Uploading Dataset

```
from google.colab import files

uploaded = files.upload()
```

```
Choose Files No file chosen          Upload widget is only available when the cell has been executed in
browser session. Please rerun this cell to enable.
Saving spam.csv to spam.csv
```

Importing requried 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
```

Reading Dataset & Pre-Processing

```
df = pd.read_csv('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) #dropping unwanted co
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
```

```
# Count of Spam and Ham values
df.groupby(['v1']).size()
```

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

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

```
# Test and train 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)
```

Create Model & Add Layers (LSTM, Dense-(Hidden Layers), Output)

```
# Creating 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)
```

Compiling the model

```
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
_________________________________________________________________
```

Fitting the Model

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

```
Epoch 1/10
30/30 [==============================] - 13s 299ms/step - loss: 0.3233 - accuracy: 0.88
Epoch 2/10
30/30 [==============================] - 8s 277ms/step - loss: 0.0900 - accuracy: 0.977
Epoch 3/10
30/30 [==============================] - 10s 344ms/step - loss: 0.0471 - accuracy: 0.98
Epoch 4/10
30/30 [==============================] - 9s 297ms/step - loss: 0.0340 - accuracy: 0
```

```
Epoch 5/10
30/30 [==============================] - 10s 321ms/step - loss: 0.0271 - accuracy: 0.992
Epoch 6/10
30/30 [==============================] - 12s 414ms/step - loss: 0.0206 - accuracy: 0.994
Epoch 7/10
30/30 [==============================] - 9s 280ms/step - loss: 0.0155 - accuracy: 0.9958
Epoch 8/10
30/30 [==============================] - 8s 276ms/step - loss: 0.0127 - accuracy: 0.9966
Epoch 9/10
30/30 [==============================] - 11s 373ms/step - loss: 0.0115 - accuracy: 0.995
Epoch 10/10
30/30 [==============================] - 8s 277ms/step - loss: 0.0060 - accuracy: 0.9984
<keras.callbacks.History at 0x7ff7a6d18650>
```

Saving the Model

```
model.save("model_1")
```

```
WARNING:absl:Function `_wrapped_model` contains input name(s) InputLayer with unsupporte
WARNING:absl:Found untraced functions such as lstm_cell_layer_call_fn, lstm_cell_layer_c
```

Testing the Model

```
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 24ms/step - loss: 12.3751 - accuracy: 0.1842
Accuracy: 0.184
```

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

```
27/27 [==============================] - 2s 47ms/step
[[1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
 [1.]
```

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

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

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

Colab paid products - Cancel contracts here