```
# *************** hw05.py ***************
# Name: Sulav Regmi
# ID: 50211843
# Machine Learning
# Dr. Hung-chi Su

from sklearn import svm
from sklearn.datasets import load_svmlight_file
import os
import re
import collections


def read_files():
    """
    Reading files from the spam and nspam folders, looping through the text
files, getting all the unique words from the
    files and storing them into a dictionary data type while also inserting
the dictionary keys into spam and nspam
    dictionary with the value 1 ordering by keys without repeating any keys
per text file.
    :return: [ordered_spam (list), ordered_nspam(list), dictionary(dict())]
    """
    # dictionary data type to store the words
    dictionary = {}
    # defining a key for the dictionary that holds the words
    i = 1
    # getting all the files/folders in the current directory
    folders = os.listdir('.')
    # initializing empty lists for storing ordered dict() for returning
    ordered_spam, ordered_nspam = [], []

    # looping through the current folder
    for folder in folders:
        # do stuff found in the spam folder
        if folder == 'spam':
            # getting all the files in the spam folder
            spams = os.listdir('spam')
            # looping through all the files in the spam folder
            for spam in spams:
                # initializing a spam dict() that holds the keys for the words
that are stored in the main dictionary
                spam_dict = {}
                # opening the current .txt file
                # encoding required for MACs, didn't have to be specified for
Windows.
                with open(folder+'/'+spam, 'r', encoding='ISO-8859-1') as f:
                    # reading the file
                    content = f.read()
                    # reading one line at a time
                    for line in content.splitlines():
```

```
                        # if line is not empty
                        if line != '':
                            # splitting the line with multiple delimiters,
comma (,) | quote(") | space( ) | period(.)
                            for word in re.split(',|\"| |\.', line):
                                # stripping the whitespace and converting word
into lowercase for uniqueness
                                mword = word.lower().strip()
                                # if word is not empty
                                if mword != '':
                                    # if word is not found in the dictionary,
then add it to the dictionary and also
                                    # add the key of the word to the spam
dictionary with the value and increment i,
                                    # else if word is found in the dictionary,
get the key for that word and store it
                                    # in the spam dictionary with value 1
                                    if mword not in dictionary.values():
                                        dictionary[i] = mword
                                        spam_dict[i] = 1
                                        i += 1
                                    else:
                                        dict_key =
check_dictionary(dictionary, mword)
                                        if dict_key:
                                            spam_dict[dict_key] = 1
                    # for each file, sort the dictionary which also removes
duplicates and append it to the list

ordered_spam.append(collections.OrderedDict(sorted(spam_dict.items())))
        # do stuff found in the nspam folder (not spams)
        if folder == 'nspam':
            # getting all the files in the nspam folder
            nspams = os.listdir('nspam')
            # looping through all the files in the nspam folder
            for nspam in nspams:
                # initializing a nspam dict() that holds the keys for the
words that are stored in the main dictionary
                nspam_dict = {}
                # opening the current .txt file
                # encoding required for MACs, didn't have to be specified for
Windows.
                with open(folder+'/'+nspam, 'r', encoding='ISO-8859-1') as f:
                    # reading the file
                    content = f.read()
                    # reading one line at a time
                    for line in content.splitlines():
                        # if line is not empty
                        if line != '':
                            # splitting the line with multiple delimiters,
comma (,) | quote(") | space( ) | period(.)
```

```
                            for word in re.split(',|\"| |\.', line):
                                # stripping the whitespace and converting word
into lowercase for uniqueness
                                mword = word.lower().strip()
                                # if word is not empty
                                if mword != '':
                                    # if word is not found in the dictionary,
then add it to the dictionary and also
                                    # add the key of the word to the nspam
dictionary with the value and increment i,
                                    # else if word is found in the dictionary,
get the key for that word and store it
                                    # in the nspam dictionary with value 1
                                    if mword not in dictionary.values():
                                        dictionary[i] = mword
                                        nspam_dict[i] = 1
                                        i += 1
                                    else:
                                        dict_key =
check_dictionary(dictionary, mword)
                                        if dict_key:
                                            nspam_dict[dict_key] = 1
                    # for each file, sort the dictionary which also removes
duplicates and append it to the list

ordered_nspam.append(collections.OrderedDict(sorted(nspam_dict.items())))
    return [ordered_spam, ordered_nspam, dictionary]


def read_tests(dictionary):
    """
    Reading files from the test folder, looping through the text files,
getting all the words from the files and
    storing them into a dictionary data type while also inserting the
dictionary keys into test dictionary with the
    value 1 ordering by keys without repeating any keys per text file.
    :param dictionary:
    :return: ordered_test (list)
    """
    # initializing empty list for storing ordered dict() for returning
    ordered_test = []
    # getting all the files in the test folder
    tests = os.listdir('test')
    # looping through all the files in the test folder
    for test in tests:
        # initializing a test dict() that holds the keys for the words that
are stored in the main dictionary
        test_dict = {}
        # opening the current .txt file
        # encoding required for MACs, didn't have to be specified for Windows.
        with open('test/'+test, 'r', encoding='ISO-8859-1') as f:
```

```
            # reading the file
            content = f.read()
            # reading one line at a time
            for line in content.splitlines():
                # if line is not empty
                if line != '':
                    # splitting the line with multiple delimiters, comma (,) |
quote(") | space( ) | period(.)
                    for word in re.split(',|\"| |\.', line):
                        # stripping the whitespace and converting word into
lowercase for checking with word dictionary
                        mword = word.lower().strip()
                        # if word is not empty
                        if mword != '':
                            # if word is found in the dictionary, then get the
key for that word and store it
                            # in the test dictionary with value 1, we don't
store the word in the main word dictionary,
                            # since we're testing the email
                            if mword in dictionary.values():
                                dict_key = check_dictionary(dictionary, mword)
                                if dict_key:
                                    test_dict[dict_key] = 1
            # for each file, sort the dictionary which also removes duplicates
and append it to the list

ordered_test.append(collections.OrderedDict(sorted(test_dict.items())))
    return ordered_test


def check_dictionary(word_dictionary, word):
    """
    Checks word with the dictionary, if found, returns the key for that word,
if not returns False
    :param word_dictionary: dict()
    :param word: string
    :return: integer (key) or boolean (False)
    """
    for k in word_dictionary.keys():
        if word == word_dictionary[k]:
            return k
    return False

# opening text file to write training data
train = open('train.txt', 'w', encoding='ISO-8859-1')

# assigning return values from the read_files() function to the variables
is_spam, not_spam, dictionary_words = read_files()

# assigning return value from the read_tests() function to the variable
test_arr = read_tests(dictionary_words)
```

```
# assigning a variable to check the keys in the test data and the train data
max_key = 0
# opening a test file to write the test data
t = open('test.txt', 'w',  encoding='ISO-8859-1')
# looping through the list passed by the function read_tests() to write to the
testing file
for arr in test_arr:
    # writing 0 (neutral) label for each line
    t.write('0 ')
    # looping through the items in the list
    for key, value in arr.items():
        # writing the key and the value as index:value (e.g. 25:1)
        t.write(str(key) + ':' + str(value) + ' ')
        # if the current key is greater than previously assigned max_key, then
assign max_key to the current key
        if max_key < key:
            max_key = key
    # writing a newline after each item
    t.write('\n')
t.close()

# working with spam list
# looping through the list passed by the function read_files() to write to the
training file
for arr in is_spam:
    # writing -1 (spam) label for each line
    train.write('-1 ')
    # looping through the items in the list
    for key, value in arr.items():
        # we don't write anything greater than the max_key from the test file,
because we have to make sure, the keys
        # are not higher than whatever the test file has
        if key <= max_key:
            train.write(str(key) + ':' + str(value) + ' ')
    # writing a newline after each item
    train.write('\n')

# working with not spam list
# looping through the list passed by the function read_files() to write to the
training file
for arr in not_spam:
    # writing 1 (not spam) label for each line
    train.write('1 ')
    # looping through the items in the list
    for key, value in arr.items():
        # we don't write anything greater than the max_key from the test file,
because we have to make sure, the keys
        # are not higher than whatever the test file has
        if key <= max_key:
            train.write(str(key) + ':' + str(value) + ' ')
```

```
    train.write('\n')

# closing the training file after writing
train.close()

# opening a new file to write dictionary words along with the key (e.g. 1:
Viagra)
dictionary_file = open('dictionary.txt', 'w',  encoding='ISO-8859-1')
for key, value in dictionary_words.items():
    dictionary_file.write(str(key) + ': ' + str(value) + '\n')
dictionary_file.close()

# defining SVM regularization parameter
C = 1.0
# passing the training file to the load_svmlight_file function that returns
x_train and y_train
x_train, y_train = load_svmlight_file('train.txt')
# defining SVC parameters
# kernel's can be 'linear', 'poly', 'rbf', 'sigmoid'
svc = svm.SVC(kernel='linear', C=C)
# fitting the model based on the training data from the train.txt file
svc.fit(x_train, y_train)

# passing the testing file to the load_svmlight_file function that returns
x_test and y_test
x_test, y_test = load_svmlight_file('test.txt')
# predicting the test files based on the model fitted above
predict = svc.predict(x_test)

print(predict)
```