```
# import the libraries
import numpy as np
import pandas as pd
from collections import Counter
import os
import glob
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords


# String used to pre-process the dataset
sp = '!@#$%^&*(){}[]+=|:;,./?><\'\"-1234567890'
stop_words = set(stopwords.words("english"))
ps = PorterStemmer()


def feature_vector(path_func):

 # This function calculates the feature vector, pre-processing like stemming,
 # Also removes stop words and punctuations.

    len_func = 0
    func_vec_allfile = dict()
    for filename in glob.glob(os.path.join(path_func,'*.txt')):
        len_func+=1
        f = open(filename, 'r', encoding = 'utf8').read()
        for i in f:
            if i in sp:
                f = f.replace(i,'')
        f=f.lower()
        f=f.split()
        for words in f:
            if words not in stop_words:
                words_stem = ps.stem(words)
                if words_stem in func_vec_allfile:
                    func_vec_allfile[words_stem]+=1
                else:
                    func_vec_allfile[words_stem]=1
    return(func_vec_allfile,len_func)


# calculating the feature vector for entire document
f_t_allfile= dict()
path_all = 'C:/imd_data/aclImdb/train/allfiles'
all_file = feature_vector(path_all)
f_t_allfile = all_file[0]
len_allfile = all_file[1]


# calculating the feature vector for positive class
pos_vec_allfile = dict()
path_pos = 'C:/imd_data/aclImdb/train/pos'
pos_file = feature_vector(path_pos)
pos_vec_allfile = pos_file[0]
len_pos = pos_file[1]


# calculating the feature vector for negative class
neg_vec_allfile = dict()
path_neg = 'C:/imd_data/aclImdb/train/neg'
neg_file = feature_vector(path_neg)
neg_vec_allfile = neg_file[0]
```

```
len_neg = neg_file[1]


# calculates length of feature vector, total probabilities for both classes
feature_vec_len = len(f_t_allfile)
pos_prob = len_pos/len_allfile
neg_prob = len_neg/len_allfile


# function to calculate conditional probability of each word
def calculate_probability(class_prob):
    prob_both = dict()
    for i in f_t_allfile.keys():
        if i in class_prob:
            prob_both[i] = (class_prob[i]+1)/float(feature_vec_len+ sum
(class_prob.values()))
        else:
            prob_both[i] = (1/float(feature_vec_len+ sum(class_prob.values())))
    return (prob_both)



# calling the function to calculate the probability in each class
prob_posit = dict()
prob_posit = calculate_probability(pos_vec_allfile)
prob_negat = dict()
prob_negat = calculate_probability(neg_vec_allfile)



def test_files(path_test_func):

 # function to test the model. we have used log for the calculation.
 # stemming and removal of punctation and stop words are done here

    pos_value_list = list()
    neg_value_list = list()
    for filename in glob.glob(os.path.join(path_test_func,'*.txt')):
        pos_value = 0
        neg_value=0
        f = open(filename, 'r', encoding = 'utf8').read()
        for i in f:
            if i in sp:
                f = f.replace(i,'')
        f=f.lower()

        for test_word in f.split():
            if test_word not in stop_words:
                test_word_stem = ps.stem(test_word)
                if test_word_stem in prob_posit:
                    a = prob_posit[test_word_stem]
                    b = np.log(a)
                    pos_value+=b
                else:
                    pos_value+=-5.333
                if test_word_stem in prob_negat:
                    c = prob_negat[test_word_stem]
                    d = np.log(c)
                    neg_value+=d
                else:
                    neg_value+=-5.333
        pos_value_list.append(pos_value)
```

```
        neg_value_list.append(neg_value)
    return(pos_value_list,neg_value_list)



# calculating the results for positive test class
path_test_pos = 'C:/imd_data/aclImdb/test/pos'
test_result1 = test_files(path_test_pos)
pos_value_list_1 = test_result1[0]
neg_value_list_1 = test_result1[1]
pos_value_list_1 = [x * pos_prob for x in pos_value_list_1]
neg_value_list_1 = [x * neg_prob for x in neg_value_list_1]



# calculating the results for negative test class
path_test_neg = 'C:/imd_data/aclImdb/test/neg'
test_result2 = test_files(path_test_neg)
pos_value_list_2 = test_result2[0]
neg_value_list_2 = test_result2[1]
pos_value_list_2 = [x * pos_prob for x in pos_value_list_2]
neg_value_list_2 = [x * neg_prob for x in neg_value_list_2]



# comparing the positive and negative list
def comparison(list1, list2):
    result = list()
    for i,j in zip(list1, list2):
        if i > j:
            result.append(True)
        else:
            result.append(False)
    return result



# results from comparison for both classes(positive and negative)
result_pos = comparison(pos_value_list_1,neg_value_list_1)
result_neg = comparison(neg_value_list_2,pos_value_list_2)



# calculating the sum.
result_pos_sum = np.sum(result_pos)
result_neg_sum = np.sum(result_neg)


# Accuracy of the model.
np.sum(result_pos_sum + result_neg_sum)/25000
```