VIT®

Vellore Institute of Technology
(Deemed to be University under section 3 of UGC Act, 1956)

Naive Bayes

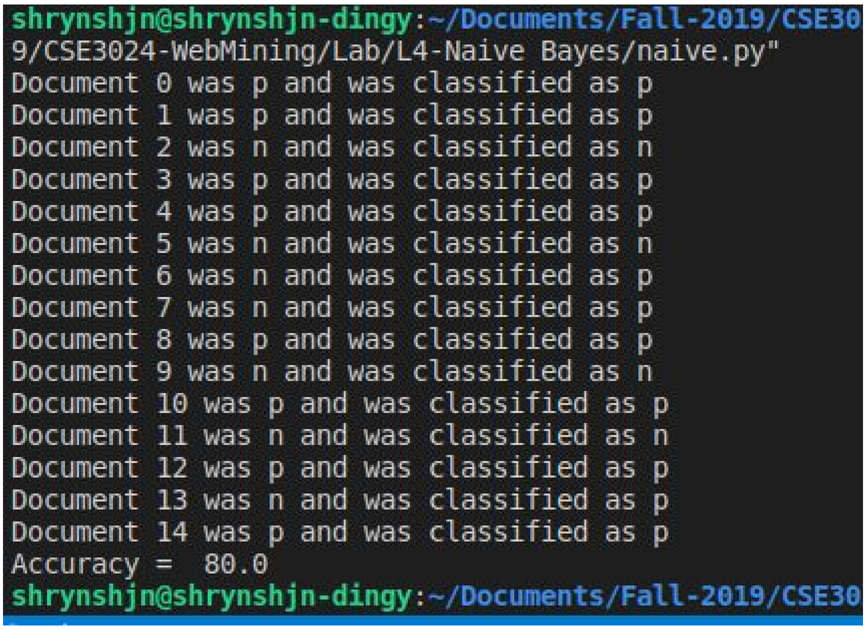
B.Tech in Computer Science and Engineering **(CSE)**, **Fall** Semester **2019**

Name:	Shreyansh Jain
Registration Number:	17BCE0292
Slot:	L49+L50
Faculty Name:	Anuradha J.

Code:

```
import nltk
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
import os
import re
num_format = re.compile("^[\-]?[1-9][0-9]*\.?[0-9]+$")
ps = PorterStemmer()
stop_words = set(stopwords.words('english')) # define a set of
stopwords
punctuations = set([',', '.', '`', ':', '?', ';', '(', ')', '[',
']', '{', '}', '/', '+', '-', '*', '"', "'", '//'])
stop_words = stop_words.union(punctuations) # all the words and
character we want to filter out

def factorial(num):
    fact = 1
    for i in range(1, num+1):
        fact *= i
    return fact

doc_file = [] # list that will store all the documents, each item
is a document
words_docs = [] # list that will convert a document in doc in
doc_file to a list of words it has every item is a list of words
docs = [] # list will store the words in a document after
stopword removal, each item is a list of words in doc(i) after
stopword removal
i = 0
count_p = 0 # number of positive documents
count_n = 0 # number of negative documents
for doc_name in os.listdir("data"):
    if (doc_name.endswith("p.txt")):
        f = open("data/" + doc_name)
        doc_file.append(f.read())
        words_docs.append(word_tokenize(doc_file[i]))
                    docs.append(("p", [ps.stem(word.replace("'",
"").replace("`", "")).lower() for word in words_docs[i] if
((ps.stem(word).replace("'", "").replace("`", "") not in
stop_words) and (not re.match(num_format, word)))]))
        i += 1
    else:
        f = open("data/" + doc_name)
        doc_file.append(f.read())
        words_docs.append(word_tokenize(doc_file[i]))
                    docs.append(("n", [ps.stem(word.replace("'",
"").replace("`", "")).lower() for word in words_docs[i] if
((ps.stem(word).replace("'", "").replace("`", "") not in
stop_words) and (not re.match(num_format, word)))]))
        i += 1

#docs(i) is a 2-tuple with the class as "n" or "p" as the first
element
# docs(i)[1] is the document with stopword removed and each word
at it's index

words_list = set() # a set of all words in all our documents, bag
of words
for i in range(10):
    words_list = words_list.union(set(docs[i][1]))
words_list = list(words_list)
# now words_list has the bag of words for our current documents

# we have 15 documents
# we use one as test document and train the data on the others
# we iterate through the documents so that each document is used
as test data once
correct_results = 0
for test in range(len(docs)):
    term_prob_p = [0]*len(words_list) # probability a term occurs
in +ve document
    term_prob_n = [0]*len(words_list) # probability a term occurs
in -ve document
    test_vector = []
    for train in range(len(docs)):
        for j in range(len(words_list)):
            word = words_list[j]
                if (test == train): # if at the test document,
calculate term frequency vector for it
                test_vector +=  [docs[test][1].count(word)]
            else:
                if (docs[train][0] == "p"):
                    term_prob_p[j] += docs[train][1].count(word)
                else:
                    term_prob_n[j] += docs[train][1].count(word)

    p_sum = sum(term_prob_p)
    n_sum = sum(term_prob_n)
    product_p = 1
    product_n = 1
    for i in range(len(words_list)):
            term_prob_p[i]  =  (term_prob_p[i]  +  1)/(p_sum  +  2)  #
smoothing
            term_prob_n[i]  =  (term_prob_n[i]  +  1)/(n_sum  +  2)  #
smoothing
                                                                  p_term      =
(term_prob_p[i]**test_vector[i])/factorial(test_vector[i])
                                                                  n_term      =
(term_prob_n[i]**test_vector[i])/factorial(test_vector[i])
        product_p *= p_term
        product_n *= n_term
    total_words = sum(test_vector)
    pre_prod = factorial(total_words)
      prob_p  =  product_p  #  probability  that  document  belongs  to
class P (+ve)
    prob_n = product_n
    doc_class = "p" if (prob_p >= prob_n) else "n"
    correct_results += (doc_class == docs[test][0])
          print("Document   {}   was   {}   and   was   classified   as
{}".format(test, docs[test][0], doc_class))
print("Accuracy = " , correct_results/len(docs)*100)
```

Output:



```
shrynshjn@shrynshjn-dingy:~/Documents/Fall-2019/CSE30
9/CSE3024-WebMining/Lab/L4-Naive Bayes/naive.py"
Document 0 was p and was classified as p
Document 1 was p and was classified as p
Document 2 was n and was classified as n
Document 3 was p and was classified as p
Document 4 was p and was classified as p
Document 5 was n and was classified as n
Document 6 was n and was classified as p
Document 7 was n and was classified as p
Document 8 was p and was classified as p
Document 9 was n and was classified as n
Document 10 was p and was classified as p
Document 11 was n and was classified as n
Document 12 was p and was classified as p
Document 13 was n and was classified as p
Document 14 was p and was classified as p
Accuracy =  80.0
shrynshjn@shrynshjn-dingy:~/Documents/Fall-2019/CSE30
```