Capstone Project Introduction to Machine Learning
Regression flavor

Preamble: In 2006, Netflix released over 100 million ratings from over 400,000 users on over 17,000 films. Whoever improved the RMSE of Netflix's "Cinematch" algorithm by 10% within 5 years received $1,000,000. Here, we revisit this notorious contest by using a subset of this dataset, with a different test set than the original contest.

Scope: This is a "for real" project, that includes parts of all aspects of the class. It is designed to stretch your abilities. No one is expecting perfection – the question is what you can come up with in a couple of weeks, while you have other ongoing commitments, which simulates typical industry conditions. If achieving reasonable performance, you will be able to use this capstone project as a "portfolio" item that you can highlight in interviews for internships and jobs. Note that this is not a toy dataset or project. As real data always has (many) real issues, any success in prediction will be hard earned and not easily forthcoming.

Academic integrity: You are expected to do this project by yourself, individually, so that we are able to determine *your* ability to solve machine learning problems. We'll be on the lookout for suspicious similarities, so please refrain from copying off someone else (also note that the idiosyncratic seed keys the correct answers to you, and not anyone else, making it effectively pointless to copy off others).

Data: The file dataSet.zip contains two files: data.txt and movieTitles.csv

The data.txt file contains the ratings from the ~400k users on 5k movies, in the following format: Each row represents either a movie id, or a movie rating. If a row contains only a number followed by a colon, e.g.
758:
it represents that the following rows represent the ratings for the 758th movie in the dataset, until encountering the row that contains only
759:
the following rows represent the ratings for the 759th movie in the dataset, until the next number/column only row. Rows between those contain a first integer, a second integer and a date, from left to right, separated by commas. The first integer represents the id of the user (from 1 to ~400k) that gave the rating, the second integer represents the rating given by that user to the movie with the last encountered movie id (the integer before the colon). This second integer will always be a number between 1 and 5, representing a rating of 1 to 5 stars. There are no half-stars. The date is the date on which the user rated the movie, in YYYY-MM-DD format.

The movieTitles.csv file contains information as to which movie is represented by the movie id in the data.txt file. For instance, the movies "758" and "759" mentioned above are "Mean Girls" and "Fifteen Minutes", respectively. Each row in this file represents a movie, the first column is the movie id (corresponding to the number in front of the colon of the rows in the data.txt file), the 2nd column contains the release date and the 3rd column the movie title. Strictly speaking, this file is provided for your edification only – it probably won't have much impact on your model performance per se. However, this could be a good source of material for the extra credit observations. For instance, how does release date impact rating, how about the interval between release data and rating date, how about the linguistic features of the title? Have titles gotten shorter over time, etc.?

Here is what we want you to do:
*Download the file dataSet.zip from the website and uncompress it to retrieve the two files data.txt and movieTitles.csv
*Write code that creates a regression model (of your choice, can be anything we covered in class – including linear regression models, regularized regression models, trees, forests, neural networks, etc.; it is up to you whether you want to integrate a suitable dimensionality reduction or clustering step – it might improve your model performance, but you don't have to) that predicts the rating of a movie in the test set from the rest of the data.
*In the code, before anything else happens, we want you to initialize the seed of the random number generator with your NYU N-number. For instance, if your N-number (on the back of your NYU ID) is N18994097, we would expect the first lines of your code to be:
import random
random.seed(18994097)
Note: Don't use "18994097", unless that is your N-number. Use yours instead. This is so that the correct answers (which depend on which movie ratings go into the test set) are specific to *you*.
*Make sure to do the following train/test split: Use *all* ratings for a given movie in the training set, except for *one* randomly picked rating. That one rating (per movie) constitutes the test set. So the test set will be 5000x1 randomly picked ratings (one per movie). Use all the other ratings in the training set.
*Use this test set to determine the RMSE of your model. Try to get the RMSE as low as possible without having any leakage between training and test set.
*Write a brief report (1-2 pages) as to how you built your model, how you made your design choices (why you did what you did) and addressing how you handled the challenges below. Make sure to state your final RMSE at the bottom of the report.
*For extra credit, include some interesting non-trivial observation or data visualization in your report.
*Upload both the report (pdf only) and your code (some kind of python or notebook format only) to the Brightspace portal by the due date.

Some key challenges you can expect when taking on this project:
*The ratings data is natively provided in a large text file. With the ratings of each movie just glued one after each other. You need to find a way to parse this file, extract the relevant information and put it into some kind of usable format (likely an array or dataframe).
*There is lots of missing data. In fact, most of the data will be missing. Most users have not rated most of the movies. You will need to find a way to handle that, either by imputation or in some other way. There might even be information in the missing data (as this might reflect a choice by a user NOT to see a movie that they anticipated not to enjoy)
*There is lots of data. This is genuinely a "big" dataset, so you need a model/infrastructure that can handle that. Don't build a model that will take months to train, as you don't have that much time.
*The movie ratings themselves (per movie) are unlikely to be normally distributed.
*There might be temporal or seasonal effects in movie ratings. You do not have to include that information in your model (as to when a movie was rated), but you can expect the model to improve if you do so. However, doing so will be challenging (you would have to model some kind of temporal or seasonal kernel). So including this kind of information is optional (but could be interesting, if you like to be challenged).