2.5	42	+
3.8	51	+
-0.3	-1	+
0.7	3	-
1.6	26	-
2.3	41	-

Scaled Dataset

x1	x2	label
0.68		+
1.00		+
0.00		+
0.24		-
0.46		-
0.63		-

(b) When applying scaling of the attributes in k-NN, it is important to scale any new (test) examples using the same formula that was used to scale the attributes in the training set. Using the scaling formula you computed for the training set above, scale the new example $x = \begin{bmatrix} 3.9 \\ 4 \end{bmatrix}$

and classify it using k-NN with the Euclidean distance measure, for $k = 1$. (The scaled version of the new example can have attribute values that are NOT between 0 and 1.)

What is the predicted label for the example? Show your work.

Part II: Programming Exercise

In this exercise, you will experiment with a sentiment analysis dataset.

This dataset is a version of a dataset downloaded from `http://nifty.stanford.edu/2016/manley-urness-movie-review-sentiment/`, which was previously used in a Stanford project and a Kaggle competition. (See the page referenced in the URL for further details.)

The dataset consists of comments (short movie reviews) from Rotten Tomatoes, where people wrote above movies that they watched. The problem here is to categorize comments as being "positive" (they liked the movie) or "negative" (they didn't like the movie).

An example in this dataset consists of a comment, which is a short piece of text. To simplify processing of these comments, capitalized letters have been converted to lower case, and each comment has been separated into "tokens". We define a token to be a non-empty sequence of non-whitespace characters, separated from the other tokens by whitespace characters or newlines. So, for example

```
a delightful , if minor , movie .
```

has 7 different tokens, and 8 total tokens (the comma token appears twice).

The training data appears in the file `reviewstrain.txt`. Each line of a file contains one comment (example). The first character in the line is the label of the example, which is either 1 (for positive) or 0 (for negative). It is NOT part of the example.

There is also a test file, `reviewstest.txt`. The first character in each line of the test file is the label of the example. It is NOT part of the example.

1. (a) Implement a version of k-Nearest Neighbor to classify the test examples, using the training examples.

   Use the following distance function. For two comments c_1 and c_2, define the distance between c_1 and c_2 to be $\frac{1}{|inter(c_1,c_2)|}$, where $inter(c_1, c_2)$ is the set of distinct tokens appearing in both c_1 and c_2.

For example, if c_1 is `boy oh boy bad .` and c_2 is `oh boy this is great .`, then $inter(c_1, c_2)$ is { boy, oh, . }, which is a set of size 3. Therefore, the distance between c_1 and c_2 is $\frac{1}{3}$.

Run your algorithm with $k = 1$ and with $k = 5$. For $k = 1$, if there is more than one "nearest neighbor" for a test example (because they are all at the same distance from the test example), predict the label that is most common among these nearest neighbors. If there is a tie for the most common label, predict 1. For example, suppose the closest neighbor is at distance 1/5, there are 4 neighbors at distance 1/5, and they have labels $0, 0, 1, 0$. Then predict 0. But if the labels are $0, 0, 1, 1$, there is a tie for the most common label, so predict 1.

For $k = 5$, do something analogous. Sort the training examples by distance from the test example, smallest to largest. If there are other neighbors at the same distance as the 5th one on this list, include them also, and predict the majority class. (If there is a tie for the majority class, predict 1) For example, if the 8 top examples in the sorted list are at distance 1/8, 1/8, 1/8, 1/7, 1/7, 1/7, 1/6, 1/6, then consider all 6 examples at distance 1/8 or 1/7 (since the 5th element in this list is at distance 1/7, we include all examples at distance 1/7). Predict the majority label among these examples, or 1 if there is a tie for the majority label.

Important: When you run your nearest neighbor algorithm, make sure that you do NOT include the labels when you compute the distance between two examples!

Note: You may be tempted to store the training examples in a matrix A, with one row per comment, one column for each possible token, where $A[i, j] = 1$ if token j appears in example i, and $A[i, j] = 0$ otherwise. This will result in a very large matrix, with many entries that are zeroes. It is better to use a sparse matrix (scipy.sparse), or to use other data structures that take less space and support more efficient computation.

Answer the following questions:

i. For $k = 1$, what is the predicted label for the following example in the test set: `It leaves little doubt that Kidman has become one of our best actors .` (This is line 18 of the test file.)
ii. What is the confusion matrix (on the test set) for $k = 1$?
iii. Report the accuracy, the true positive rate, and the false positive rate, on the test set for $k = 1$.
iv. For $k = 5$, what is the predicted label for the following example in the test set: `It leaves little doubt that Kidman has become one of our best actors .` (This is line 18 of the test file.)
v. What is the confusion matrix (on the test set) for $k = 5$?
vi. Report the accuracy, the true positive rate, and the false positive rate, on the test set for $k = 5$.
vii. What is the accuracy on the test set for $k = 5$?
viii. Suppose we used the very simple Zero-R classifier on this dataset, rather than k-NN. That is, we classify all examples in the test set as belonging to the class that is more common in the training set. What is the resulting confusion matrix (on the test set)?

(b) In the dataset we are using here, the examples (comments) are all fairly short, containing relatively few tokens. Suppose that we had a dataset consisting of documents of very different lengths, ranging from e.g., 10 tokens in length, to 10,000 tokens in length. If we applied k-NN to such a dataset, the distance function we are using here might not be a good choice. Why not?

(c) Implement 5-fold cross-validation on the training set to determine which of the following values of k works better in k-NN: 3, 7, 99. (When there are more than k possible nearest neighbors, because of multiple points at the same distance from the test set, handle this analogously to how you handled it in part 1.)

More particularly, to implement the 5-fold cross-validation, divide the training set into 5 sets of equal size. In practice you may want to randomly permute the data before dividing it into 5 sets, but for this assignment, just take the first 1/5 of the examples listed in the file, then the second 1/5, etc. (and DON'T PERMUTE) the examples first.

Then for each of the 3 values of k, do the following. (1) For each of the 5 sets, train on the examples in the other four sets, and test on the examples in the 5th set. The result is that a prediction has been made on each example in the training set. (2) Calculate the percentage of these predictions that were correct. This is the cross-validation accuracy (for this value of k).

i. For each of the 3 values of k, what is the cross-validation accuracy?
ii. Take the k that had the highest cross-validation accuracy. Run k-NN on the entire training set for this value of k, and then test on the test set. Give the confusion matrix and the accuracy (for the test set).

(d) Experiment with using a different distance function. Add an option to your program which allows the user to either choose the distance function above, or to use a second distance function that you define.

There are many different ways to define the distance function. You may want to explore ways of defining the tokens differently (e.g., by removing common "stop-words" or punctation from the examples), ways of giving tokens different weights depending on how common they are (e.g., tf-idf weighting), and/or some other standard ways of defining the distance between two vectors representing two documents (e.g., cosine similarity).

You can choose whatever distance function you like, but should choose something that you think might yield higher accuracy than the first distance function.

Run k-NN with your distance function, using the same training and test sets, to classify the examples in the test set.

i. Describe your distance function. How is the distance between two comments computed? Include an example in your explanation.
ii. Why did you think that your distance function would do better than the first one?
iii. What is the confusion matrix for $k = 1$?
iv. Report the accuracy, the true positive rate, and the false positive rate, on the test set for $k = 1$?
v. What is the confusion matrix for $k = 5$?
vi. Report the accuracy, the true positive rate, and the false positive rate, on the test set for $k = 5$?
vii. Did your distance function achieve higher accuracy (for $k = 1$ and $k = 5$) than the first distance function? If it didn't, what is a possible reason that it didn't?