Data processing using pandas library

```
import pandas as pd
```

Importing Dataset

```
from google.colab import drive
drive.mount('/content/drive/')
data = '/content/spam.tsv'
```

```
Drive already mounted at /content/drive/; to attempt to forcibly remount, call
```

Loading Dataset

```
dataset=pd.read_csv(data,sep='\t',header=None)
```

```
dataset
```

	0	1
0	ham	I've been searching for the right words to tha...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	Nah I don't think he goes to usf, he lives aro...
3	ham	Even my brother is not like to speak with me. ...
4	ham	I HAVE A DATE ON SUNDAY WITH WILL!!!
...	...	...
5562	spam	This is the 2nd time we have tried 2 contact u...
5563	ham	Will ü b going to esplanade fr home?
5564	ham	Pity, * was in mood for that. So...any other s...
5565	ham	The guy did some bitching but I acted like i'd...
5566	ham	Rofl. Its true to its name

5567 rows × 2 columns

Exploratory Data Analysis (EDA)

The better your domain knowledge on the data, the better your ability to engineer more features from it. Feature engineering is a very large part of spam detection in general.

```
dataset.columns=["label","message"]
```

```
dataset.head()
```

	label	message
0	ham	I've been searching for the right words to tha...
1	spam	Free entry in 2 a wkly comp to win FA Cup fina...
2	ham	Nah I don't think he goes to usf, he lives aro...
3	ham	Even my brother is not like to speak with me. ...
4	ham	I HAVE A DATE ON SUNDAY WITH WILL!!!

```
## Data Cleaning and text preprocessing
```

```
import re
import nltk
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
from nltk.corpus import stopwords
stopwords.words('english')
```

```
['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
 'herself',
 'it',
```

```
      "it's",
      'its',
      'itself',
      'they',
      'them',
      'their',
      'theirs',
      'themselves',
      'what',
      'which',
      'who',
      'whom',
      'this',
      'that',
      "that'll",
      'these',
      'those',
      'am',
      'is',
      'are',
      'was',
      'were',
      'be',
      'been',
      'being',
      'have',
      'has',
      'had',
      'having',
      'do',
      'does'.

stopword_list=['i',
 'me',
 'my',
 'myself',
 'we',
 'our',
 'ours',
 'ourselves',
 'you',
 "you're",
 "you've",
 "you'll",
 "you'd",
 'your',
 'yours',
 'yourself',
 'yourselves',
 'he',
 'him',
 'his',
 'himself',
 'she',
 "she's",
 'her',
 'hers',
```

```
 'herself',
 'it',
 "it's",
 'its',
 'itself',
 'they',
 'them',
 'their',
 'theirs',
 'themselves',
 'what',
 'which',
 'who',
 'whom',
 'this',
 'that',
 "that'll",
 'these',
 'those',
 'am',
 'is',
 'are',
 'was',
 'were',
 'be',
 'been',
 'being',
 'have',
 'has',
 'had',
 'having',
 'do',
 'does',
 'did',
 'doing',
 'a',
 'an',
 'the',
 'and',
 'but',
 'if',
 'or',
 'because',
 'as',
 'until',
 'while',
 'of',
 'at',
 'by',
 'for',
 'with',
 'about',
 'against',
 'between',
 'into',
```

```
 'through',
 'during',
 'before',
 'after',
 'above',
 'below',
 'to',
 'from',
 'up',
 'down',
 'in',
 'out',
 'on',
 'off',
 'over',
 'under',
 'again',
 'further',
 'then',
 'once',
 'here',
 'there',
 'when',
 'where',
 'why',
 'how',
 'all',
 'any',
 'both',
 'each',
 'few',
 'more',
 'most',
 'other',
 'some',
 'such',
 'nor',
 'not',
 'only',
 'own',
 'same',
 'so',
 'than',
 'too',
 'very',
 's',
 't',
 'can',
 'will',
 'just',
 'should',
 "should've",
 'now',
 'd',
 'll',
```

```
 'm',
 'o',
 're',
 've',
 'y',
 'isn',
 "isn't",
 'ma',
 'mightn',
 "mightn't",
 'mustn',
 "mustn't",
 'needn',
 "needn't",
 'shan',
 "shan't",
 'shouldn',
 "shouldn't",
 'wasn',
 "wasn't",
 'weren',
 "weren't",
 'won',
 "won't",
 'wouldn',
 "wouldn't"]
```

```
from nltk.stem.porter import PorterStemmer
ps=PorterStemmer()
```

Tokenization-(process of converting the normal text strings in to a list of tokens(also known as lemmas)).

```
corpus=[]
for i in range(0,len(dataset)):
    review=re.sub('[^a-zA-Z0-9]',' ',dataset['message'][i])
    review=review.lower()
    review=review.split() #tokenise
    review=[ps.stem(word) for word in review if not word in stopword_list]
    review=' '.join(review)
    corpus.append(review)
```

```
corpus[:3]
```

```
['search right word thank breather promis wont take help grant fulfil promis
wonder bless time',
 'free entri 2 wkli comp win fa cup final tkt 21st may 2005 text fa 87121
receiv entri question std txt rate c appli 08452810075over18',
 'nah don think goe usf live around though']
```

```
## Train Test split
```

```
## independent features and dependent
y=pd.get_dummies(dataset['label'],drop_first=True)
```

```
y
```

	spam
0	0
1	1
2	0
3	0
4	0
...	...
5562	1
5563	0
5564	0
5565	0
5566	0

5567 rows × 1 columns

```
# Train Test Split

from sklearn.model_selection import train_test_split
X_train, X_test, y_train, y_test = train_test_split(corpus, y, test_size = 0.20, r
```

```
X_train[:3]
```

```
['lover need',
 'go 4 lunch wif famili aft dat go str 2 orchard lor',
 'pl dont forget studi']
```

```
dataset['label'].value_counts()
```

```
ham     4821
spam     746
Name: label, dtype: int64
```

Now we need to convert each of those messages into a vector the SciKit Learn's algorithm models can work with and machine learning model which we will gonig to use can understand

creating the Bag of Words (BOW)

```
from sklearn.feature_extraction.text import CountVectorizer
cv=CountVectorizer(max_features=2500,ngram_range=(1,2))
X_train=cv.fit_transform(X_train).toarray()
```

```
X_train
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
X_test=cv.transform(X_test).toarray()
```

```
X_test
```

```
array([[0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       ...,
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0],
       [0, 0, 0, ..., 0, 0, 0]])
```

```
X_train.shape
```

```
(4453, 2500)
```

```
y_train.shape #target column
```

```
(4453, 1)
```

```
cv.vocabulary_ #a dictionary with the mapping of the word index
```

```
{'lover': 1290,
 'need': 1468,
 'go': 899,
 'lunch': 1303,
 'wif': 2406,
 'famili': 763,
 'aft': 189,
 'dat': 588,
 'orchard': 1581,
 'lor': 1275,
 'pl': 1648,
 'dont': 662,
 'forget': 816,
 'studi': 2047,
 'pl dont': 1650,
```

```
       'centr': 436,
       'someth': 1973,
       'like': 1238,
       'road': 1814,
       'someth like': 1974,
       'free': 825,
       '1st': 58,
       'week': 2373,
       'no1': 1508,
       'nokia': 1512,
       'tone': 2178,
       'ur': 2256,
       'mob': 1393,
       'everi': 732,
       'txt': 2220,
       '8007': 139,
       'get': 877,
       'txting': 2230,
       'tell': 2104,
       'mate': 1334,
       'www': 2454,
       'getz': 888,
       'co': 478,
       'uk': 2236,
       'pobox': 1669,
       '36504': 94,
       'w45wq': 2328,
       'norm150p': 1524,
       '16': 56,
       'free 1st': 827,
       '1st week': 60,
       'week no1': 2377,
       'no1 nokia': 1509,
       'nokia tone': 1518,
       'tone ur': 2183,
       'ur mob': 2271,
       'mob everi': 1394,
       'everi week': 735,
       'week txt': 2379,
       'txt nokia': 2223,
       'nokia 8007': 1515,
       'get txting': 884,
       'txting tell': 2231
```

With messages represented as vectors, we can finally train our spam/ham classifier. Now we can actually use almost any sort of classification algorithms. For a variety of reasons, the Naive Bayes classifier algorithm is a good choice.

Random Forest Classifier

```
from sklearn.ensemble import RandomForestClassifier
classifier=RandomForestClassifier().fit(X_train,y_train)
```

```
<ipython-input-118-9cf6a58fb282>:2: DataConversionWarning: A column-vector y w
  classifier=RandomForestClassifier().fit(X_train,y_train)
```

```
y_pred=classifier.predict(X_test)
```

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

```
accuracy_score(y_test,y_pred)
```

```
0.9865350089766607
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[957,   1],
       [ 14, 142]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      1.00      0.99       958
           1       0.99      0.91      0.95       156

    accuracy                           0.99      1114
   macro avg       0.99      0.95      0.97      1114
weighted avg       0.99      0.99      0.99      1114
```

```
## 10 min break
```

Spam Classification Application

```
msg = input("Enter Message: ")
msgInput = cv.transform([msg])
predict = classifier.predict(msgInput)
if(predict[0]==0):
    print("------------------------MESSAGE-SENT-[CHECK-SPAM-FOLDER]---------------
else:
    print("---------------------------MESSAGE-SENT-[CHECK-INBOX]------------------
```

```
Enter Message: Thanks for your subscription to Ringtone UK your mobile will be
------------------------MESSAGE-SENT-[CHECK-SPAM-FOLDER]-----------------------
```

Naives Bayes Classifier

```
from sklearn.naive_bayes import GaussianNB
clf = GaussianNB()
clf.fit(X_train,y_train)
```

```
y_pred=clf.predict(X_test)
```

```
from sklearn.metrics import classification_report,accuracy_score,confusion_matrix
```

```
accuracy_score(y_test,y_pred)
```

```
0.8447037701974865
```

```
confusion_matrix(y_test,y_pred)
```

```
array([[792, 166],
       [  7, 149]])
```

```
print(classification_report(y_test,y_pred))
```

```
              precision    recall  f1-score   support

           0       0.99      0.83      0.90       958
           1       0.47      0.96      0.63       156

    accuracy                           0.84      1114
   macro avg       0.73      0.89      0.77      1114
weighted avg       0.92      0.84      0.86      1114
```

```
msg = input("Enter Message: ")
msgInput = cv.transform([msg])
msgInput = msgInput.toarray()

predict = clf.predict(msgInput)
if(predict[0]==0):
    print("-----------------------MESSAGE-SENT-[CHECK-SPAM-FOLDER]--------------
else:
    print("---------------------------MESSAGE-SENT-[CHECK-INBOX]-----------------
```

```
Enter Message: Thanks for your subscription to Ringtone UK your mobile will be
-----------------------MESSAGE-SENT-[CHECK-SPAM-FOLDER]----------------------
```