

ROHITH.N(110819104029)

```
import pandas as pd
import numpy as np
from keras import utils
import matplotlib.pyplot as plt
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 to_categorical
%matplotlib inline
```

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

📁 Mounted at /content/drive

```
ls
```

```
drive/ sample_data/
```

READ DATASET

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

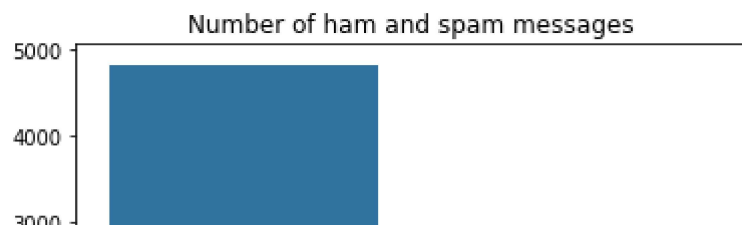
PREPROCESSING

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

```
<class 'pandas.core.frame.DataFrame'>
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
```

```
sns.countplot(df.v1)
plt.xlabel('Label')
plt.title('Number of ham and spam messages')
```

```
/usr/local/lib/python3.7/dist-packages/seaborn/_decorators.py:43: FutureWarning: Pass the following variable as a keyword
FutureWarning
Text(0.5, 1.0, 'Number of ham and spam messages')
```



```
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 = 100
tok = Tokenizer(num_words=max_words)
tok.fit_on_texts(X_train)
sequences = tok.texts_to_sequences(X_train)
sequences_matrix = utils.pad_sequences(sequences,maxlen=max_len)
```

```
sequences_matrix.shape
```

```
(4736, 100)
```

```
sequences_matrix.ndim
```

```
2
```

```
sequences_matrix = np.reshape(sequences_matrix,(4736,100,1))
```

```
sequences_matrix.ndim #3d shape verification to proceed to RNN LSTM
```

```
3
```

```
from keras.models import Sequential
from keras.layers import Dense
from keras.layers import LSTM
from keras.layers import Embedding
```

```
model = Sequential()
model.add(Embedding(max_words,50,input_length=max_len))
```

```
model.add(LSTM(units=64,input_shape = (sequences_matrix.shape[1],1),return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64,return_sequences=True))
model.add(LSTM(units=64))
model.add(Dense(units = 256,activation = 'relu'))
model.add(Dense(units = 1,activation = 'sigmoid'))
```

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

Model: "sequential"

Layer (type)	Output Shape	Param #
=====		
embedding (Embedding)	(None, 100, 50)	50000
lstm (LSTM)	(None, 100, 64)	29440
lstm_1 (LSTM)	(None, 100, 64)	33024
lstm_2 (LSTM)	(None, 100, 64)	33024
lstm_3 (LSTM)	(None, 64)	33024
dense (Dense)	(None, 256)	16640

dense_1 (Dense) (None, 1) 257

```
=====
Total params: 195,409
Trainable params: 195,409
Non-trainable params: 0
=====
```

FIT THE MODEL

```
M = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=7,validation_split=0.2)
```

```
Epoch 1/7
30/30 [=====] - 33s 802ms/step - loss: 0.3238 - accuracy: 0.8949 - val_loss: 0.1223 - val_accu
Epoch 2/7
30/30 [=====] - 24s 788ms/step - loss: 0.0914 - accuracy: 0.9736 - val_loss: 0.0967 - val_accu
Epoch 3/7
30/30 [=====] - 22s 729ms/step - loss: 0.0594 - accuracy: 0.9826 - val_loss: 0.0669 - val_accu
Epoch 4/7
30/30 [=====] - 22s 730ms/step - loss: 0.0454 - accuracy: 0.9873 - val_loss: 0.0609 - val_accu
Epoch 5/7
30/30 [=====] - 24s 814ms/step - loss: 0.0364 - accuracy: 0.9905 - val_loss: 0.0889 - val_accu
Epoch 6/7
30/30 [=====] - 22s 736ms/step - loss: 0.0292 - accuracy: 0.9923 - val_loss: 0.0683 - val_accu
Epoch 7/7
30/30 [=====] - 22s 736ms/step - loss: 0.0233 - accuracy: 0.9934 - val_loss: 0.0613 - val_accu
```

SAVE THE MODEL

```
model.save
```

```
<bound method Model.save of <keras.engine.sequential.Sequential object at 0x7f542e45eb90>>
```

TEST THE MODEL

```

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

accr = model.evaluate(test_sequences_matrix,Y_test)

27/27 [=====] - 4s 65ms/step - loss: 0.0579 - accuracy: 0.9856

l = accr[0]
a =accr[1]
print('Test set\n  Loss: {:.3f}\n  Accuracy: {:.3f}'.format(l,a))

Test set
  Loss: 0.058
  Accuracy: 0.986

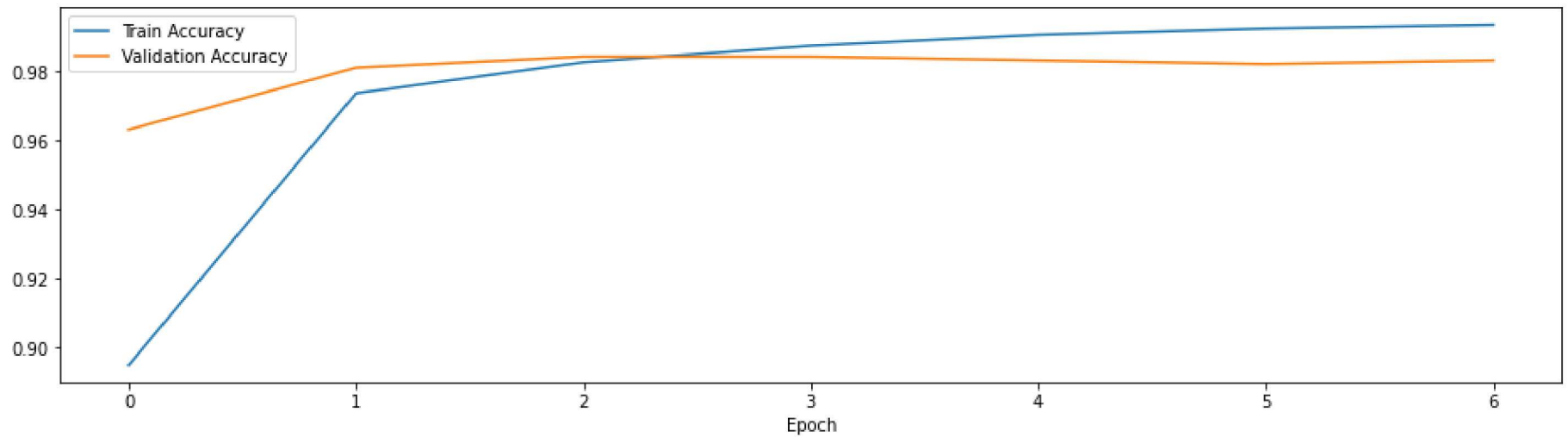
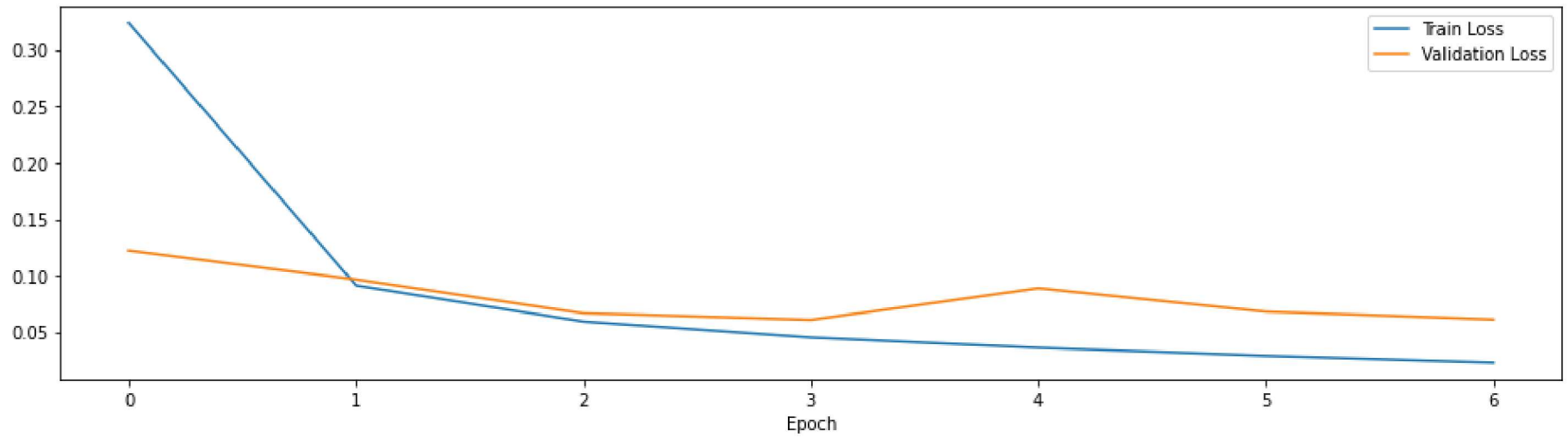
```

ACCURACY AND LOSS GRAPH

```

results = pd.DataFrame({"Train Loss": M.history['loss'], "Validation Loss": M.history['val_loss'], "Train Accuracy": M.history['acc'], "Validation Accuracy": M.history['val_acc']})
fig, ax = plt.subplots(nrows=2, figsize=(16, 9))
results[["Train Loss", "Validation Loss"]].plot(ax=ax[0])
results[["Train Accuracy", "Validation Accuracy"]].plot(ax=ax[1])
ax[0].set_xlabel("Epoch")
ax[1].set_xlabel("Epoch")
plt.show()

```



[Colab paid products](#) - [Cancel contracts here](#)

✓ 0s completed at 3:31 PM

