Python Decal Final Project Spring 2023

MOVIE RECOMMENDATION SYSTEM

★ Zhi Wang, Roberto Velador, Caprice Neman ★

Table of Contents

Background Information

Netflix Prize: a competition launched by Netflix in 2006 with the aim of improving the accuracy of their movie recommendation algorithm by at least 10%.

- ★ Our goal was to create a similar system using collaborative filtering similar to Netflix to create a recommendation system for movies, based on datasets of previous user ratings in addition to our own.
- ★ The system lists a total of ten films on the screen (one at a time), in which a user rates each film on a scale from one to five.
- ★ Each film listed on the screen is chosen at random and is not based on the previous films nor their ratings.
- ★ After the ten films are rated by the user, the ratings are then used to suggest a total of ten new films based on their selections.

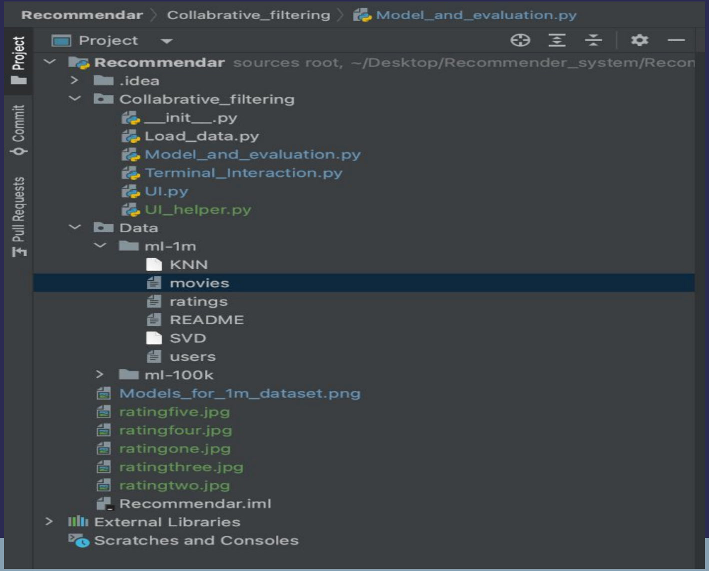
Background Information for UI

- ★ A user interface is important because it’s how the user interacts with the algorithm without directly diving into the code.
- ★ The main goal of a user interface is to be user friendly; easy to understand and easy to use
- ★ Creating a UI requires understanding what inputs and objects users are using: buttons, text fields, lists, icons, messages boxes, progress bar
- ★ For a movie recommendation system the main ingredient is the buttons to rate the movie and the text
- ★ A useful skill for UI design is graphic design, it needs to be pretty without it being overwhelming.

Background Information

Code was split into two components:

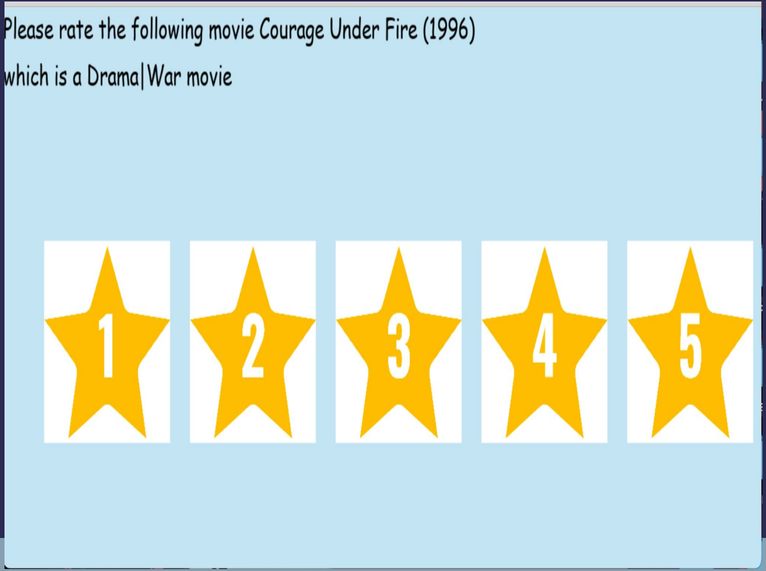
1. Recommendation System



Recommendar 〉 Collabrative_filtering 〉 Model_and_evaluation.py

```
Project
Recommendar sources root, ~/Desktop/Recommender_system/Recon
  > .idea
  v Collabrative_filtering
      __init__.py
      Load_data.py
      Model_and_evaluation.py
      Terminal_Interaction.py
      UI.py
      UI_helper.py
  v Data
    v ml-1m
        KNN
        movies
        ratings
        README
        SVD
        users
    > ml-100k
    Models_for_1m_dataset.png
    ratingfive.jpg
    ratingfour.jpg
    ratingone.jpg
    ratingthree.jpg
    ratingtwo.jpg
    Recommendar.iml
> External Libraries
  Scratches and Consoles
```

2. Pygame Interface



Please rate the following movie Courage Under Fire (1996)

which is a Drama|War movie

Methods & Techniques

Libraries Used:

NumPy	Provides some basic data structures and generate random integers to get random movies.
Pandas	Used to import and utilize datasets. e.g. we use pd.read_csv() to read in movies as panda dataframe and .iloc() to access movies.
Matplotlib	Plot model performance.
Sklearn	Provides methods to split data set into training and test parts.
Surprise	A Python library for building and evaluating recommendation systems. The library provides tools for loading and preprocessing data, as well as model building such as searching for optimal parameters, evaluating accuracy, storing and loading models, etc.
PyGame	Used for creation of graphic UI and enabling mouse-click interaction
time	Time how long it takes to build models, just for fun.

Methods & Techniques

Methodology:

Collaborative Filtering	Collaborative filtering is a technique that can filter out items that a user might like on the basis of reactions by similar users. It works by searching a large group of people and finding a smaller set of users with tastes similar to a particular user. This is how the “evil” TikTok works, by using information of what videos you watched, liked, skipped, and commented etc to find similar users to you and recommend videos they prefer to you, in real time and large scale. It’s all about identifying similarity.
Singular Value Decomposition (SVD)	Will be explained in detail later
K-Nearest Neighbor	Will be explained in detail later
MovieLens 100K Dataset, MovieLens 1M Dataset	Datasets used for collaborative filtering. Relatively small and clean for pandas to preprocess.
Object oriented programming	To glue two parts together, we created a “UI_helper” class and instantiate a UI_helper in UI file to take care of everything a model should do. The Button in the UI is also a class.

K-Nearest Neighbors (KNN)

Parameters:

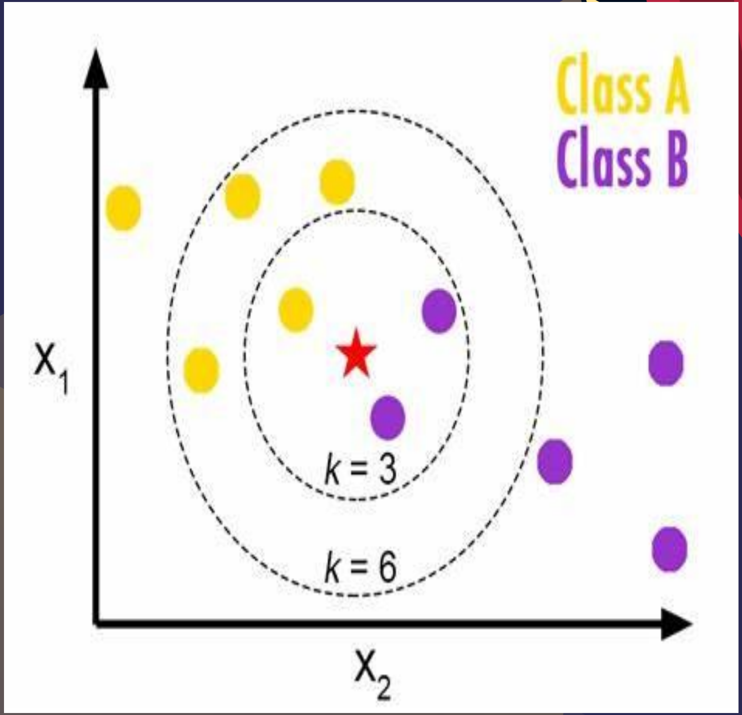
- Similarity metric d: how to determine two are similar
- Number of neighbors k
- User-based or item-based:
  E.g. TikTok wants to find similarity based a user behavior vector, but Amazon may prefer classifying items.

Suppose we use user-based algo:

1. Find k most similar users using metric d.
2. Take the weight average of their ratings as the predicted rating vector.

How to find these parameters?

- GridsearchCV in surprise library, give it an evaluation metric.
- if different from gradient descent if you are familiar with neural network, the difference is, Gridsearch find the best parameters in the set you provide, but neural network doesn't require you to provide a set of hyperparameters. Gridsearch only involve arithmetic but neural networks use calculus and linear algebra.



Singular Value Decomposition (SVD)

Matrix factorization is now one of the most useful techniques in dealing with data in large scale.

- a team of researchers led by Simon Funk, who used a variant of the SVD algorithm called incremental SVD to handle the large-scale dataset, is one of the winners of the Netflix Prize. Actually, the success of the competition popularized the use of matrix factorization in recommendation systems.

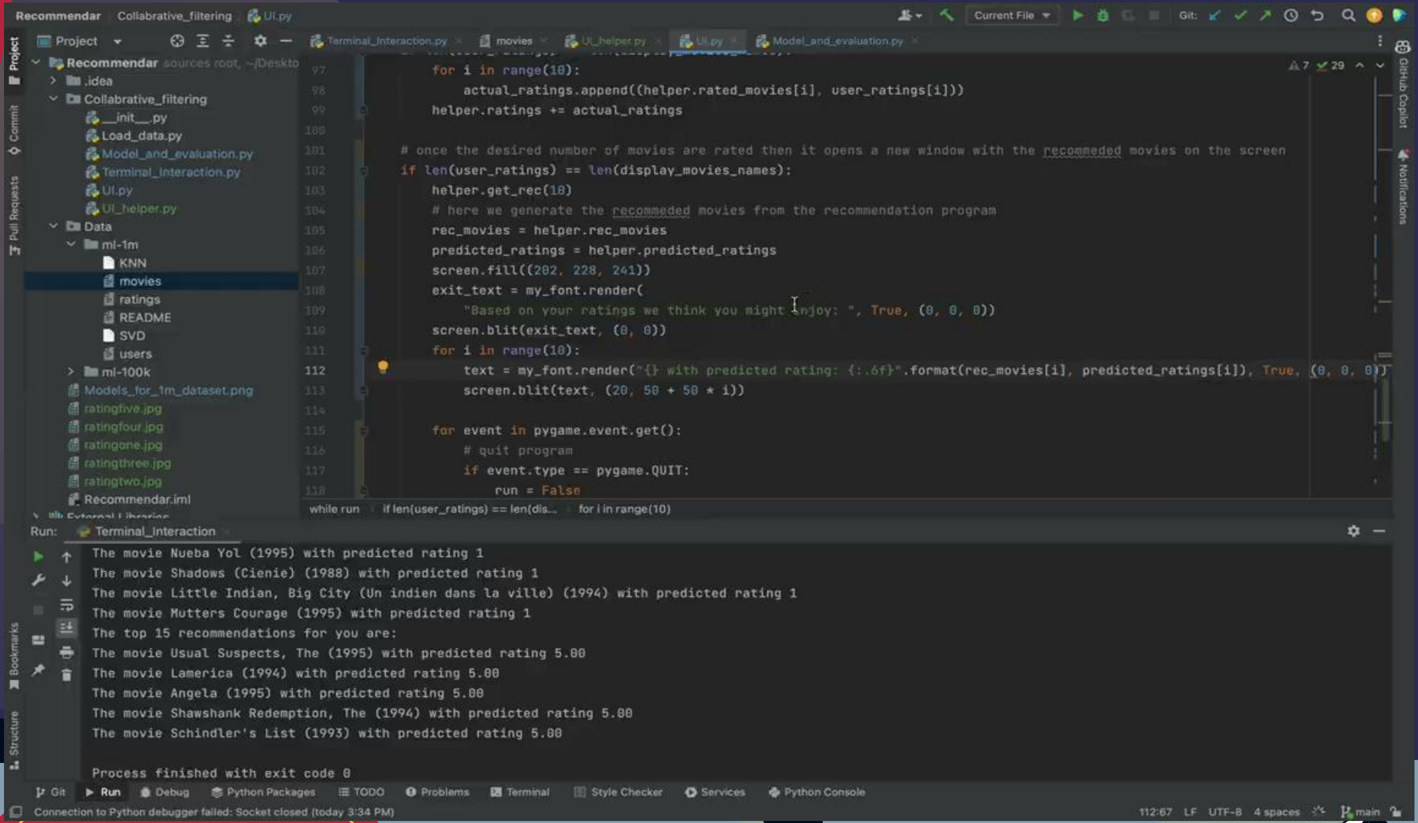
Below are steps of SVD algorithm (extremely simplified version) :

- The user-item rating matrix is decomposed into two low-rank matrices representing users and items, and a diagonal matrix of singular values representing the importance of each latent factor, R = UDV, think of U as users, V as items, and D as their relations.
- To make a prediction for a user-item pair, the dot product of the corresponding rows in the user and item matrices is computed, the global mean is added back, and the final prediction is returned.
- Use stochastic gradient descent to update U and V. (minimize a loss function, a lot of hyperparameters and regularization, omitted here, we can still use gridsearch to tune the hyperparameters)
- To get prediction, simply provide user id, i.e row of U, and item id, i.e column of V, take their dot product to get the prediction.

Findings & Results

03

Findings & Results



```python
        for i in range(10):
            actual_ratings.append((helper.rated_movies[i], user_ratings[i]))
        helper.ratings += actual_ratings

    # once the desired number of movies are rated then it opens a new window with the recommeded movies on the screen
    if len(user_ratings) == len(display_movies_names):
        helper.get_rec(10)
        # here we generate the recommeded movies from the recommendation program
        rec_movies = helper.rec_movies
        predicted_ratings = helper.predicted_ratings
        screen.fill((202, 228, 241))
        exit_text = my_font.render(
            "Based on your ratings we think you might enjoy: ", True, (0, 0, 0))
        screen.blit(exit_text, (0, 0))
        for i in range(10):
            text = my_font.render("{} with predicted rating: {:.6f}".format(rec_movies[i], predicted_ratings[i]), True, (0, 0, 0))
            screen.blit(text, (20, 50 + 50 * i))

        for event in pygame.event.get():
            # quit program
            if event.type == pygame.QUIT:
                run = False
```

```
The movie Nueba Yol (1995) with predicted rating 1
The movie Shadows (Cienie) (1988) with predicted rating 1
The movie Little Indian, Big City (Un indien dans la ville) (1994) with predicted rating 1
The movie Mutters Courage (1995) with predicted rating 1
The top 15 recommendations for you are:
The movie Usual Suspects, The (1995) with predicted rating 5.00
The movie Lamerica (1994) with predicted rating 5.00
The movie Angela (1995) with predicted rating 5.00
The movie Shawshank Redemption, The (1994) with predicted rating 5.00
The movie Schindler's List (1993) with predicted rating 5.00

Process finished with exit code 0
```

Connection to Python debugger failed: Socket closed (today 3:34 PM)

03

Issues

1. We didn’t adopt object oriented programming at the first place, so when we try to glue two parts together, we have to make a new class “UI_helper” to encapsulate the same functionality we had before. Nothing new there, but we have to do it to apply the models to the user interface.
2. Limited inputs from user, thus all recommendations have predicted ratings 5. It’s not a practically useful program, only for the purpose of adding some interactions.
3. The only confirmation of a successful rating that the user gets is the text changing, may lead to user not being confident if their answer was recorded correctly
4. Back buttons are useful, yet we do not have one :(

References & Citations

"Chat.openai.com." *Chat GPT*, https://chat.openai.com/.

Ibtesama. "Getting Started with a Movie Recommendation System." *Kaggle*, Kaggle, 16 May 2020, https://www.kaggle.com/code/ibtesama/getting-started-with-a-movie-recommendation-system/notebook.

"Movielens 100K Dataset." *GroupLens*, 2 Mar. 2021, https://grouplens.org/datasets/movielens/100k/.

"Movielens 1M Dataset." *GroupLens*, 2 Mar. 2021, https://grouplens.org/datasets/movielens/1m/.

Netflix. "Netflix Prize Data." *Kaggle*, 13 Nov. 2019, https://www.kaggle.com/datasets/netflix-inc/netflix-prize-data.

"PyGame Beginner Tutorial in Python - Adding Buttons." *YouTube*, YouTube, 26 May 2021, https://www.youtube.com/watch?v=G8MYGDf_9ho.

Real Python. "Build a Recommendation Engine with Collaborative Filtering." *Real Python*, Real Python, 18 Aug. 2022, https://realpython.com/build-recommendation-engine-collaborative-filtering/#the-dataset.

Rounakbanik. "Movie Recommender Systems." *Kaggle*, Kaggle, 6 Nov. 2017, https://www.kaggle.com/code/rounakbanik/movie-recommender-systems/notebook.

Thank you!