CS 614 – Applications of ML

Project 2 – Recommender System

Submitted by: Manil Shrestha

<u>Project: Anime Recommendation System</u>

1. Pitch:

State the circumstances an organization or sets of users would be willing to fund the proposed ML application based on their perceived value. Value could be financial, or any other type of outcome organizations or users need to obtain (e.g., institutional image, customer satisfaction, increased quality, or efficiency).

Any video streaming platform anime recommendation system would be eager to fund a recommender system since they want to provide personalized recommendations to their users, increase user engagement, improve user satisfaction and retention, or drive revenue through increased sales or subscriptions. Additionally, an anime recommendation system may be of interest to organizations or users who want to analyze user behavior and preferences to gain insights into their target audience.

2. Data source:

Indicate with a link where you obtained the data. If you generated the data yourself, please provide a link to the code of the used approach.

This dataset was obtained from a Kaggle competition:
https://www.kaggle.com/datasets/CooperUnion/anime-recommendations-database
The recommendation data was compiled from www.myanimelist.net.

3. Model and data justification:

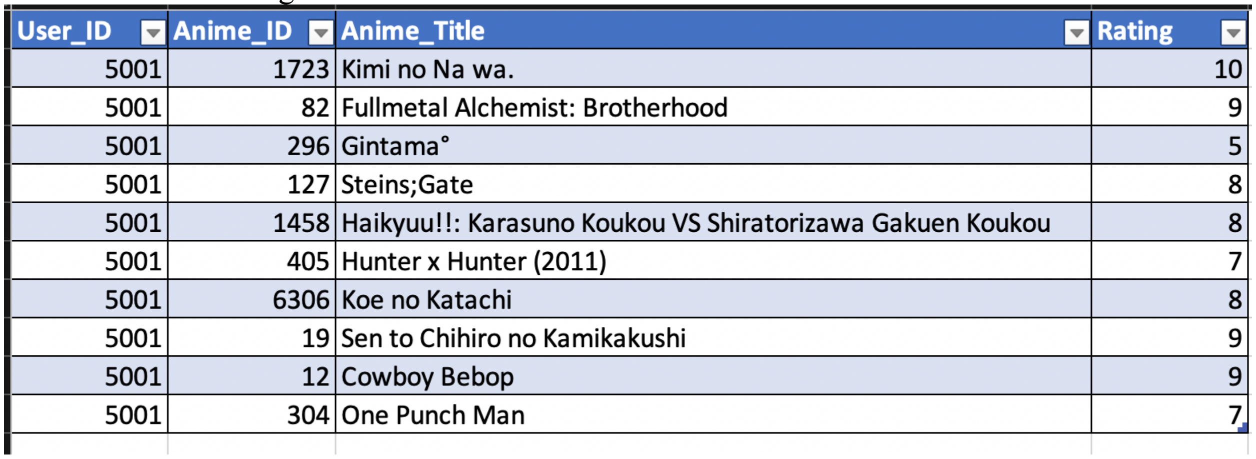
Justify why you chose a specific model to learn from the selected data. If you learned that a given more would be suitable for a type of data from a publication, please provide a link to the source. Note you still have to justify it with your own words (as always, limit to three sentences).

Since this was my first recommendation system modeling, I went with a simple but effective way which is through Singular Value Decomposition and used Surprise python library (Hug, 2020). Singular Value Decomposition (SVD) is a matrix factorization technique used in recommender systems to identify latent factors or features that capture the preferences and characteristics of users and items. This is a collaborative filtering approach, and I chose this method because it can effectively handle missing data and sparsity in user-item rating. Additionally, the team that won the Netflix grand prize in 2009 used SVD method along with ensemble of ML techniques. (Toscher et al., 2009).

4. Commented examples:

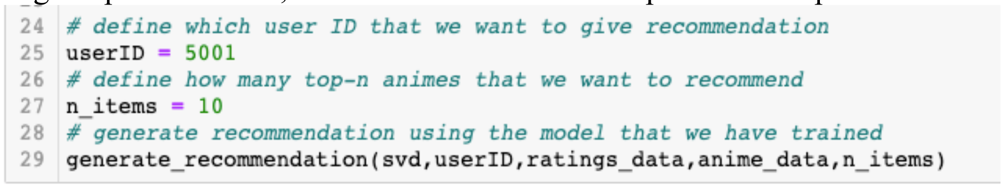
Indicate the input where trained model is applied, the output and whether it is as expected or any observations you may have.

Cold start problem has not been addressed in this project. So, to make recommendation, a new user was simulated (a friend of mine who is an avid anime fan). They gave me their ratings for 10 anime whose rating I stored in the dataset:

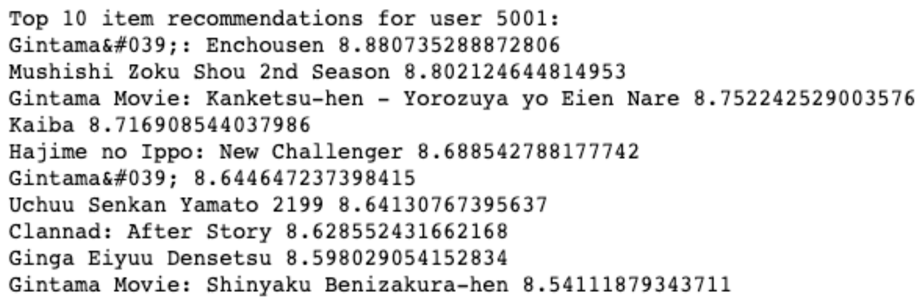


User_ID	Anime_ID	Anime_Title	Rating
5001	1723	Kimi no Na wa.	10
5001	82	Fullmetal Alchemist: Brotherhood	9
5001	296	Gintama°	5
5001	127	Steins;Gate	8
5001	1458	Haikyuu!!: Karasuno Koukou VS Shiratorizawa Gakuen Koukou	8
5001	405	Hunter x Hunter (2011)	7
5001	6306	Koe no Katachi	8
5001	19	Sen to Chihiro no Kamikakushi	9
5001	12	Cowboy Bebop	9
5001	304	One Punch Man	7

Using the provided data, I used the trained model to predict the top-10 anime for the user 5001:



```python
# define which user ID that we want to give recommendation
userID = 5001
# define how many top-n animes that we want to recommend
n_items = 10
# generate recommendation using the model that we have trained
generate_recommendation(svd,userID,ratings_data,anime_data,n_items)
```



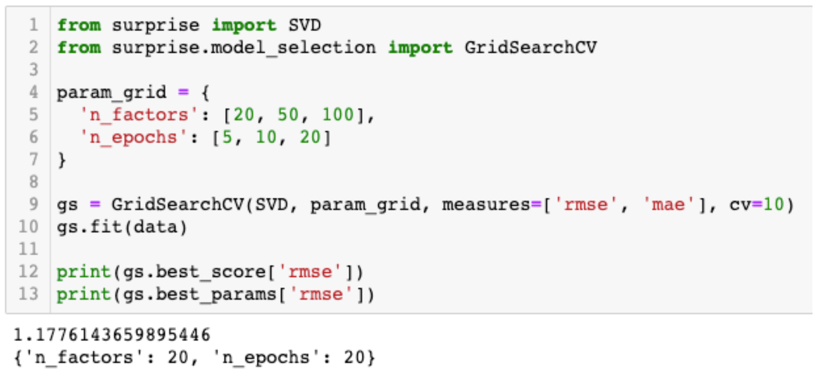
```
Top 10 item recommendations for user 5001:
Gintama&#039;: Enchousen 8.880735288872806
Mushishi Zoku Shou 2nd Season 8.802124644814953
Gintama Movie: Kanketsu-hen - Yorozuya yo Eien Nare 8.752242529003576
Kaiba 8.716908544037986
Hajime no Ippo: New Challenger 8.688542788177742
Gintama&#039; 8.644647237398415
Uchuu Senkan Yamato 2199 8.64130767395637
Clannad: After Story 8.628552431662168
Ginga Eiyuu Densetsu 8.598029054152834
Gintama Movie: Shinyaku Benizakura-hen 8.54111879343711
```

I sent them this list and they were agreeable on 8 of the 10 ratings provided by the model.

5. Testing:

Provide a confusion matrix with one or more metrics and comment the results.

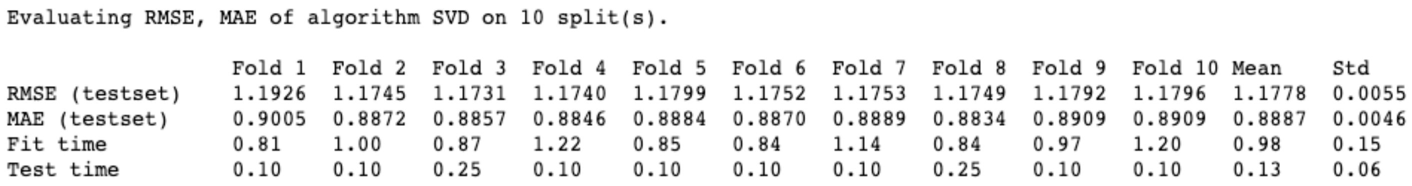
For this assignment, instead of confusion matrix I conducted a 10-fold cross validation of Root Mean Square Error (RMSE) and Mean Absolute Error (MAE).
First, a grid search was performed to find the best performing hyperparameter:



```python
from surprise import SVD
from surprise.model_selection import GridSearchCV

param_grid = {
  'n_factors': [20, 50, 100],
  'n_epochs': [5, 10, 20]
}

gs = GridSearchCV(SVD, param_grid, measures=['rmse', 'mae'], cv=10)
gs.fit(data)

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

```
1.1776143659895446
{'n_factors': 20, 'n_epochs': 20}
```

Fig 1: Grid search for best hyperparameter of n_factors and epochs for SVD



```
Evaluating RMSE, MAE of algorithm SVD on 10 split(s).

                  Fold 1  Fold 2  Fold 3  Fold 4  Fold 5  Fold 6  Fold 7  Fold 8  Fold 9  Fold 10 Mean    Std
RMSE (testset)    1.1926  1.1745  1.1731  1.1740  1.1799  1.1752  1.1753  1.1749  1.1792  1.1796  1.1778  0.0055
MAE (testset)     0.9005  0.8872  0.8857  0.8846  0.8884  0.8870  0.8889  0.8834  0.8909  0.8909  0.8887  0.0046
Fit time          0.81    1.00    0.87    1.22    0.85    0.84    1.14    0.84    0.97    1.20    0.98    0.15
Test time         0.10    0.10    0.25    0.10    0.10    0.10    0.10    0.25    0.10    0.10    0.13    0.06
```

Fig 2: 10-fold cross validation on the SVD algorithm trained with 20 factors and 20 epochs.

6. Code and instructions to run it:

Provide a link to the code and any required instructions to run it. Please include some testing examples so we can quickly experience what you experienced with the model.

Here is the link to Jupyter Notebook which has been uploaded in the github:
https://github.com/ManilShrestha/AppliedMLProjects/blob/main/RecommenderSystem/Anime-RecSys-(Surprise).ipynb

You can recreate the training and experiments by following the steps:

1. Download and unzip the file from: https://github.com/ManilShrestha/AppliedMLProjects/blob/main/RecommenderSystem/data/anime.zip
2. Maintain the data directory structure as './data/anime/anime_ratings.dat' and './data/anime/anime_info.dat' relative to the jupyter notebook.
3. Make sure all the dependencies are installed in your machine.
4. Run all the cells in the notebook.

References

Töscher, Andreas, Michael Jahrer, and Robert M. Bell. "The bigchaos solution to the netflix grand prize." *Netflix prize documentation* (2009): 1-52.

Hug, Nicolas. "Surprise: A Python library for recommender systems." *Journal of Open Source Software* 5.52 (2020): 2174.