**Email Spam Detection Using Python & Machine Learning**

**Email spam**, also called junk email, is unsolicited messages sent in bulk by email (spamming). The name comes from Spam luncheon meat by way of a Monty Python sketch in which Spam is ubiquitous, unavoidable, and repetitive.

Programming

Description: This program detects if an email is spam (1) or not (0)

Import the libraries

#Import libraries  
import numpy as np   
import pandas as pd   
import nltk  
from nltk.corpus import stopwords  
import string

Load the data and print the first 5 rows.

#Load the data  
#from google.colab import files # Use to load data on Google Colab  
#uploaded = files.upload() # Use to load data on Google Colab  
df = pd.read\_csv('emails.csv')  
df.head(5)

The first 5 rows of data

Let’s explore the data and get the number of rows & columns.

#Print the shape (Get the number of rows and cols)  
df.shape

Image for post

Number of rows: 5728, Number of columns: 2

Get the column names in the data set.

#Get the column names  
df.columns

Image for post

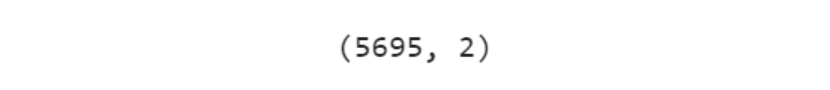
The column names ‘text’ & ‘spam’

Check for duplicates and remove them.

#Checking for duplicates and removing them  
df.drop\_duplicates(inplace = True)

Show the new number of the rows and columns (if any) .

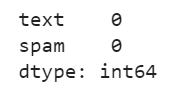
#Show the new shape (number of rows & columns)  
df.shape



Number of rows: 5695, Number of columns: 2

Show the number of missing data for each column.

#Show the number of missing (NAN, NaN, na) data for each column  
df.isnull().sum()



Download the stop words. Stop words in natural language processing, are useless words (data).

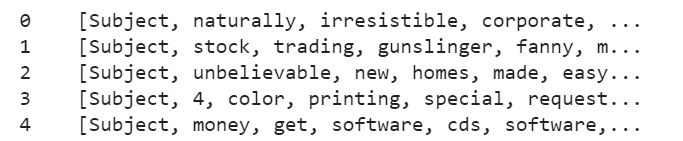
#Need to download stopwords  
nltk.download('stopwords')

We created a function to clean the text and return the tokens. The cleaning of the text can be done by first removing punctuations and then removing the useless words also known as stop words.

#Tokenization (a list of tokens), will be used as the analyzer  
#1.Punctuations are [!"#$%&'()\*+,-./:;<=>?@[\]^\_`{|}~]  
#2.Stop words in natural language processing, are useless words (data).  
def process\_text(text):  
   
 #1 Remove Punctuationa  
 nopunc = [char for char in text if char not in string.punctuation]  
 nopunc = ''.join(nopunc)  
   
 #2 Remove Stop Words  
 clean\_words = [word for word in nopunc.split() if word.lower() not in stopwords.words('english')]  
   
 #3 Return a list of clean words  
 return clean\_words

The process of returning tokens from text is known as Tokenization. We Showed the Tokenization of the first 5 rows of text data from our data set by applying the function process\_text .

#Show the Tokenization (a list of tokens )  
df['text'].head().apply(process\_text)



Convert the text into a matrix of token counts.

from sklearn.feature\_extraction.text import CountVectorizermessages\_bow = CountVectorizer(analyzer=process\_text).fit\_transform(df['text'])

We have to split the data into training & testing sets, and print them. We will use this one row of data for testing to make our prediction later on and test to see if the prediction matches with the actual value.

The testing feature (independent) data set will be stored in X\_test and the testing target (dependent) data set will be stored in y\_test .

The training feature (independent) data set will be stored in X\_train and the training target (dependent) data set will be stored in y\_train .

#Split data into 80% training & 20% testing data setsfrom sklearn.model\_selection import train\_test\_split  
X\_train, X\_test, y\_train, y\_test = train\_test\_split(messages\_bow, df['spam'], test\_size = 0.20, random\_state = 0)

Get the shape of the data.

#Get the shape of messages\_bow  
messages\_bow.shape

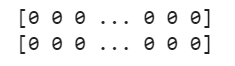
Image for post

Create and train the Multinomial Naive Bayes classifier which is suitable for classification with discrete features (e.g., word counts for text classification)

from sklearn.naive\_bayes import MultinomialNB  
classifier = MultinomialNB()  
classifier.fit(X\_train, y\_train)

Print the classifiers prediction and actual values on the data set.

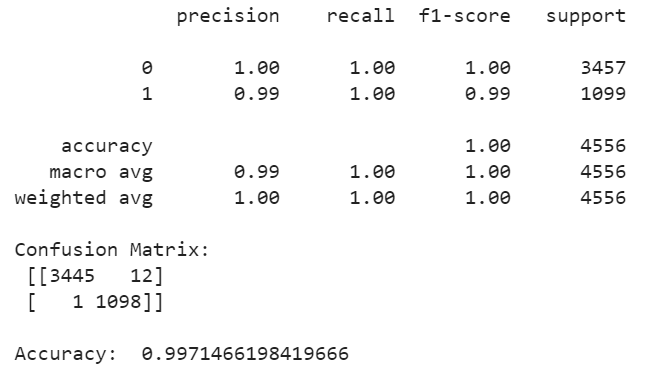
#Print the predictions  
print(classifier.predict(X\_train))#Print the actual values  
print(y\_train.values)



top: predicted values, bottom: actual values

We can see how well the model performed by evaluating the Naive Bayes classifier and showing the report, confusion matrix & accuracy score.

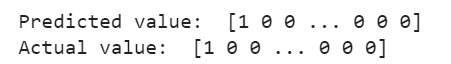
#Evaluate the model on the training data set  
from sklearn.metrics import classification\_report,confusion\_matrix, accuracy\_score  
pred = classifier.predict(X\_train)  
print(classification\_report(y\_train ,pred ))  
print('Confusion Matrix: \n',confusion\_matrix(y\_train,pred))  
print()  
print('Accuracy: ', accuracy\_score(y\_train,pred))



Metrics report followed by the confusion matrix and accuracy score

It looks like the model / classifier used is 99.71% accurate. Let’s test the model / classifier on the test data set (X\_test& y\_test) by printing the predicted value, and the actual value to see if the model can accurately classify the email text/message.

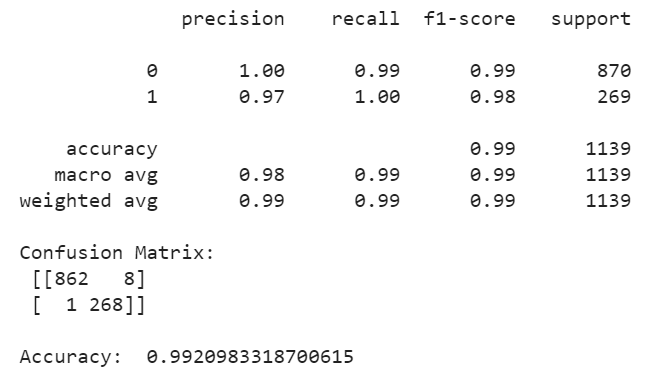
#Print the predictions  
print('Predicted value: ',classifier.predict(X\_test))#Print Actual Label  
print('Actual value: ',y\_test.values)



Sample of the predicted/actual values.

Evaluate the model on the test data set

#Evaluate the model on the test data set  
from sklearn.metrics import classification\_report,confusion\_matrix, accuracy\_score  
pred = classifier.predict(X\_test)  
print(classification\_report(y\_test ,pred ))print('Confusion Matrix: \n', confusion\_matrix(y\_test,pred))  
print()  
print('Accuracy: ', accuracy\_score(y\_test,pred))



The classifier accurately identified the email messages as spam or not spam with 99.2 % accuracy on the test data !

Conclusion and Resources

That is it, we are done creating the email spam detection program !

Resources: [*Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems.*](https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291/ref=sr_1_1?crid=10QDWDNMUMCYF&keywords=hands-on%20machine%20learning%20with%20scikit-learn%20and%20tensorflow&qid=1563929293&s=books&sprefix=hands-on%20machine%2Cstripbooks-intl-ship%2C143&sr=1-1&source=post_page---------------------------)

[Hands-On Machine Learning with Scikit-Learn and TensorFlow: Concepts, Tools, and Techniques to Build Intelligent Systems](https://www.amazon.com/Hands-Machine-Learning-Scikit-Learn-TensorFlow/dp/1491962291/ref=sr_1_1?crid=10QDWDNMUMCYF&keywords=hands-on%20machine%20learning%20with%20scikit-learn%20and%20tensorflow&qid=1563929293&s=books&sprefix=hands-on%20machine%2Cstripbooks-intl-ship%2C143&sr=1-1&source=post_page---------------------------)

Other Resources:

(1) [TFIDF Transformer](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfTransformer.html)  
(2) [Count Vectorizer](https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.CountVectorizer.html)  
(3) [Simple Spam Filter Naive Bayes](https://www.kaggle.com/astandrik/simple-spam-filter-using-naive-bayes)  
(4)[Spam Ham Detection Using Naive Bayes](https://www.kaggle.com/dilip990/spam-ham-detection-using-naive-bayes-classifier)  
(5) [Bag of Words](https://www.geeksforgeeks.org/bag-of-words-bow-model-in-nlp/)  
(6) [Spam Detection With Logistic Regression](https://towardsdatascience.com/spam-detection-with-logistic-regression-23e3709e522)  
(7) [Spam Detection](https://github.com/SharmaNatasha/Machine-Learning-using-Python/blob/master/Classification%20project/Spam_Detection.ipynb)  
(8) [Data Source](https://www.kaggle.com/balakishan77/spam-or-ham-email-classification/data)