SIDDHARTH V

```
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/
```

```
df = pd.read_csv('/content/drive/MyDrive/NalaiyaThiranIBM_Siddharth/spam.csv',delimiter=',
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

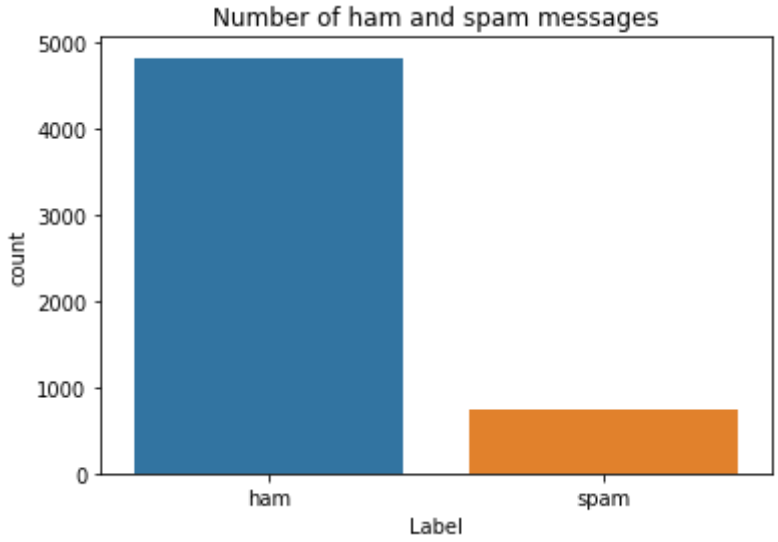
```
df.drop(['Unnamed: 2', 'Unnamed: 3', 'Unnamed: 4'],axis=1,inplace=True)
df.info()
```

```
<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
  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
```

```
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
_________________________________________________________________
```

```
M = model.fit(sequences_matrix,Y_train,batch_size=128,epochs=10,validation_split=0.2)
```

```
Epoch 1/10
30/30 [==============================] - 42s 848ms/step - loss: 0.3194 - accuracy: 0
Epoch 2/10
30/30 [==============================] - 23s 766ms/step - loss: 0.0817 - accuracy: 0
Epoch 3/10
30/30 [==============================] - 23s 766ms/step - loss: 0.0575 - accuracy: 0
```

```
Epoch 4/10
30/30 [==============================] - 25s 841ms/step - loss: 0.0422 - accuracy: 0
Epoch 5/10
30/30 [==============================] - 25s 825ms/step - loss: 0.0334 - accuracy: 0
Epoch 6/10
30/30 [==============================] - 25s 809ms/step - loss: 0.0259 - accuracy: 0
Epoch 7/10
30/30 [==============================] - 29s 982ms/step - loss: 0.0192 - accuracy: 0
Epoch 8/10
30/30 [==============================] - 23s 766ms/step - loss: 0.0151 - accuracy: 0
Epoch 9/10
30/30 [==============================] - 23s 761ms/step - loss: 0.0324 - accuracy: 0
Epoch 10/10
30/30 [==============================] - 23s 776ms/step - loss: 0.0087 - accuracy: 0
```

```
model.save
```

```
<bound method Model.save of <keras.engine.sequential.Sequential object at
0x7fa8e8b846d0>>
```

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

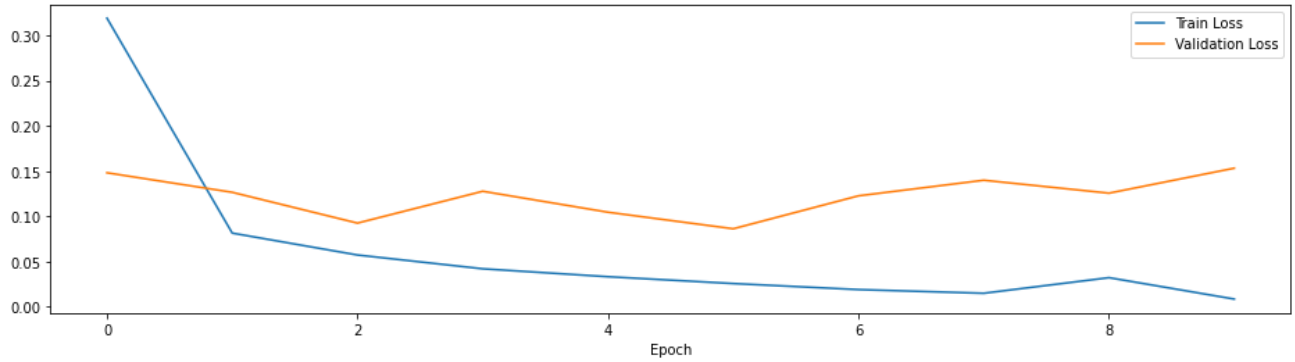
```
accr = model.evaluate(test_sequences_matrix,Y_test)
```

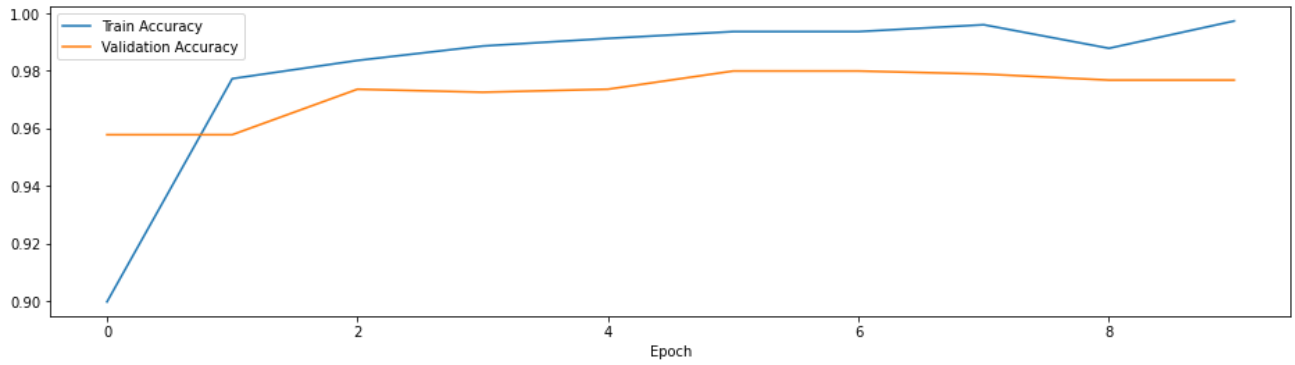
```
27/27 [==============================] - 4s 85ms/step - loss: 0.0832 - accuracy: 0.98
```

```
l = accr[0]
a =accr[1]
print('Test set  Loss: {:0.3f}  Accuracy: {:0.3f}'.format(l,a))
```

```
Test set  Loss: 0.083  Accuracy: 0.987
```

```
DataFrame({"Train Loss": M.history['loss'], "Validation Loss": M.history['val_loss'],"Trai
.subplots(nrows=2, figsize=(16, 9))
in Loss", "Validation Loss"]].plot(ax=ax[0])
in Accuracy", "Validation Accuracy"]].plot(ax=ax[1])
bel("Epoch")
bel("Epoch")
```





Colab paid products - Cancel contracts here

✓ 0s completed at 9:08 PM