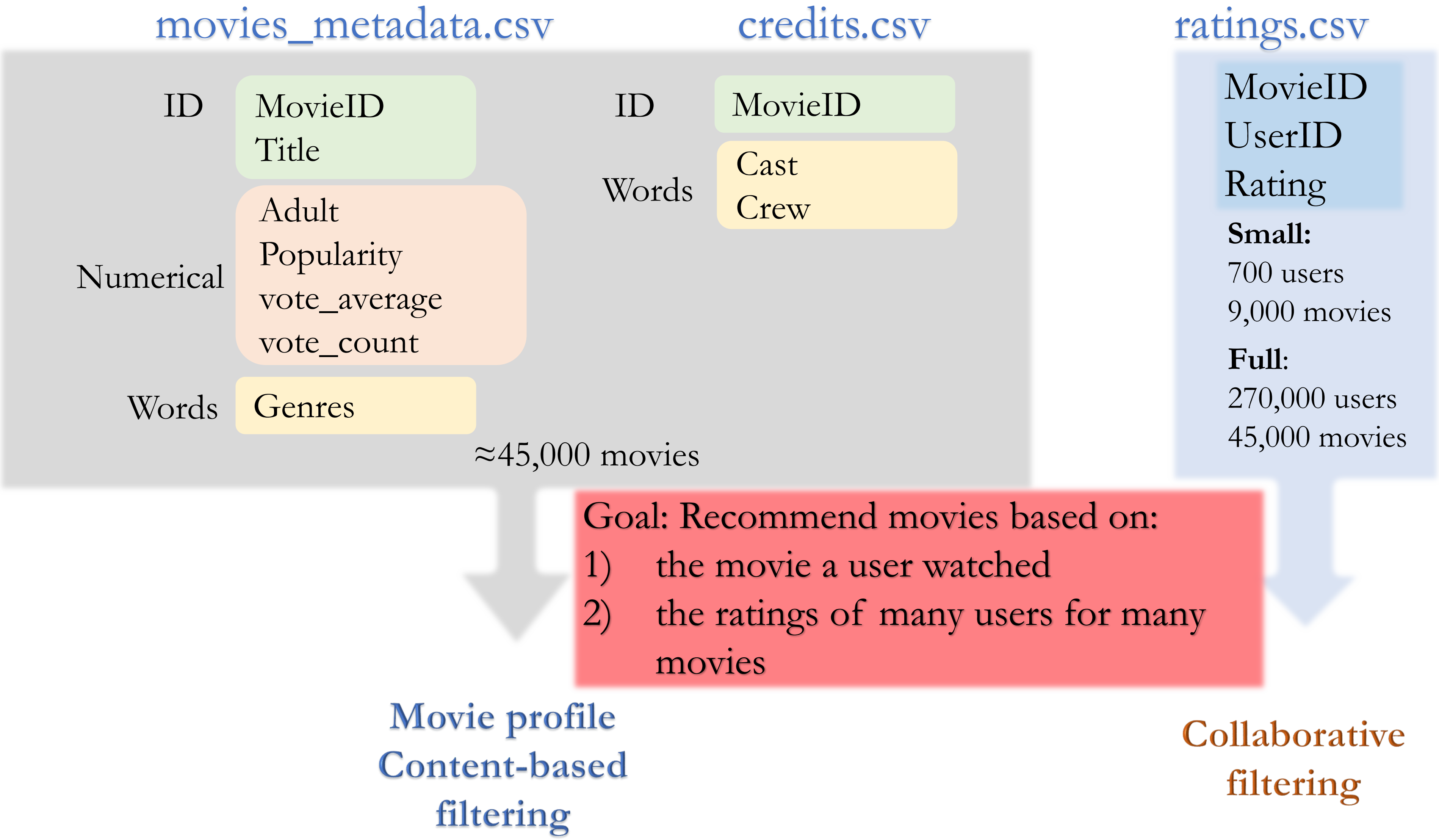
Movie recommendation engine with content-based & collaborative filtering

Big Data Algorithm

Srivardhan Mhetre

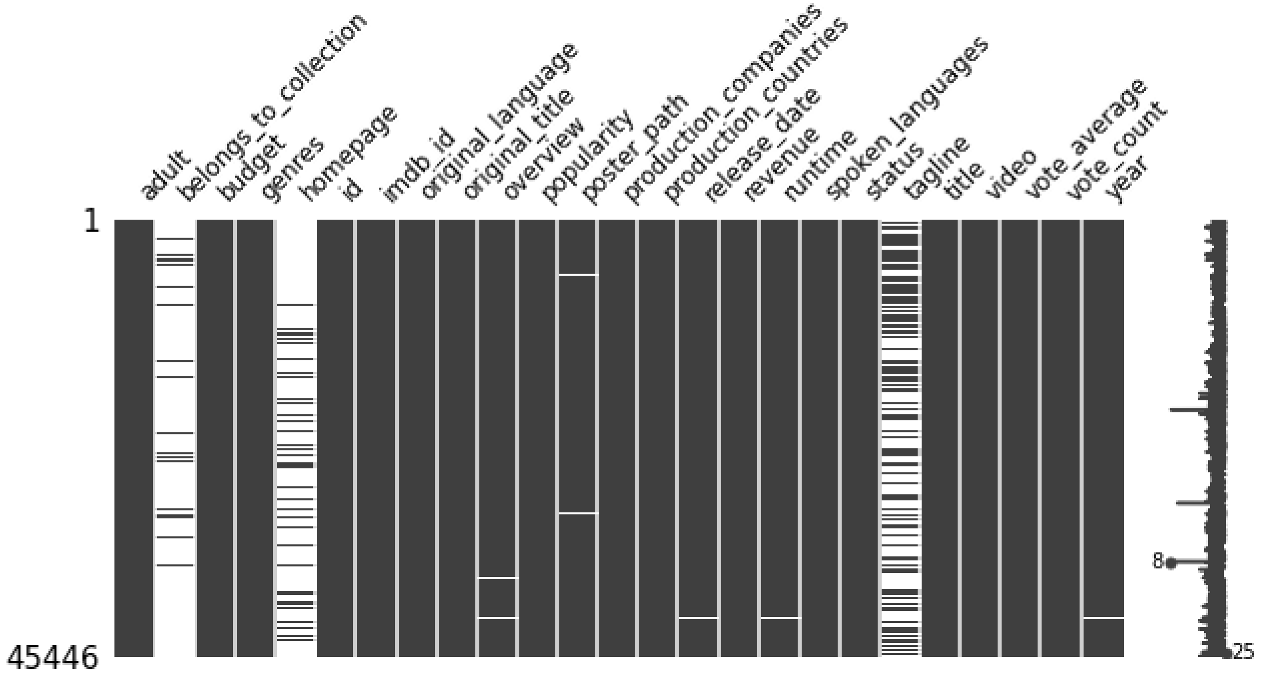
Dec 9, 2021

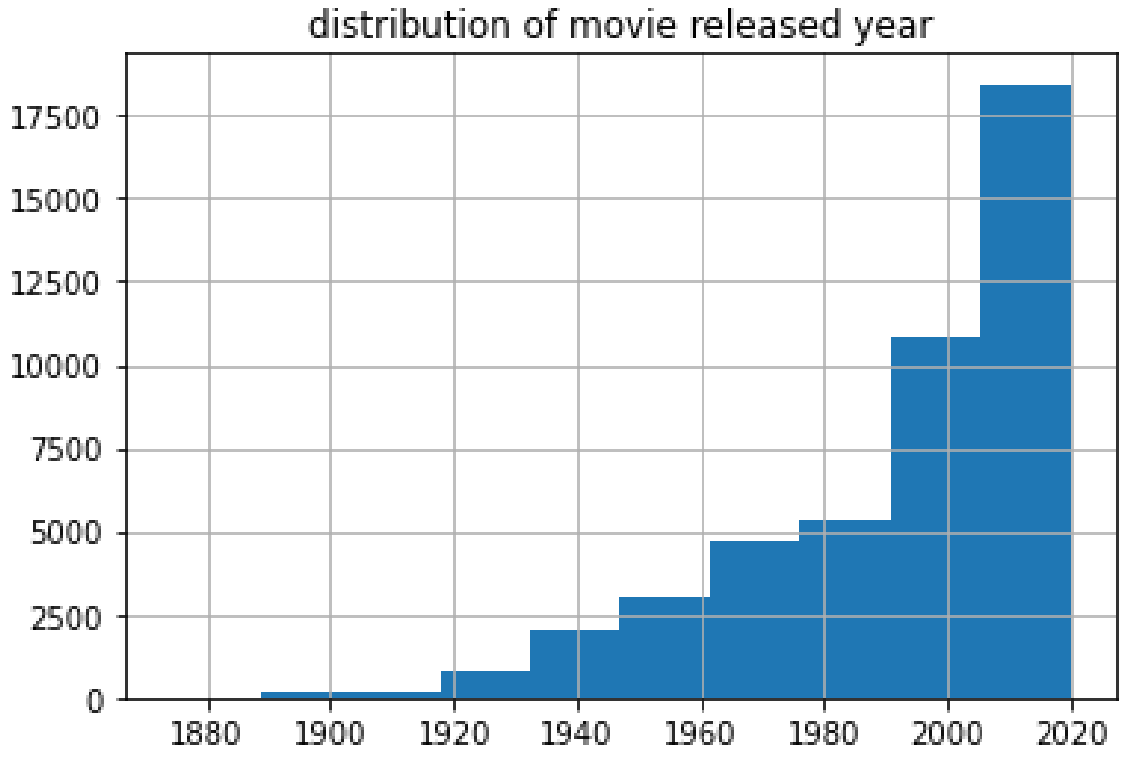
1. Overview


movies_metadata.csv
credits.csv
ratings.csv
ID
MovieID
Title
Numerical
Adult
Popularity
vote_average
vote_count
Words
Genres
≈45,000 movies
ID
MovieID
Words
Cast
Crew
MovieID
UserID
Rating
Small:
700 users
9,000 movies
Full:
270,000 users
45,000 movies
Goal: Recommend movies based on:
1) the movie a user watched
2) the ratings of many users for many movies
Movie profile
Content-based filtering
Collaborative filtering

Basic analysis

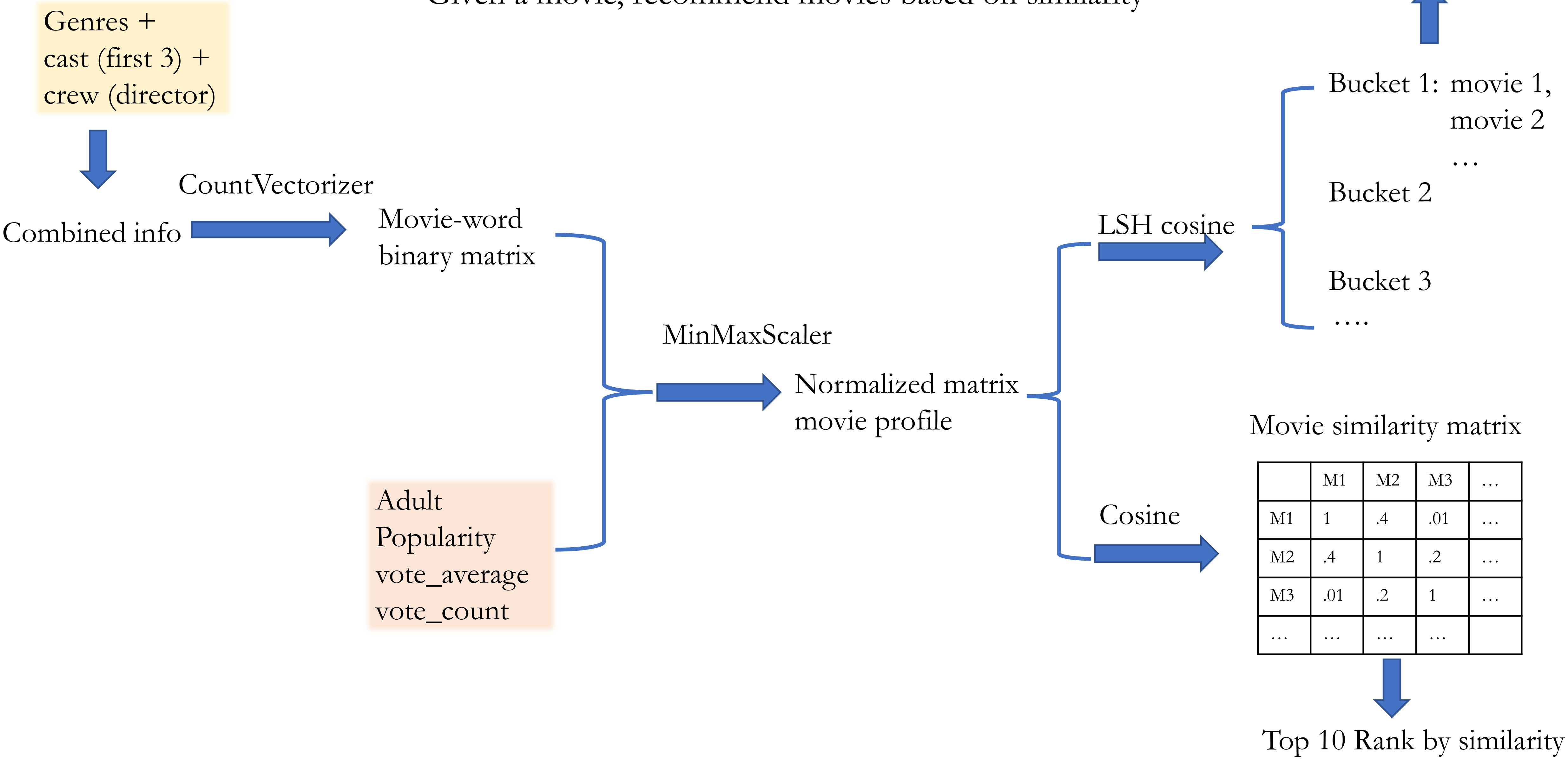
Plot of missing data in each column





2.1 Content-based Filtering - Movie Profile

Given a movie, recommend movies based on similarity



	M1	M2	M3	…
M1	1	.4	.01	…
M2	.4	1	.2	…
M3	.01	.2	1	…
…	…	…	…	

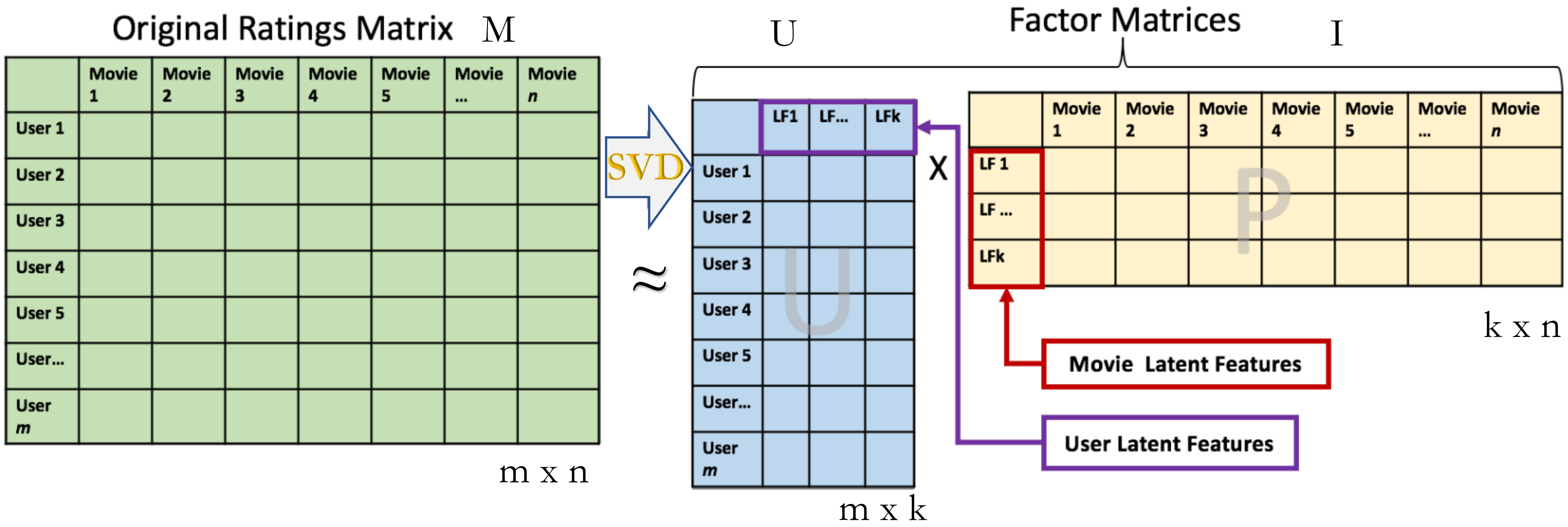
Tips

When dealing with the cast and crew columns:

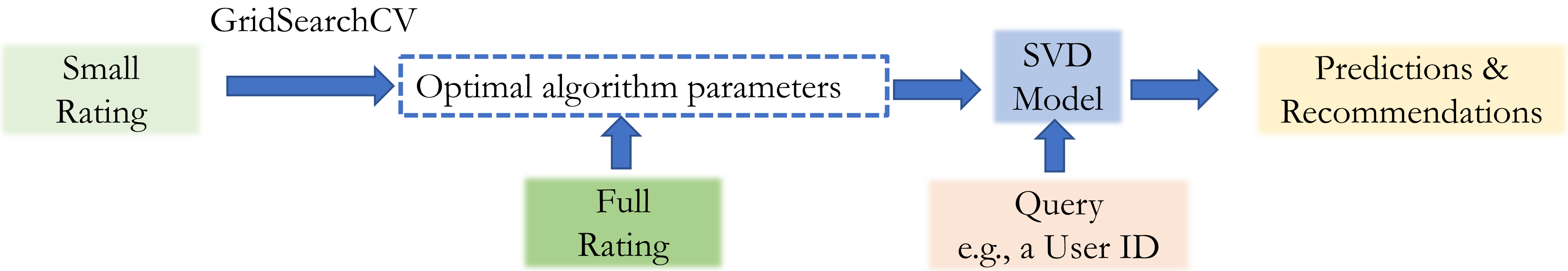
- Remove the space between first and last name, so that two person with same first name and different last name won't be treated as same person.
- Only include the first three actors, the full list will dramatically increase vector dimensions and less meaningful data
- Limit the number of features in CountVectorizer to control the dimensinality

2.2. Collaborative Filtering – Rating Matrix

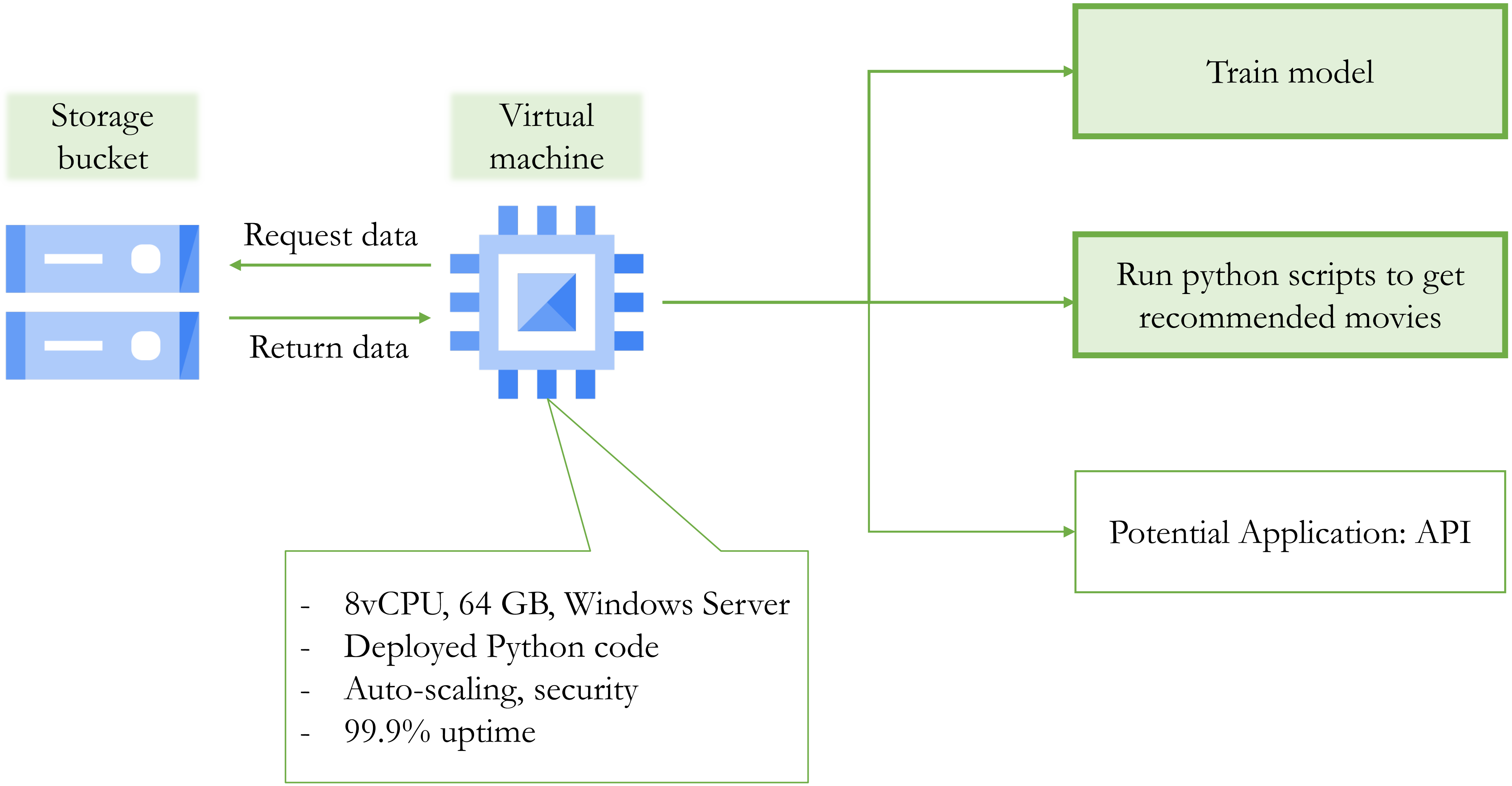
- Using the SVD matrix factorization algorithm



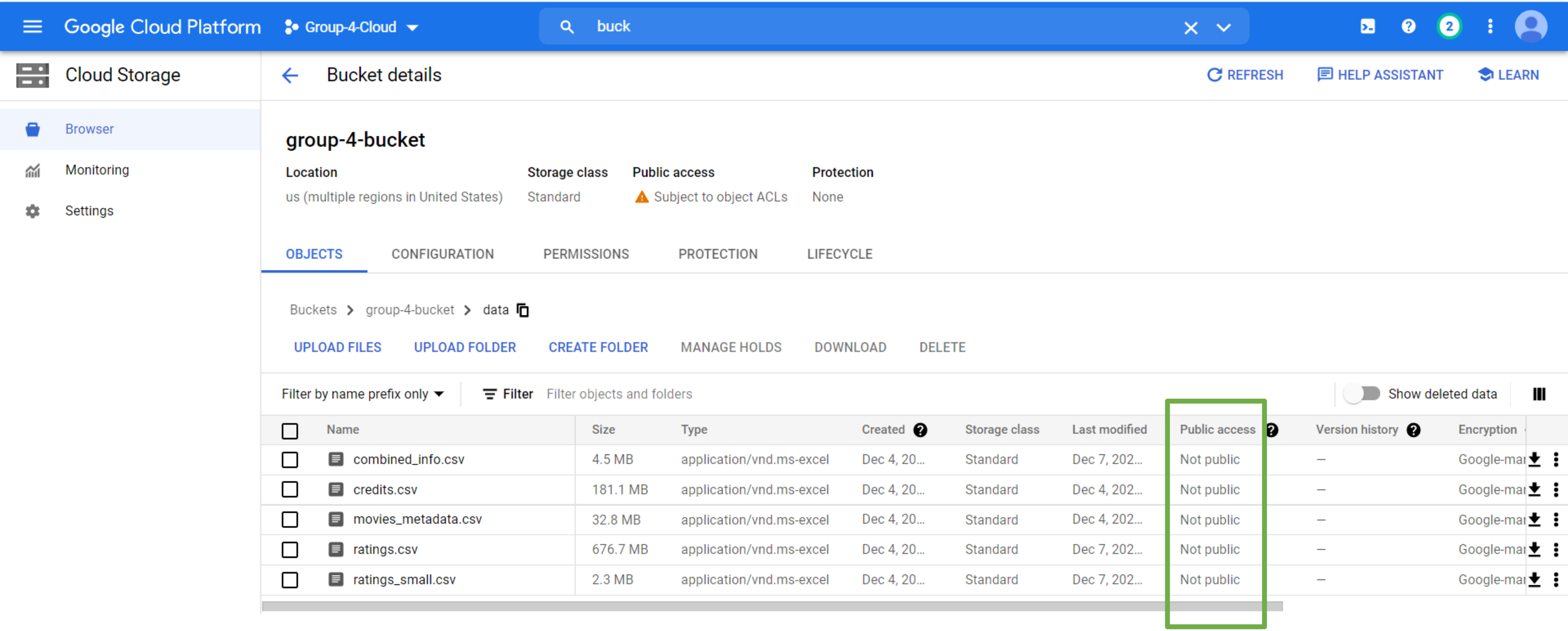
Scikit-surprise package



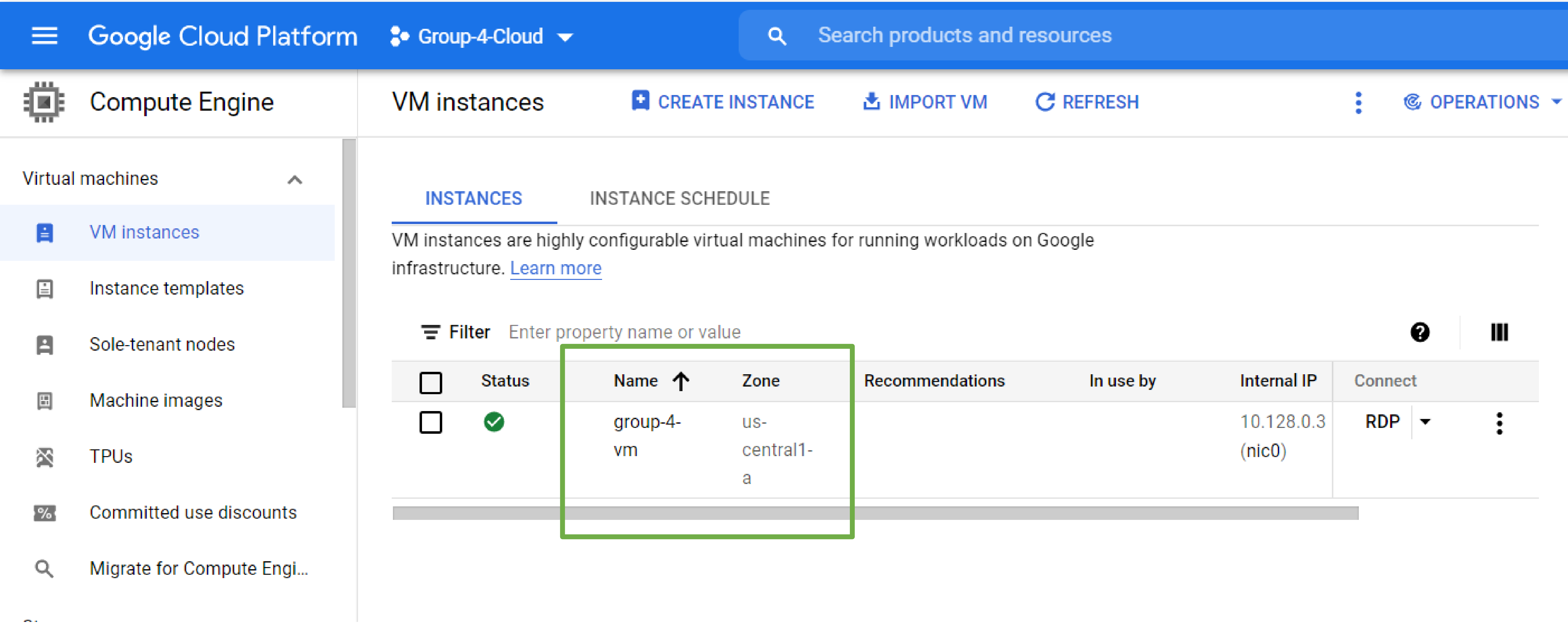
2.3. GCP resources



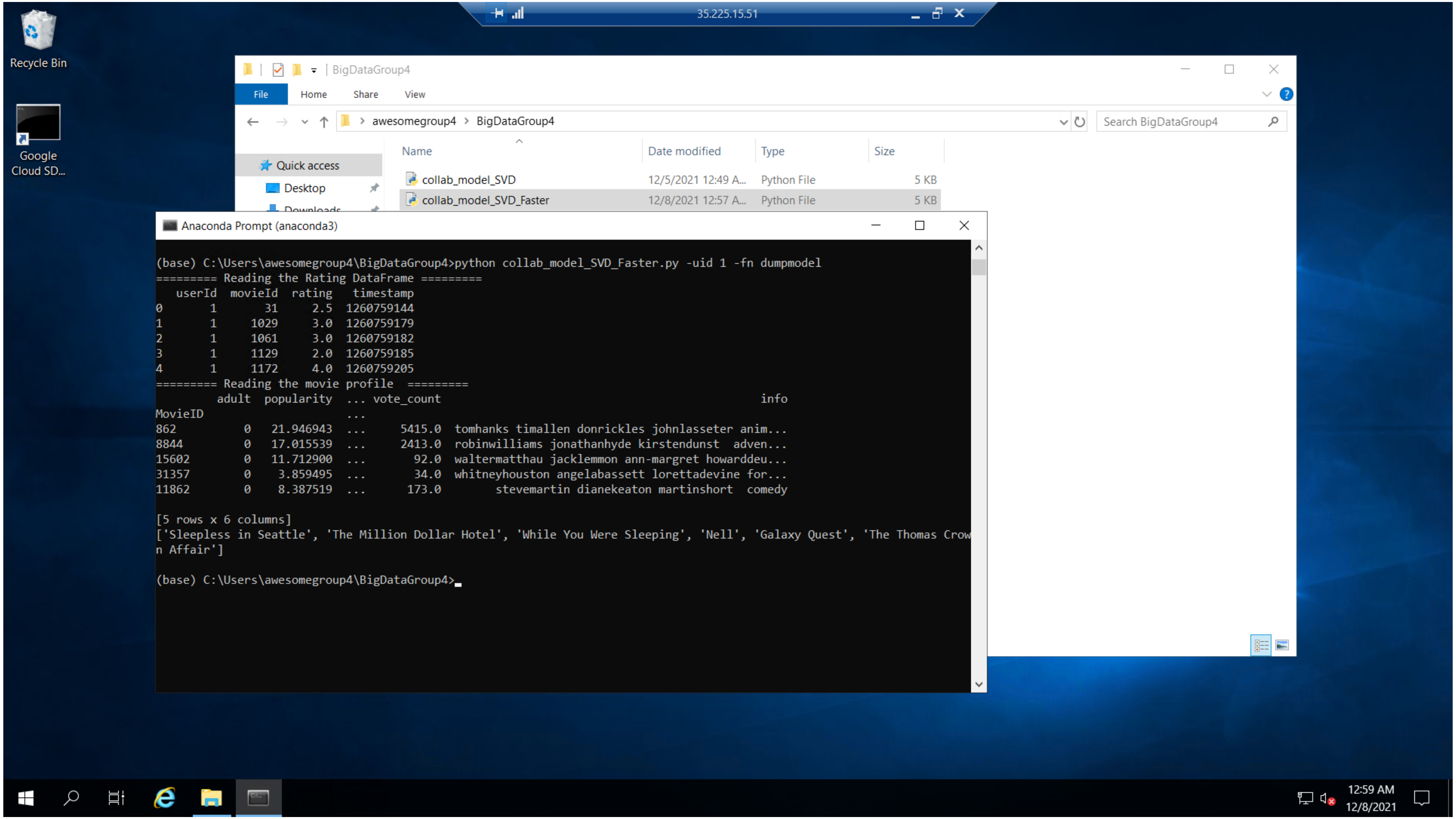
3. Project Interface



3. Project Interface



3. Project Interface



4. Results

4.1 Contend-based filtering recommendations

Using a sample

Cosine similarity

```
1  get_recommendations('Jumanji')
```

```
['The Princess Bride',
 'The Wizard of Oz',
 'Labyrinth',
 'Return to Oz',
 'Small Soldiers',
 'Aladdin and the King of Thieves',
 'The Indian in the Cupboard',
 'The Fifth Element',
 'Back to the Future Part II',
 'Aladdin']
```

LSH cosine

```
1  get_recommendations_popular('Jumanji')
```

```
['Star Wars',
 'The Godfather: Part II',
 'Raiders of the Lost Ark',
 'The Empire Strikes Back',
 'Addams Family Values',
 'Robin Hood: Men in Tights',
 'The Good, the Bad and the Ugly',
 'Field of Dreams',
 'The English Patient',
 'The Princess Bride']
```

```
1  get_recommendations_cosine('Jumanji')
```

```
['Bushwhacked',
 'Disclosure',
 'Before the Rain',
 'Living in Oblivion',
 'First Knight',
 'The Neon Bible',
 'Boys on the Side',
 'Moonlight and Valentino',
 'The Browning Version',
 'The Babysitter']
```

Using the full dataset

Cosine similarity

```
(xl) [tw543@slepner065 code]$ python content_based_gc.py -lsh n -tl Jumanji
=========feature vectorization results ==========
        aamirkhan  aaroneckhart  abernal  ...  vote_average  vote_count  adult
MovieID                                   ...
862             0             0        0  ...           7.7      5415.0      0
8844            0             0        0  ...           6.9      2413.0      0

[2 rows x 1004 columns]
Giving recommendation based on cosine similarity
        aamirkhan  aaroneckhart  abernal  abhishekbachchan  action  adamsandler  adolphemenjou  adrienbrody  ...  zoã  ðµñ  ð¼ð  ð½ð  popularity  vote_average  vote_count  adult
MovieID                                                                                                     ...
862           0.0           0.0      0.0               0.0     0.0          0.0            0.0          0.0  ...  0.0  0.0  0.0  0.0    0.040087          0.77    0.384725    0.0
8844          0.0           0.0      0.0               0.0     0.0          0.0            0.0          0.0  ...  0.0  0.0  0.0  0.0    0.031079          0.69    0.171439    0.0
15602         0.0           0.0      0.0               0.0     0.0          0.0            0.0          0.0  ...  0.0  0.0  0.0  0.0    0.021394          0.65    0.006536    0.0
31357         0.0           0.0      0.0               0.0     0.0          0.0            0.0          0.0  ...  0.0  0.0  0.0  0.0    0.007049          0.61    0.002416    0.0
11862         0.0           0.0      0.0               0.0     0.0          0.0            0.0          0.0  ...  0.0  0.0  0.0  0.0    0.015320          0.57    0.012291    0.0

[5 rows x 1004 columns]
(45502, 1001)
['Clash of the Titans', 'Paws', 'Aladdin and the King of Thieves', "Halloweentown II: Kalabar's Revenge", 'Snow Queen', 'The Wiz', "The Shamer's Daughter", 'Peter Pan', 'Jack and the Beansta
lk', 'The Slipper and the Rose']
--- 33.92224407196045 seconds ---
```

Pitfalls of LSH

```
(xl) [tw543@hal0034 code]$ python content_based_gc.py -lsh y -st pop
         aamirkhan  aaroneckhart  abernal  ...  vote_average  vote_count  adult
MovieID                                    ...
862              0             0        0  ...           7.7      5415.0      0
8844             0             0        0  ...           6.9      2413.0      0

[2 rows x 1004 columns]
Number of buckets:  17927
Number of candidate pairs:  168143393
```

Forming candidate pairs of movies in the same bucket are time-consuming for very large dataset.
Find the buckets a movie falls in and search candidate pairs is complex, an item can fall into many buckets (the number of bands used in LSH)

4.2 Collaborative filtering recommendations

Finding optimal parameter using GridSearchCV and small rating dataset

```python
from surprise import SVD

param_grid = {
    "n_epochs": [5, 10],
    "lr_all": [0.002, 0.005],
    "reg_all": [0.4, 0.6]
}
gs = GridSearchCV(SVD, param_grid, measures=["rmse", "mae"], cv=3)

gs.fit(data)

print(gs.best_score["rmse"])
print(gs.best_params["rmse"])
```

```
0.9136866638671162
{'n_epochs': 10, 'lr_all': 0.005, 'reg_all': 0.4}
```

Training model for full rating dataset

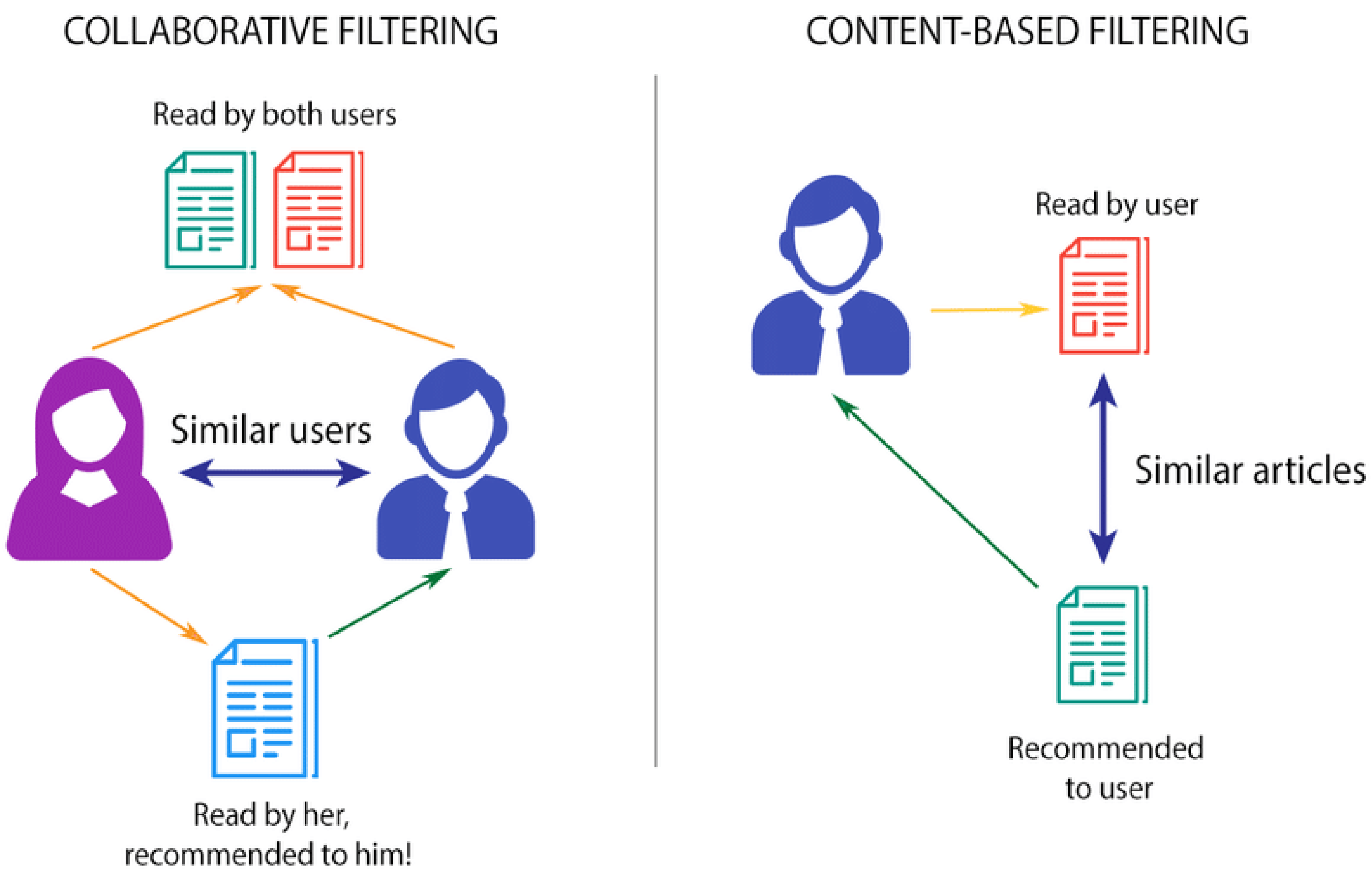
```
(xl) [tw543@slepner065 code]$ python collab_model_SVD_gc.py -uid 1 -fn full
========= Reading the Rating DataFrame =========
   userId  movieId  rating   timestamp
0       1      110     1.0  1425941529
1       1      147     4.5  1425942435
2       1      858     5.0  1425941523
3       1     1221     5.0  1425941546
4       1     1246     5.0  1425941556
========= Reading the movie profile  =========
          adult  ...                                                  info
ovieID           ...
62            0  ...  tomhanks timallen donrickles johnlasseter anim...
844           0  ...  robinwilliams jonathanhyde kirstendunst  adven...
15602         0  ...  waltermatthau jacklemmon ann-margret howarddeu...
31357         0  ...  whitneyhouston angelabassett lorettadevine for...
11862         0  ...        stevemartin dianekeaton martinshort  comedy

[5 rows x 6 columns]
========= Training model using SVD algorithm  =========
The dump has been saved as file /home/tw543/Xuelian/model/full
['The Million Dollar Hotel', 'Sleepless in Seattle']
--- 807.1520252227783 seconds ---
```

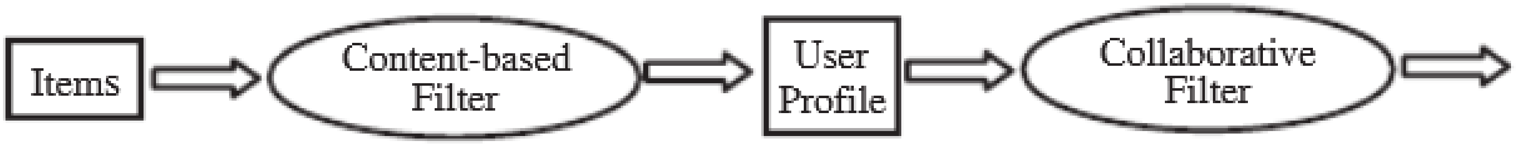
4.3 Comparison of Content-based & Collaborative filtering

Our project includes both Collaborative filtering and Content-based filtering. Each of them has their own advantages in some situations. Most of the modern recommender systems combine both of these approaches to make a robust hybrid recommender.

The greatest advantage of Collaborative Filtering is that it does not suffer from cold start problems because the features are based on the characteristics of the data. But the model has limited ability to expand on the user's existing interests.
On the other hand, Collaborative Filtering has an advantage over the quality of recommendation results because the model can help users discover new interests. However, it suffers from cold-start problem because it requires the data from the users.

So, the best approach is to combine both these methods. However, we have implemented it separately because of huge size of the dataset which greatly eliminates the cold-start problem in Collaborative filtering.


COLLABORATIVE FILTERING
Read by both users
Similar users
Read by her,
recommended to him!
CONTENT-BASED FILTERING
Read by user
Similar articles
Recommended
to user


Items
Content-based
Filter
User
Profile
Collaborative
Filter

Thank you