Importing Model building libraries

SMS SPAM Classification

```
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'))
```

```
     [nltk_data] Downloading package stopwords to /root/nltk_data...
     [nltk_data]   Unzipping corpora/stopwords.zip.
```

READ DATASET AND DO PREPROCESSING

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

```
     Mounted at /content/drive
```

```
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()
```

```
<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
```

```
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)
```

Create Model

*Add layers (LSTM ,Dense-(HiddenLayers),Ouput) *

```
#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
_________________________________________________________________
```

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

```
Epoch 1/25
30/30 [==============================] - 11s 286ms/step - loss: 0.3359 - accuracy: 0.869
Epoch 2/25
30/30 [==============================] - 8s 265ms/step - loss: 0.0871 - accuracy: 0.9791
Epoch 3/25
30/30 [==============================] - 8s 267ms/step - loss: 0.0422 - accuracy: 0.9881
Epoch 4/25
30/30 [==============================] - 8s 267ms/step - loss: 0.0335 - accuracy: 0.9902
Epoch 5/25
30/30 [==============================] - 9s 307ms/step - loss: 0.0232 - accuracy: 0.9926
Epoch 6/25
30/30 [==============================] - 8s 263ms/step - loss: 0.0206 - accuracy: 0.9931
Epoch 7/25
30/30 [==============================] - 10s 326ms/step - loss: 0.0184 - accuracy: 0.994
Epoch 8/25
30/30 [==============================] - 8s 263ms/step - loss: 0.0105 - accuracy: 0.9968
Epoch 9/25
30/30 [==============================] - 8s 268ms/step - loss: 0.0086 - accuracy: 0.9966
Epoch 10/25
30/30 [==============================] - 8s 267ms/step - loss: 0.0107 - accuracy: 0.9968
Epoch 11/25
30/30 [==============================] - 8s 268ms/step - loss: 0.0057 - accuracy: 0.9987
Epoch 12/25
30/30 [==============================] - 8s 264ms/step - loss: 0.0047 - accuracy: 0.9989
Epoch 13/25
30/30 [==============================] - 8s 265ms/step - loss: 0.0042 - accuracy: 0.9989
Epoch 14/25
30/30 [==============================] - 8s 269ms/step - loss: 0.0039 - accuracy: 0.9984
Epoch 15/25
30/30 [==============================] - 8s 268ms/step - loss: 0.0026 - accuracy: 0.9989
Epoch 16/25
30/30 [==============================] - 8s 268ms/step - loss: 0.0022 - accuracy: 0.9995
Epoch 17/25
30/30 [==============================] - 8s 270ms/step - loss: 0.0026 - accuracy: 0.9997
Epoch 18/25
30/30 [==============================] - 8s 269ms/step - loss: 0.0031 - accuracy: 0.9992
Epoch 19/25
30/30 [==============================] - 8s 269ms/step - loss: 0.0018 - accuracy: 0.9995
Epoch 20/25
30/30 [==============================] - 8s 268ms/step - loss: 0.0031 - accuracy: 0.9989
Epoch 21/25
30/30 [==============================] - 8s 265ms/step - loss: 0.0019 - accuracy: 0.9995
Epoch 22/25
30/30 [==============================] - 8s 265ms/step - loss: 0.0018 - accuracy: 0.9997
Epoch 23/25
30/30 [==============================] - 8s 266ms/step - loss: 0.0020 - accuracy: 0.9995
Epoch 24/25
30/30 [==============================] - 8s 265ms/step - loss: 0.0021 - accuracy: 0.9995
Epoch 25/25
30/30 [==============================] - 8s 269ms/step - loss: 0.0014 - accuracy: 0.9997
<keras.callbacks.History at 0x7f12b5682a10>
```

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

```
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
```

```
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 23ms/step - loss: 0.1436 - accuracy: 0.9892
Accuracy: 0.989
```

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

```
27/27 [==============================] - 1s 24ms/step
[[0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [1.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.877]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]
 [0.   ]]
```

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

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