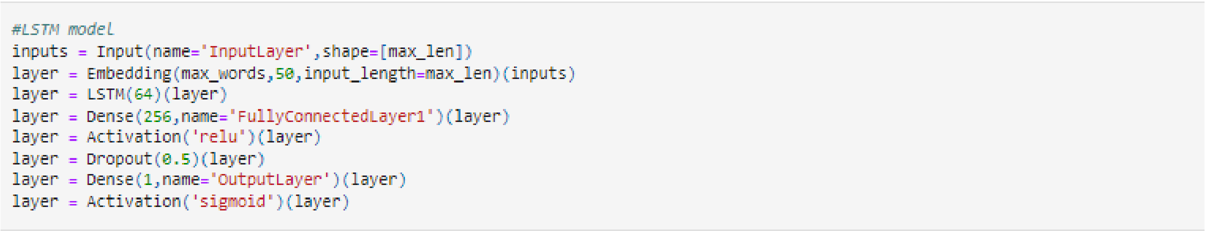
```
# Test and train data split
X_train,X_test,Y_train,Y_test = train_test_split(X,Y,test_size=0.15)
```

```
# Tokenisation function
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 = sequence.pad_sequences(sequences,maxlen=max_len)
```

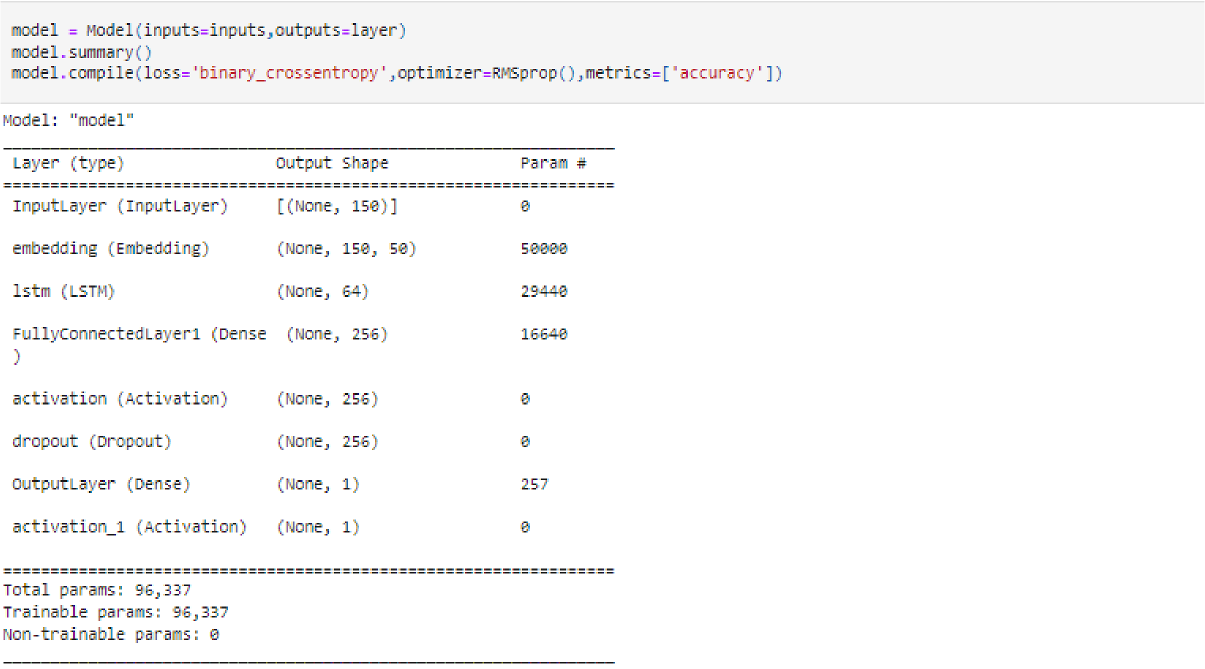
Question-4:

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

Solution:



```
#LSTM model
inputs = Input(name='InputLayer',shape=[max_len])
layer = Embedding(max_words,50,input_length=max_len)(inputs)
layer = LSTM(64)(layer)
layer = Dense(256,name='FullyConnectedLayer1')(layer)
layer = Activation('relu')(layer)
layer = Dropout(0.5)(layer)
layer = Dense(1,name='OutputLayer')(layer)
layer = Activation('sigmoid')(layer)
```



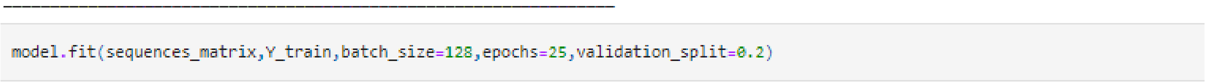
```
model = Model(inputs=inputs,outputs=layer)
model.summary()
model.compile(loss='binary_crossentropy',optimizer=RMSprop(),metrics=['accuracy'])
```

```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #   
=================================================================
 InputLayer (InputLayer)     [(None, 150)]             0         
                                                                 
 embedding (Embedding)       (None, 150, 50)           50000     
                                                                 
 lstm (LSTM)                 (None, 64)                29440     
                                                                 
 FullyConnectedLayer1 (Dense  (None, 256)              16640     
 )                                                               
                                                                 
 activation (Activation)     (None, 256)               0         
                                                                 
 dropout (Dropout)           (None, 256)               0         
                                                                 
 OutputLayer (Dense)         (None, 1)                 257       
                                                                 
 activation_1 (Activation)   (None, 1)                 0         
                                                                 
=================================================================
Total params: 96,337
Trainable params: 96,337
Non-trainable params: 0
_________________________________________________________________
```

Question-5:

Compile The Model

Solution:

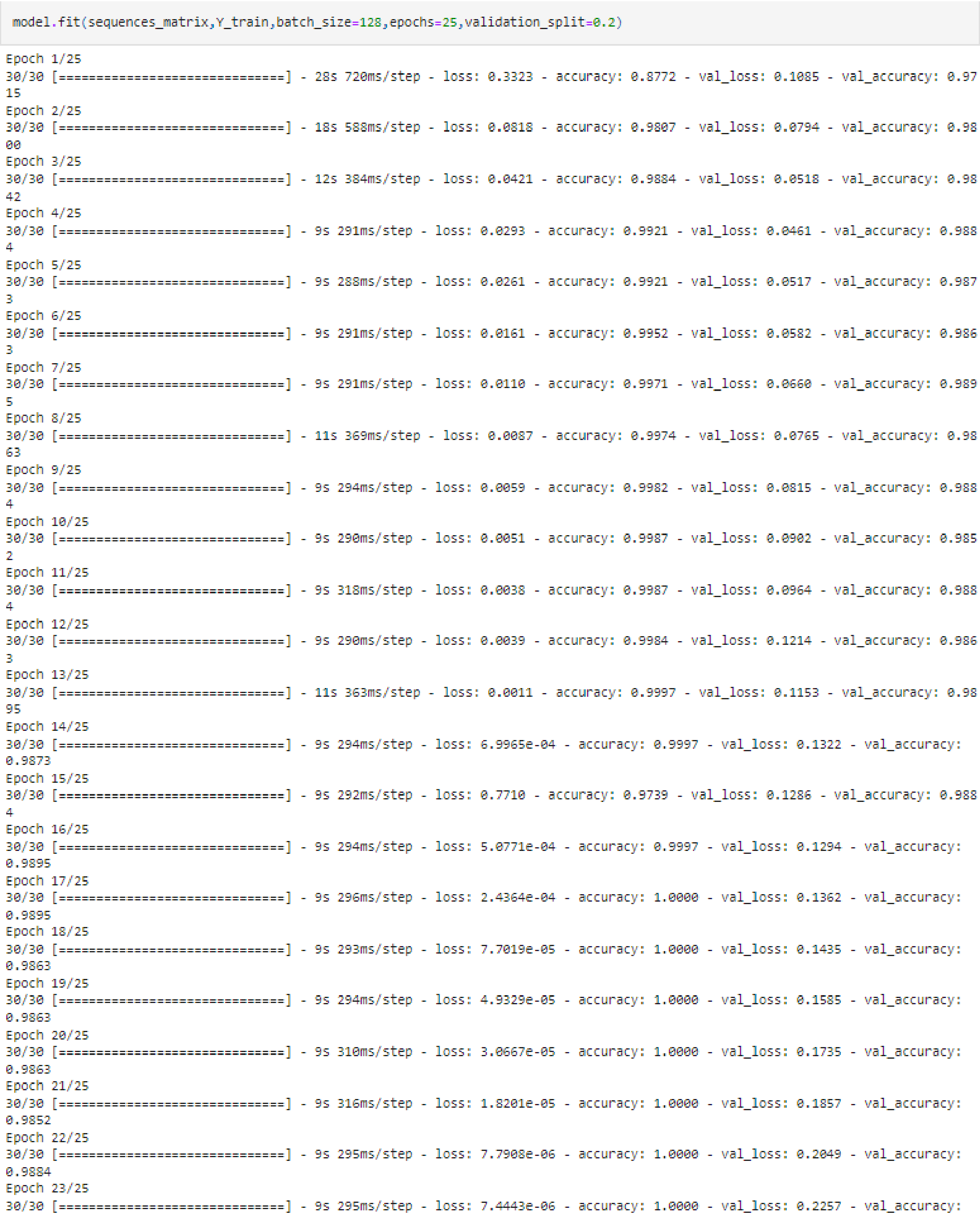


```
_________________________________________________________________
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=25,validation_split=0.2)
```

Question-6:

Fit The Model

Solution:



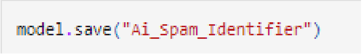
```
model.fit(sequences_matrix,Y_train,batch_size=128,epochs=25,validation_split=0.2)
```

```
Epoch 1/25
30/30 [==============================] - 28s 720ms/step - loss: 0.3323 - accuracy: 0.8772 - val_loss: 0.1085 - val_accuracy: 0.97
15
Epoch 2/25
30/30 [==============================] - 18s 588ms/step - loss: 0.0818 - accuracy: 0.9807 - val_loss: 0.0794 - val_accuracy: 0.98
00
Epoch 3/25
30/30 [==============================] - 12s 384ms/step - loss: 0.0421 - accuracy: 0.9884 - val_loss: 0.0518 - val_accuracy: 0.98
42
Epoch 4/25
30/30 [==============================] - 9s 291ms/step - loss: 0.0293 - accuracy: 0.9921 - val_loss: 0.0461 - val_accuracy: 0.988
4
Epoch 5/25
30/30 [==============================] - 9s 288ms/step - loss: 0.0261 - accuracy: 0.9921 - val_loss: 0.0517 - val_accuracy: 0.987
3
Epoch 6/25
30/30 [==============================] - 9s 291ms/step - loss: 0.0161 - accuracy: 0.9952 - val_loss: 0.0582 - val_accuracy: 0.986
3
Epoch 7/25
30/30 [==============================] - 9s 291ms/step - loss: 0.0110 - accuracy: 0.9971 - val_loss: 0.0660 - val_accuracy: 0.989
5
Epoch 8/25
30/30 [==============================] - 11s 369ms/step - loss: 0.0087 - accuracy: 0.9974 - val_loss: 0.0765 - val_accuracy: 0.98
63
Epoch 9/25
30/30 [==============================] - 9s 294ms/step - loss: 0.0059 - accuracy: 0.9982 - val_loss: 0.0815 - val_accuracy: 0.988
4
Epoch 10/25
30/30 [==============================] - 9s 290ms/step - loss: 0.0051 - accuracy: 0.9987 - val_loss: 0.0902 - val_accuracy: 0.985
2
Epoch 11/25
30/30 [==============================] - 9s 318ms/step - loss: 0.0038 - accuracy: 0.9987 - val_loss: 0.0964 - val_accuracy: 0.988
4
Epoch 12/25
30/30 [==============================] - 9s 290ms/step - loss: 0.0039 - accuracy: 0.9984 - val_loss: 0.1214 - val_accuracy: 0.986
3
Epoch 13/25
30/30 [==============================] - 11s 363ms/step - loss: 0.0011 - accuracy: 0.9997 - val_loss: 0.1153 - val_accuracy: 0.98
95
Epoch 14/25
30/30 [==============================] - 9s 294ms/step - loss: 6.9965e-04 - accuracy: 0.9997 - val_loss: 0.1322 - val_accuracy:
0.9873
Epoch 15/25
30/30 [==============================] - 9s 292ms/step - loss: 0.7710 - accuracy: 0.9739 - val_loss: 0.1286 - val_accuracy: 0.988
4
Epoch 16/25
30/30 [==============================] - 9s 294ms/step - loss: 5.0771e-04 - accuracy: 0.9997 - val_loss: 0.1294 - val_accuracy:
0.9895
Epoch 17/25
30/30 [==============================] - 9s 296ms/step - loss: 2.4364e-04 - accuracy: 1.0000 - val_loss: 0.1362 - val_accuracy:
0.9895
Epoch 18/25
30/30 [==============================] - 9s 293ms/step - loss: 7.7019e-05 - accuracy: 1.0000 - val_loss: 0.1435 - val_accuracy:
0.9863
Epoch 19/25
30/30 [==============================] - 9s 294ms/step - loss: 4.9329e-05 - accuracy: 1.0000 - val_loss: 0.1585 - val_accuracy:
0.9863
Epoch 20/25
30/30 [==============================] - 9s 310ms/step - loss: 3.0667e-05 - accuracy: 1.0000 - val_loss: 0.1735 - val_accuracy:
0.9863
Epoch 21/25
30/30 [==============================] - 9s 316ms/step - loss: 1.8201e-05 - accuracy: 1.0000 - val_loss: 0.1857 - val_accuracy:
0.9852
Epoch 22/25
30/30 [==============================] - 9s 295ms/step - loss: 7.7908e-06 - accuracy: 1.0000 - val_loss: 0.2049 - val_accuracy:
0.9884
Epoch 23/25
30/30 [==============================] - 9s 295ms/step - loss: 7.4443e-06 - accuracy: 1.0000 - val_loss: 0.2257 - val_accuracy:
```

Question-7:

Save The Model

Solution:

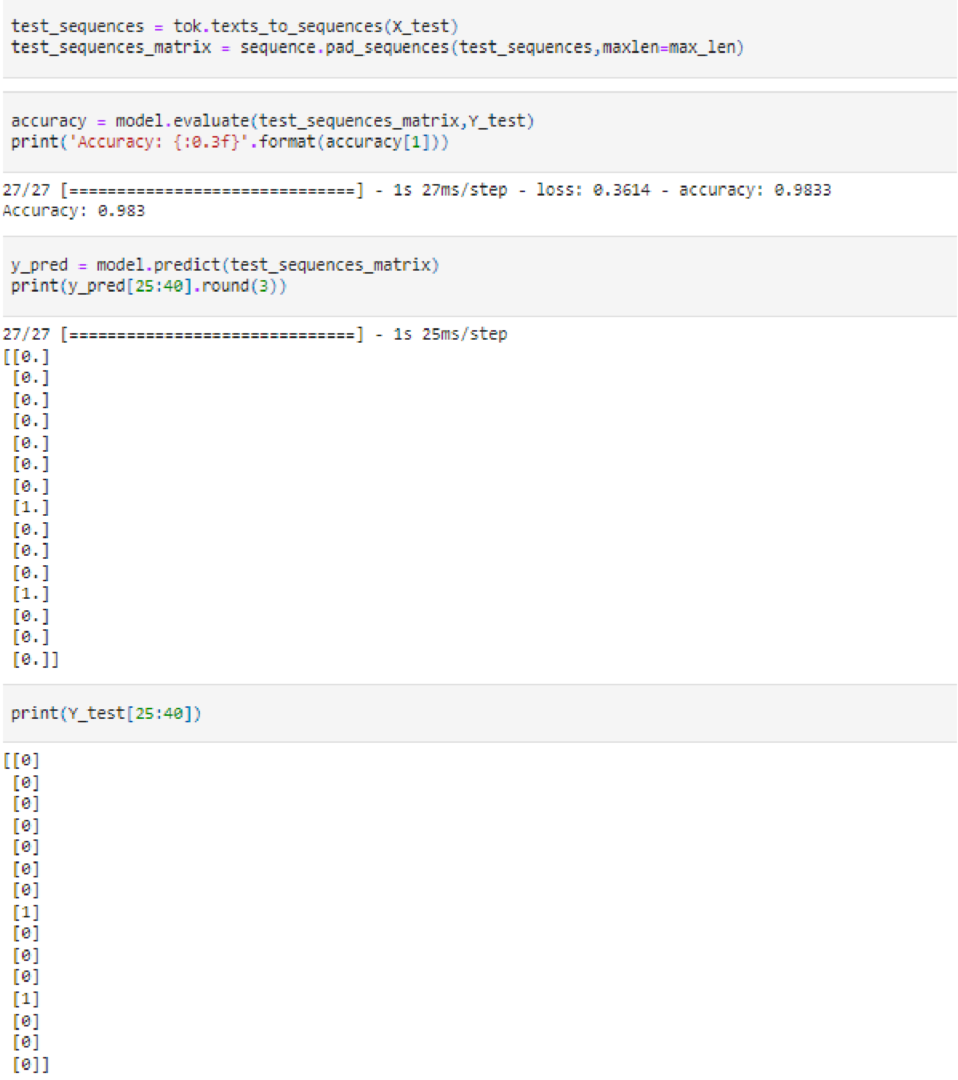


```
model.save("Ai_Spam_Identifier")
```

Question-8:

Test The Model

Solution:



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

```
accuracy = model.evaluate(test_sequences_matrix,Y_test)
print('Accuracy: {:0.3f}'.format(accuracy[1]))
```

```
27/27 [==============================] - 1s 27ms/step - loss: 0.3614 - accuracy: 0.9833
Accuracy: 0.983
```

```
y_pred = model.predict(test_sequences_matrix)
print(y_pred[25:40].round(3))
```

```
27/27 [==============================] - 1s 25ms/step
[[0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [0.]
 [1.]
 [0.]
 [0.]
 [0.]
 [1.]
 [0.]
 [0.]
 [0.]]
```

```
print(Y_test[25:40])
```

```
[[0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]
 [1]
 [0]
 [0]
 [0]]
```