Carla Mariana Fera - CA02

LINK TO GOOGLE DRIVE:

https://colab.research.google.com/drive/1QVVbeq33jeXrbTGdvt461VNxnvFyr7DP

Code:

```
#Import all necessary packages in other to run the Naibe Bayes Algorith
m.

import os
import numpy as np
from collections import Counter
from sklearn.naive_bayes import GaussianNB
from sklearn.metrics import accuracy_score

----------------------------------------------------------------------

#The main objective of this function is to make a dictionary that conta
ins the most frequent/common words in the emails.
#This is because according to the most common used words on each type o
f email (not spam/spam),
#Naive Bayes Algorithm will classify each one of them to their correspo
nding category.

def make_Dictionary(root_dir): #def is the function to define another f
unction, in this case the function called "make_Dictionary".
  all_words = [] # this line of code creates an empty list called "all
words".
  emails = [os.path.join(root_dir,f) for f in os.listdir(root_dir)] #th
is line of code passes a directory into a variable, in this case, the d
irectory of the emails.
  for mail in emails: # here we start a for loop to open each email, an
d go throw each line of the email, and split each line into words. Then
 save the words into the list "all_words".
    with open(mail) as m:
      for line in m:
        words = line.split()
        all_words += words # this line of code sabes each word into a l
ist, all_words.
  dictionary = Counter(all_words) # Counter is a class which is used wi
th hashable objects. It creates a dictionarty with the count of each co
rresponding word.
  list_to_remove = list(dictionary) # converts the dictionary created a
bove into a list.
```

```
#In this part of the code, we are trying to remove the numeric values,
which are not relevant to our study, and also any alpha-
numeric value that is 1 character long.
  for item in list_to_remove:
    if item.isalpha() == False: # If alpha is False, meaning if its not
 a word compounded with letters , it will delete this number.
      del dictionary[item]
    elif len(item) == 1: #In this statement, if is not a number, but it
 is of length 1, it will delete it as well.
      del dictionary[item]
  dictionary = dictionary.most_common(3000) # Finally, once we have all
 the words "cleaned", without numbers or undesirable characters, with s
elect the 3000 most commont/used words and store them.
  return dictionary
--------------------------------------------------------------------------

# The objective of this function is to generate a label and word freque
ncy matrix. The labels are the unique words in the emails.

def extract_features(mail_dir):
  files = [os.path.join(mail_dir,fi) for fi in os.listdir(mail_dir)] #
This line of code extracts the emails from the directory and save them
in files variable.
  features_matrix = np.zeros((len(files),3000)) # Here, a matrix is cre
ated filled with zeros.
  train_labels = np.zeros(len(files)) # a numpy array is created filled
 with zeros.
  count = 1;
  docID = 0;
  for fil in files: # This is a for loop to iterate through the files/e
mails.
    with open(fil) as fi: # this line of code open each file/email.
      for i, line in enumerate(fi): # Enumarate method adds a counter t
o an iterable and returns it in a form of enumerate object. In this cas
e, it enumarates the number emails to pass thorugh the for loop.
        if i ==2: #if the number of words is equal to two then:
          words = line.split() # split the words and save them into a v
ariable "word".
          for word in words: #Initiates a for loop, for each word in th
e list of "words".
            wordID = 0 # This line of codes sets the variable wordID to
 zero.
            for i, d in enumerate(dictionary): # This loop iterated thr
ough the words in the dictionary previously created in the other functi
on.
              if d[0] == word: # if the first element of the dictionary
 is equal to the first word in words list, then assing a wordID to that
 word which is i.
```

```
                wordID = i
                features_matrix[docID,wordID] = words.count(word) # cre
ate a matrix with docID, which is equal to 0, And wordID, which will ke
ep changing according to the for loop and the values of i. In this step
 we fill out our matrix.
      train_labels[docID] = 0; # set train_labels, index docID, to zero
.
      filepathTokens = fil.split('/') #slip each file/email when a "/"
appears.
      lastToken = filepathTokens[len(filepathTokens)-
1] #Selects the last 'sentence' in filepathTokens.
      if lastToken.startswith("spmsg"): #a condition in which checks if
 the email is an spam. If it is then, the following codes will run.
        train_labels[docID] = 1;
        count = count + 1 # so it starts making a counter for every spa
m email.
      docID = docID + 1
  return features_matrix, train_labels
```

```
#Finally, in this part of the code where we start applying the Naive Ba
yes Algorithm, which is based in conditional probabilities.

TRAIN_DIR = '/content/drive/My Drive/MSBA_Colab_2020/ML_Algorithms/CA02
/train-mails' # These are the paths where the emails are located.
TEST_DIR = '/content/drive/My Drive/MSBA_Colab_2020/ML_Algorithms/CA02/
test-mails'

dictionary = make_Dictionary(TRAIN_DIR) # Applies the function created
in first place and makes a dictionary of the most common words.

print ("reading and processing emails from TRAIN and TEST folders")
features_matrix, labels = extract_features(TRAIN_DIR) # Applies the sec
ond function to extract the features or variables necessary to run the
algorithm.
test_features_matrix, test_labels = extract_features(TEST_DIR) # use th
e same function for the emails in the test set.

model = GaussianNB() # Selects the type of Naive Bayes to apply.

print ("Training Model using Gaussian Naibe Bayes algorithm .....")
model.fit(features_matrix, labels) # fits the model into the training s
et.
print ("Training completed")
print ("testing trained model to predict Test Data labels")
predicted_labels = model.predict(test_features_matrix) # Predicts or cl
assify the emails on the test set.
```

```
print ("Completed classification of the Test Data .... now printing Acc
uracy Score by comparing the Predicted Labels with the Test Labels:")
print (accuracy_score(test_labels, predicted_labels)) # Prints the accu
racy of the model.

"""====================== END OF PROGRAM ========================="""
```

Results:

```
reading and processing emails from TRAIN and TEST folders
Training Model using Gaussian Naibe Bayes algorithm .....
Training completed
testing trained model to predict Test Data labels
Completed classification of the Test Data .... now printing Accuracy
Score by comparing the Predicted Labels with the Test Labels:
0.9615384615384616
'====================== END OF PROGRAM ========================='
```

The accuracy of the model represents the amount of correct predictions. A 0.96 accuracy means that the model was very accurate in predicting if the email was a spam or not. However, Naïve Bayes assumes the occurrence of one word/ feature is independent of other. But in real life it may not be so (occurrence of you is high after Thank).