```
import pandas as pd
import numpy as np
import csv


# openfile function
   # input: file path
   # output: data of each row
def of_func(filetpath):
   with open(filetpath, 'r') as csvfile:
      reader = csv.reader(csvfile)
      header = next(reader)
      data = []
      for line in reader:
         data.append(line)
   return data


# get the csv file
validation = "validation_set.csv"
train = "train_set.csv"
data1 = of_func(train)

# delete the first colon
data = pd.read_csv('test_set.csv')
data.drop('textid', axis = 1, inplace = True)
data.to_csv('new_test_set.csv', index = False)

test = "new_test_set.csv"
```

```
data2 = of_func(test)

# get validation set
data3 = of_func(validation)

# dictionary generation function
    # input: data of each row
    # output: datasum( the number of the row ) dict_x (dictionary of each word)
def dict_gen(data1):
    # extract the words and vector
    # record the length of the dataset
    # datasum is the number of the row
    datasum = len(data1)

    # combine the 2-dimension to 1-dimension
    # a is the list of the sentences
    a = [i[0] for i in data1]

    # duplicated summary
    # list_2 is a list to store each word
    # combine the words in one list
    list_2 = []
    for i in range(len(a)):
        list_1 = a[i].split(" ")
        list_2.extend(list_1)

    # create a dictionary to store the vec of each word
    # dictionary
    dict_x = {}
```

```
    for key in list_2:
        dict_x[key] = dict_x.get(key, 0) + 1

    return datasum, dict_x

# data1: train set, data2: test set, data3: valid set
# datasum( the number of the row ) dict_x (dictionary of each word)
datasum_1, dict_x_1 = dict_gen(data1)
datasum_2, dict_x_2 = dict_gen(data2)
datasum_3, dict_x_3 = dict_gen(data3)

# create a empty dict to compare and input value for 1 or 0
# based on the dictionary of train set (order is necessary)
dict_empt = {}
dict_empt.update(dict_x_1)
for i, w in enumerate(dict_empt.keys()):
    dict_empt[w] = i

# build 2-dim list [num_of_row, 2031]
# vector generation function
    # input: datasum (num of row in train dataset)
    #        data (data from each row)
    # output: vec_list(2-dim vector list)
def vec_gen(datasum, data, dict_empty):
    # init a list
    vec_list = []
    for i in range(datasum):
        # the sentence in each row
        sentence_c = data[i][0]
```

```
        word_list = sentence_c.split(" ")
        vec_list.append([])

        # find the same word and put value "1"
        va1 = np.zeros((len(dict_empty.keys()) + 1, ))
        for w in word_list:
            if w in dict_empty.keys():
                va1[dict_empty[w]] = 1
            else:
                va1[-1] = 0

        vec_list[i].append(va1)
    return np.stack(vec_list)

list_1 = vec_gen(datasum_1, data1, dict_empt)
list_2 = vec_gen(datasum_2, data2, dict_empt)
list_3 = vec_gen(datasum_3, data3, dict_empt)

# label dataset
label_1 = [i[1] for i in data1]
label_3 = [i[1] for i in data3]

# np init
label = np.array(label_1)
label3 = np.array(label_3)

def KNN_classify(k,X_train,x_train,Y_test):
    distance_matrix = np.zeros((Y_test.shape[0], X_train.shape[0]))
```

```
    for i, test_vec in enumerate(Y_test):
        for j, train_vec in enumerate(X_train):
            distance_matrix[i][j] = np.sum(np.abs(test_vec - train_vec))

    pred = []
    for distances in distance_matrix:
        row_order = np.argsort(distances, kind='mergesort')
        topK = [x_train[i] for i in row_order[:k]]
        lb, count = np.unique(topK, return_counts=True)
        pred_label = lb[np.argmax(count)]
        pred.append(pred_label)

    return pred

# accuracy based on valid set
accu_list = {}
for i in range(20):
    if i%2 != 0:
        preds = KNN_classify(i, list_1, label_1, list_3)
        x = sum(preds[i] == label_3[i] for i in range(len(label_3))) / len(label_3)
        accu_list.update({i: np.mean(x)})

print("accuracy: ", accu_list)

maxv = np.max(np.array(accu_list))
valmax = max(accu_list.items(),key=lambda x:x[1])

print("highest accuracy: ", valmax)
```

```
#predict test set
vk = valmax[0]

pred1 = KNN_classify(vk, list_1, label_1, list_2)

df = pd.read_csv('test_set.csv')
df['label'] = pd.array(pred1)
df.to_csv('test_set.csv', index = False)
```