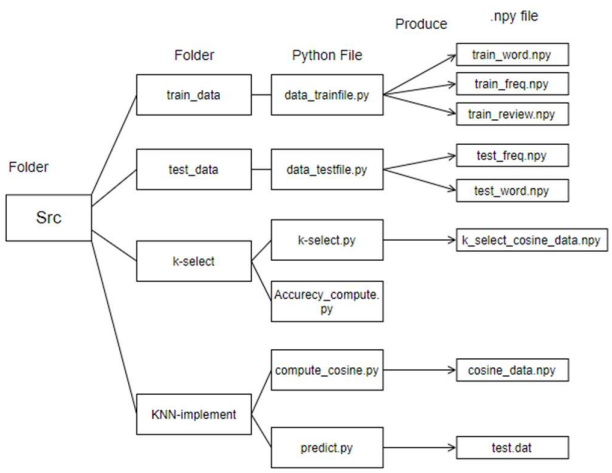
Readme

All the programs are with comments.

1. train_data、test_data folder : handle text data.
   - "data_trainfile.py" handles "train_file.dat" and create three .npy files.
     - "train_word.npy" : useful words of each sentiment.
     - "train_freq.npy" : frequency of words of each sentiment.
     - "train_review.npy" : rating of each sentiment.
   - "data_testfile.py" handles "test_file.dat" and create two .npy files.
     - "test_freq.npy" : useful words of each sentiment.
     - "test_word.npy" : useful words of each sentiment.

2. k-select folder : model selection(choose k).
   - Because computing the cosine values needs lots of time but only needed once, I sperate model selection into two programs.
   - "k-select.py" : use 20% of "train_file.dat" as test data and 80% as train data, and then compute their cosine values. Save top 40 of each sentiment as "k_select_cosine_data.npy".
   - "Accurecy_compute.py" : Read "k_select_cosine_data.npy". Sum the rating of k nearest neighbors of each sentiment. If sum is larger than 0, then predict the rating to be +1.otherwise, -1. Then, compute the accuracy from k = 1 to 40. **Running this program will print out all the result of accuracy from k = 1 to 40 on the screen. <u>This is the only program that will print out something on the screen.</u>**

3. **KNN-implement folder : implement KNN algorithm.**
   - Because computing the cosine values needs lots of time but only needed once, I sperate KNN-algorithm into two programs.
   - "compute_cosine.py" : compute the cosine values. Save top 30 of each sentiment as "cosine_data.npy".
   - "predict.py" : sum the rating of k nearest neighbors of each sentiment in "test_file.dat". If sum is larger than 0, then predict the rating to be +1.otherwise, -1. Save them as "test.dat". **After Running this program, you have to input an integer from 1 to 30 to determine the k value. <u>This is the only program that need input. The rest of programs need nothing to be ran.</u>**



The structure of src folder.