Machine Learning	**S2018**
Problem Set 2	
Handed Out: Feb 7^{th}*,* 2018	*Due: Feb* 19^{th}*,* 2018

In this assignment you will use K-nearest neighbor (KNN) to perform collaborative filtering on a dataset of movie ratings. This dataset consists of 100,000 ratings (on a scale of 1-5) provided by 943 users on 1682 different movies.

You will submit a writeup in Word or PDF that summarizes your results and all code as a zip file. Submit the writeup (with attached source code) to the Canvas submission locker before 11:59pm on the due date.

User-Based Collaborative Filtering (50 points)

Write a method that uses KNN to perform user-based collaborative filtering on this dataset. For this assignment, you've been provided with a dedicated training set (u1-base.base) and test set (u1-test.test). Recall that user-based collaborative filtering determines an item's rating by looking at how similar users rated the item. In this case, our items are movies! For simplicity's sake, we'll hold K constant at K=3 for the timem being.

Details:

- You may not use a high-level function for implementing KNN. Since distance plays a large role in this algorithm, you also may not use a high-level function to calculate distance or similarity. In addition, you may not use a high-level function to calculate error. Any high-level function outside of something that automatically performs KNN, calculates distance/similarity, or calculates error, however, is fair game.
- As mentioned at the start of class, all code should be written in Python. You can choose whether you want to use Python 2 or 3.
- You can choose whatever metric you like to calculate similarity and/or distance. Two possible candidates are Euclidean distance and cosine similarity.
- Explain any implementation choices you had to make in the final report.
- Include the test set error in the final report. Since we're predicting ratings, we'll use average mean squared error to measure test error.

Determine the best K using cross validation (30 points)

This sounds familiar, but not too familiar. One of the most important decisions to make while using the KNN algorithm is what K value to use. Use cross validation to choose the optimal value for K for this problem from the range 1-10.

You may use either leave-one-out or a fold-based method. You **must** include a description of your algorithm in your writeup that would be sufficient for someone to reimplement your method. In addition, provide explanation for any interesting behavior you observe during this process. Why do some values for K perform well or poorly? The average mean squared error for each K value should be provided in your report.

Presentation (20 points)

Your report must be complete and clear. A few key points to remember:

- Complete: the report does not need to be long, but should include everything that was requested.
- Clear: your grammar should be correct, your graphics should be clearly labeled and easy to read.
- Concise: I sometimes print out reports to ease grading, don't make figures larger than they need to be. Graphics and text should be large enough to get the point across, but not much larger.
- Credit (partial): if you are not able to get something working, or unable to generate a particular figure, explain why in your report. If you don't explain, I can't give partial credit.

Bonus: Item-Based Collaborative Filtering (10 Points, required for graduate students)

Alter your KNN approach to collaborative filtering to perform item-based collaborative filtering. When predicting the rating that a user will give to an item, use that user's ratings on similar items to make the prediction. For this part of the assignment you will continue to use the training and test sets provided.

To determine to determine similarity between items (movies), I have provided some additional information that could prove very useful. The file u-item.item contains additional information about each movie rated in the training and test datasets. Of particular note should be the series of 19 binary values at the end of each line in u-item.item. These refer to a movie's genre (the translation file has also been included: u-genre.genre). Use this information to determine the similarity between movies in order to find the K most similar movies that a user has rated. For this assignment, we will use K=3.

The details for this section are the same as those for the first, but I will still list them here as a reference.

Details:

- You may not use a high-level function for implementing KNN. Since distance plays a large role in this algorithm, you also may not use a high-level function to calculate distance or similarity. In addition, you may not use a high-level function to calculate error. Any high-level function outside of something that automatically performs KNN, calculates distance/similarity, or calculates error, however, is fair game.
- As mentioned at the start of class, all code should be written in Python. You can choose whether you want to use Python 2 or 3.
- You can choose whatever metric you like to calculate similarity and/or distance. Two possible candidates are Euclidean distance and cosine similarity.
- Explain any implementation choices you had to make in the final report.
- Include the test set error in the final report. Since we're predicting ratings, we'll use average mean squared error to measure test error.