VIETNAM NATIONAL UNIVERSITY

UNIVERSITY OF SCIENCE - HCM CITY

DEPARTMENT OF INFORMATION TECHNOLOGY

COMPUTER SCIENCE

# Course project report

**Topic: Big Data Application: Real-problem Project**

**Course: Big Data Application**

*Students:*
Nguyễn Thế Thiện (21127170)
Nguyễn Trần Trung Kiên (21127327)
Châu Tấn Kiệt (21127329)

*Guidance instructors:*
Dr. Nguyễn Ngọc Thảo
Dr. Bùi Duy Đăng

27th April 2025

# Contents

## 1 Abstract

The project is about the application of Big Data knowledge and techniques into a real world problem we have been conducting for approximately two months. The report is a full overview of the researches that we made, the results we got apart from what we initially expected, and our work distribution among the members, as well as our own evaluations on our work and what we could have done better.

## 2 Problem

### 2.1 Main problem

**Problem**. An anime recommendation for users based on user ratings.
**Description.** Analyze and process user ratings on anime titles using Big Data and Machine Learning tools, in order to recommend animes the user has yet to watch, based on that user's rating history.
**Input**. Existing user ID in the rating dataset.
**Output**. A ranking list for recommended animes fitting the user's taste.

### 2.2 Experiment settings

#### 2.2.1 Datasets

**Anime Dataset 2023** [1], is a collection of user and anime ratings on one of the largest anime databases and communities - MyAnimeList.
The main files from the dataset that are used;

1. **users-score-2023.csv** (1.16 GB): The main data consisting of user ratings on anime titles, provided by 270K users on 16K anime titles, with a total of 24.3M samples, 99.48% sparsity rate for only the observed users and anime titles. The table consists of 5 columns:
   (a) Rating (0-10): User's rating on anime title.
   (b) Anime ID, Anime Name, User ID, Username: Details of users and animes in the ratings.
2. **anime-dataset-2023.csv** (15.92 MB): Details of around 25K anime titles on MyAnimeList. The table consists of 16 columns:
   (a) Mal ID, Username: User's ID and name on the platform.
   (b) Gender, Birthday, Location: Personal information of the user, given voluntarily.
   (c) Joined, Days Watched, Mean Score, Watching, Completed, On Hold, Dropped, Plan to Watch, Total Entries, Rewatched, Episodes Watched: User's behaviors on the platform.

[1] Anime Dataset 2023, from Kaggle, by username Sajid, https://www.kaggle.com/datasets/dbdmobile/myanimelist-dataset

3. **user-details-2023.csv**. (73.93MB): Details of around 730K users registered on MyAnimeList. The table consists of 24 columns:
   (a) Anime ID, Name, English name, Other name: Anime ID, original name, translated English name and another (usually abbreviated) name it is known of.
   (b) Score, Rank, Popularity, Favorites, Scored by, Members: Anime title's mean score, ranking and side information, on the platform.
   (c) Genres, Synopsis, Type, Episodes, Aired, Premiered, Status Producers, Licensors, Studios, Source, Duration, Rating, Image URL: Extra information on the anime title.

### 2.2.2 Main tasks

1. Data preprocessing: Apache Spark preprocesses the datasets retrieved from static les, reformat, clean and prepare the raw data before feeding into the recommendation system.
2. RS model training: Train the dataset on ALS model from Apache MLlib.
3. RS in use: Input user's rating history to predict a ranking list for recommended animes which user has not watched.
4. Dashboard: Visualize predictions and trends with NetworkX-assisted Matplotlib.

# 3 Application

This section shall list and explain the applied knowledge and tools we used throughout the final submission of the project. Our final version for this project consists of the following sections:

- Data pre-processing: transformation of the original dataset files in .csv formats.
- Models: recommendation models used for solving the problem.
- Results: discussions on the actual results we got from applying the solutions, in comparison to our expected results.

## 3.1 Data pre-processing

The source code and line-by-line analysis are in $src/data_preprocessing.ipynb.Themainobjectiveofthissectionis$
$processthedatasetsonebyone, beforejoiningintooneunifieddatasetforALSmodel.$

$Thisisthesummaryforwhatwedidforpre-processing:$

***User dataset***

1. ***Mal ID****: User's ID. Renamed for easier processing later.*
2. ***Username****: Removed for its uselessness in content-based analysis.*
3. ***Gender****: One-hot encoding.*
4. ***Birthday****: Unix timestamp conversion, fill missing, quantile-based discretization.*
5. ***Location****: Removed for inconsistent format and incomprehensible content.*
6. ***Days Watched****: categorization using kNN.*
7. ***Mean Score****: Removed as the user's rating mean is already reflected in the rating dataset.*
8. ***Watching, On Hold, Dropped, Plan to Watch, Total Entries, Completed, Rewatched, Episodes Watched****: Correlation analysis, fill missing values and PCA for dimension reduction.*

***Anime dataset***

1. ***Name, English name, Other name****: Removed for their uselessness in content-based analysis.*
2. ***Score****: Removed as anime rating mean is already reflected in the rating dataset.*
3. ***Genres****: One-hot encoding.*
4. ***Synopsis****: Removed due to complexity and lack of time for experimentation.*
5. ***Type****: Fill missing using mode, one-hot encoding.*
6. ***Episodes, Aired****: Convert Aired to timestamps, use each other to fill missing values before filling the rest based on mean.*
7. ***Premiered****: Convert from categorical to continuous.*
8. ***Status, Producers, Licensors, Studios, Source****: One-hot encoding with minimum support to reduce exploding dimensions.*

9. ***Duration***: *Reformatting.*
10. ***Rating, Rank, Popularity***: *Removed due to little information they hold on their own, as well as can easily be reflected by other metrics.*
11. ***Favorites, Members***: *Log transformation.*
12. ***Scored by***: *Removed as it is count of users that rated the anime title, which is reflected in the rating dataset.*
13. ***Image URL***: *Removed as it is anime poster.*

*Rating dataset:*

1. ***Username, Anime Title***: *Removed as there exists* $user_idandanime_idforuniqueentries$.

2. *Unified dataset:*
   1. ***Anime, User datasets***: *PCA to convert features into one feature.*
   2. *Join all 3 datasets, then PCA to convert all remaining features into one replication of Rating column.*

## 3.2 Models

The goal of Alternating Least Squares (ALS) is to find two matrices, $U$ and $P$, such that their product is approximately equal to the original user-item rating matrix. Once such matrices have been found, we are able to predict what user $i$ will think of item $j$ by multiplying row $i$ of $U$ with row $j$ of $P$. [1]
Suppose:

- $R$ is the **user-item rating matrix** of shape $(m \times n)$, where:
  - $m$ = number of users,
  - $n$ = number of items.

ALS tries to approximate:

$$R \approx U \times P^T$$

where:

- $U$ is the **user factors matrix** of shape $(m \times k)$,
- $P$ is the **item factors matrix** of shape $(n \times k)$,
- $k$ is the number of **latent factors**.

Each **row** of $U$ represents the **latent vector** for a user (user preferences).
Each **row** of $P$ represents the **latent vector** for an item (item features).

**Prediction process:**
For a specific user $i$ and item $j$:

- The **predicted rating** is the **dot product** of:
  - the $i$-th row of $U$,
  - and the $j$-th row of $P$.

This means multiplying the corresponding elements of the user and item vectors and summing the results.

**Training ALS:**

- Fix $P$, solve for $U$ using least squares.
- Fix $U$, solve for $P$ using least squares.
- Alternate these two steps until convergence.

## 3.3 ALS model training process

### 3.3.1 Data Preparation

The dataset utilized consists of two tables:

- **rating_df**: Contains user ratings for anime, including the fields *user_id*, *anime_id*, and *rating*.
- **anime_df**: Contains detailed information about each anime title.

The data, after preprocessing, is saved in `.csv` format to facilitate the model training process.

### 3.3.2 Data Splitting

The dataset is randomly split into three parts with the ratio of 75%, 25%, 5% based on the dataset division of the Graph-less Collaborative Filtering paper[3] (the paper also proposed a model for the recommendation task) combined with a fixed seed of 42 to ensure reproducibility:

- 70% for the training set,
- 5% for the validation set,
- 25% for the test set.

### 3.3.3 ALS Model Training

The proposed model is based on the *Alternating Least Squares (ALS)* algorithm implemented in `pyspark.ml.recommendation`. The main hyperparameters are configured as follows:

- `userCol="user_id"`, `itemCol="anime_id"`, `ratingCol="rating"`: Specify the corresponding data columns.
- `nonnegative=True`: Constrain the learned factors to be non-negative.
- `coldStartStrategy="drop"`: Drop cold-start users/items during prediction.

- `maxIter=10`: Maximum number of iterations for optimization.
- `regParam=0.1`: Regularization parameter to prevent overfitting.
- `default rank=10`: Number of latent factors to factorize the matrix.

The model is trained on the training set using the `fit()` method.

#### 3.3.4 Model Evaluation Metrics

The performance is measured using the Root Mean Squared Error (RMSE):

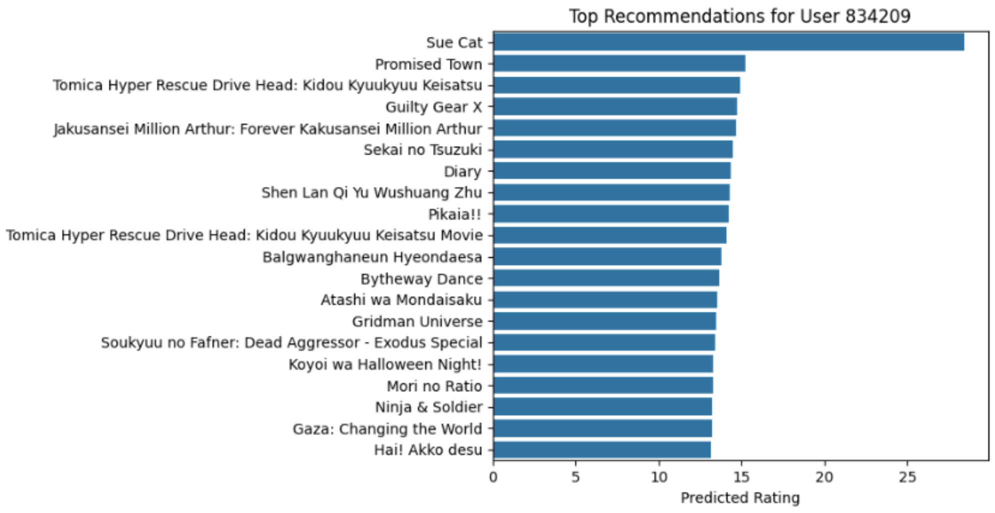
$$\text{RMSE} = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(y_{\text{true},i} - y_{\text{pred},i})^2}$$

Where:

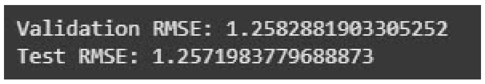
- $N$ is the number of test samples,
- $y_{\text{true},i}$ is the true rating for the $i$-th sample,
- $y_{\text{pred},i}$ is the predicted rating for the $i$-th sample.

## 3.4 Results

### 3.4.1 Output



- Above is the model output for the user with user_id 834209
- The result is a list of 20 animes that the model rated as suitable for that user and the predicted rating score of each anime.

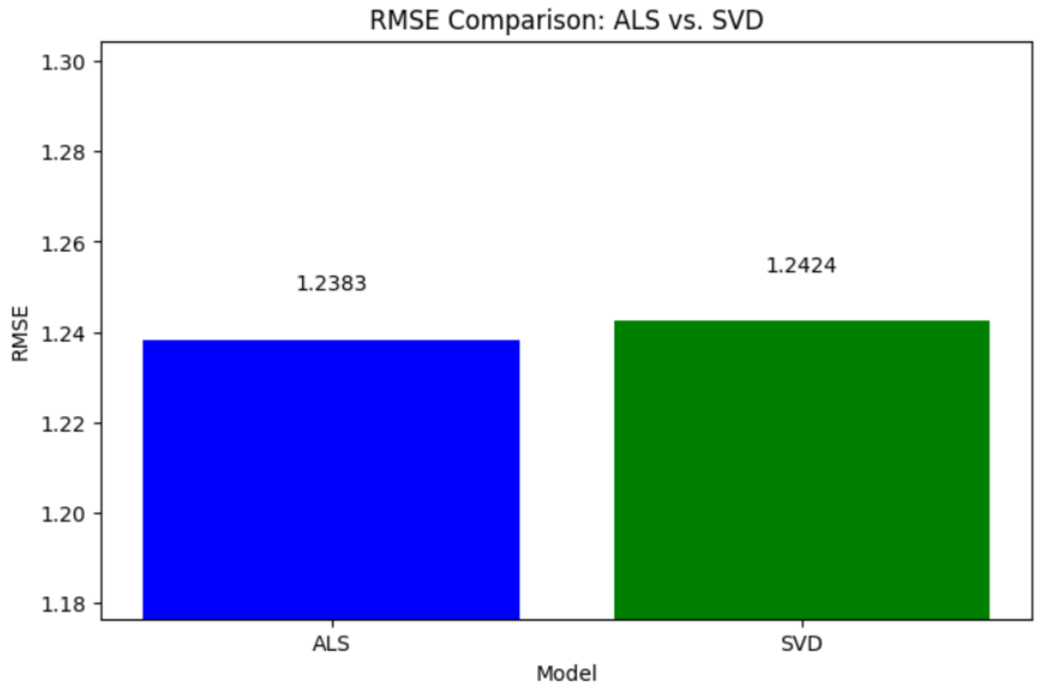


```
Validation RMSE: 1.2582881903305252
Test RMSE: 1.2571983779688873
```

- Above is the RMSE score on the 2 Validation and Test sets of the Model after the training process.
- The RMSE of the Validation and Test sets are very close (the difference is only about 0.00109), which shows that the model is not overfitting (too good for the training set) and generalizes well to new data (test set).
- In the recommendation system, an RMSE of around 1–2 can be considered good. The model's RMSE of 1.25 shows that the model is performing quite well.

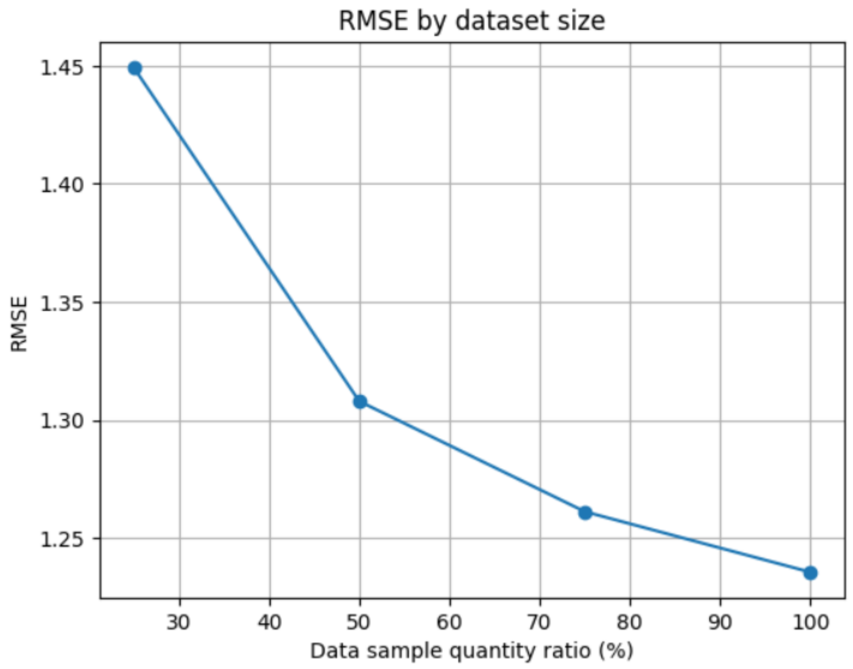
### 3.4.2 Model Evaluation

We compare the model used in this project (ALS) with the SVD model [2], which is also quite popular for recommendation tasks. We use the original dataset and split it into 80% train: 20% test ratio to evaluate the RMSE of the 2 models to see how effective the 2 models are on large datasets (>1GB)



- Based on the RMSE graph between the two models executed on this dataset, we can see that the ALS model gets better results than the SVD model.
- Thus, it can be seen that choosing the ALS model for this project is more suitable than SVD.

Next, we evaluate the model performance according to the size of the dataset by taking 25%, 50%, 75%, and 100% of the dataset size for training and testing.



- The evaluation will provide insights into how the model's performance changes with the increasing size of the dataset
- As we can see, as the dataset size increases, the RMSE decreases, which means the model accuracy is improved.
- This demonstrates that the model is suitable for handling big datasets.

# 4 Group work

## 4.1 Plans

### 4.1.1 Proposal

This is summarized from our original proposal that we submitted on March 2nd, 2025.

**Problem.** The problem was original "Real-time anime recommendation system (RS) based on user ratings". It was about a RS, after trained on the original dataset made of user ratings on anime titles, is updated as new rating is created and new data is ingested into the database, hence "real-time". The RS shall be used for anime recommendation.

**Main tasks**.

- **Real-time data ingestion**. Set up Redis database and simulate real-time data ingestion with data from the dataset files.
- **Data streaming pre-processing**. Apache Spark Streaming simulates real-time data streaming from Redis database, then pre-processes the new data before feeding into the recommendation system.
- **Real-time RS model training**. Pre-built model from Apache MLlib is trained and updated by feeding real-time data, from Spark Streaming.
- **RS in use**. Input and embed user's rating history to predict a ranking list for recommended animes which user has not watched.
- **Real-time dashboard of predictions**. Visualize analyzed data and predictions with NetworkX and Matplotlib.

### 4.1.2 Final work

**Problem**. The problem has been changed through numerous occasions, to "Anime recommendation system (RS) based on user ratings for existing users". Since the need of a real-time RS is dropped, it is now simply a RS trained on a static dataset of user ratings, then give out recommendations to users, which eliminates the need of simulating real-time data ingestion.
**Main tasks.**

- **Data pre-processing.** Apache Spark is used to process the dataset, before used to train the RS model from Apache MLlib.
- **RS training.** Apache MLlib RS model is trained on the pre-processed dataset.
- **RS in use**. Apache MLlib functions are used to recommend anime titles to users.
- **Prediction visualization.** Anime recommendation results are analyzed and visualized using Matplotlib and its supporting libraries.

#### 4.1.3 Reasons for plan changes

After the first phase of development where we had done sections of the project, it is decided that we transferred real-time to static data processing and shifted from applying recommendations for new users to existing users, after discovering that the ALS RS model in Apache Spark MLlib does not allow re-training and updating model like neural networks usually do, and we had yet to find a way to implement the support of other ML libraries (such as Keras) into Apache Spark.

### 4.2 Contributions

This section shall summarize members' impact on the project. Full details on task assignment shall be described in the **Assignments** section.

**Châu Tấn Kiệt (21127329): 40%.**

- ALS model implementation from MLlib.
- Prediction visualization and examination.
- Recommender neural network implementation from Keras. [Canceled]

**Nguyễn Thế Thiện (21127170): 30%.**

- Data pre-processing on Apache Spark based on ALS model.
- Project report and documentation: Abstract + Problem + Evaluation.
- Data pre-processing on Apache Spark based on recommender neural network from Keras. [Canceled]
- Real-time data ingestion to Redis database. [Canceled]

**Nguyễn Trần Trung Kiên (21127327): 30%.**

- Project report and documentation: Application + Group work + Conclusion, citations and references, formats.
- Database configuration and setup with Redis. [Canceled]
- Real-time data streaming from Redis to Spark Streaming. [Canceled]
- Model Evaluation

### 4.3 Assignments

The assignments listed for each member below are in the chronological order.

**Châu Tấn Kiệt (21127329)**

| Task | Components | Expected results | Actual results | Directory |
|---|---|---|---|---|
| RS model | 1. Set up a RS model.<br>2. Explanation on why choosing.<br>3. Test training on small scale. | 1. RS model setup successfully.<br>2. RS model suitable to the problem.<br>3. RS model is able to train on a small set of unprocessed data. | 1. ALS model setup successfully.<br>2. ALS model does NOT suit real-time processing.<br>3. RS model was not yet to be tested at the time of this. | /src/ALS_model.ipynb |
| Spark + ML | 1. Test RS model on pre-processed data. | 1. RS model is able to train on pre-processed data received from Thiện. | 1. Pre-processed data suits neural network and other types of RS models, not Matrix Factorization.<br>2. The RS model was trained on the rating dataset instead. | /src/ALS_model.ipynb |
| RS model: Alternative solution | 1. Set up a RS model based on neural network. | 1. RS model setup successfully.<br>2. RS model is compatible with Apache Spark for large-scale dataset. | 1. RS model setup successfully: Keras recommender neural network.<br>2. RS model is able to run on Numpy-converted data, NOT Apache Spark.<br>3. The solution is later dropped due to insufficient time, despite attempt to find a solution. | /src/legacy/Keras_model.ipynb |
| ALS model in use | 1. ALS model training on full pre-processed dataset.<br>2. Examine recommendation results for users. | 1. ALS model training successfully.<br>2. Set examples of existing users and prediction results, examine the results in response to their rating history. | 1. ALS model training successfully.<br>2. Set examples of existing users and prediction results for 2 users, examine the results in response to their rating history. | /src/ALS_model.ipynb |

## Nguyễn Thế Thiện (21127170)

| Task | Components | Expected results | Actual results | Directory |
|---|---|---|---|---|
| Data pre-processing | 1. Read dataset in the correct format.<br>2. Data cleaning: missing values, duplicates, noise.<br>3. Data integration: unified dataset.<br>4. Data transformation: Value normalization, analysis simplification.<br>5. Data reduction: Dimensionality reduction, data compression. | 1. Data pre-processing to be done thoroughly & logically.<br>2. Step-by-step analysis and documentation. | 1. Data pre-processing to be done thoroughly & logically.<br>2. Step-by-step analysis and documentation.<br>3. Data pre-processing does not match with the algorithm in use: ALS. | /src/data_preprocessing.ipynb |
| RS deploy: User embedding | 1. Save embedding information from the pre-processing.<br>2. RS deployment script that does user input, user embedding, prediction (recommendation). | 1. Easy re-embedding for user inputs.<br>2. A basic UI deployment script to solve the recommendation problem. | 1. Re-embedding process is a mess without using the built-in data processing pipeline.<br>2. The deployment script is incomplete. Not done: Prediction, due to the corresponding model canceled later in the project. | /src/legacy/UserEmbed |
| Report documentation | 1. Report the project in LaTeX format. | 1. A cleanly presented LaTeX document to summarize the project. | 1. A cleanly presented LaTeX document to summarize the project. | /report.pdf |

## Nguyễn Trần Trung Kiên (21127327)

| Task | Components | Expected results | Actual results | Directory |
|---|---|---|---|---|
| Redis database | 1. Ingest data files to database.<br>2. Stream to Apache Spark Streaming.<br>3. Simulate real-time data ingestion. | 1. Data fully ingested to database in key-value format.<br>2. Stream to Apache Spark Streaming without error at local.<br>3. Real-time data ingestion only streaming not processed keys. | 1. Data fully ingested to database in key-value format.<br>2. Stream to Apache Spark Streaming without error at local.<br>3. Real-time data ingestion only streaming not processed keys. | /src/legacy/Redis |
| SVD vs. ALS | 1. SVD application on the dataset.<br>2. Comparison between SVD and ALS. | 1. SVD application.<br>2. Explain the differences between SVD & ALS in how they work & their results.<br>3. Explain the metrics used in the comparison. | 1. SVD application.<br>2. Basic RMSE comparison.<br>3. No explanation on the metrics (how they work, usage meaning) in the comparison. | /src/Compare_ALS_SVD.ipynb |
| ALS: scalability experiment | 1. Experiments to see how the accuracy goes as the dataset size goes up. | 1. Explain the metrics used in grading the property.<br>2. Baseline numbers to tell the property.<br>3. Explain how much and direction of effects as sample size changes to the model. | 1. Plain RMSE calculations for different sample sizes. | /src/RMSE_by_datasize.ipynb |
| Report documentation | 1. Report the project in LaTeX format for: Abstract + Problem + Evaluation. | 1. A cleanly presented LaTeX document to summarize the project. | 1. A cleanly presented LaTeX document to summarize the project. | /report.pdf |

# 5 Self-evaluation

## 5.1 Expectations vs. Reality

As written in the project proposal, our first expectation was to create a model fitted for real-time recommendation system. But as times came, not only did we struggle to implement the right tools and mechanics to work with (see section **Group work/Plans**), we also had to review and change our original problem to fit our current situations, hence some of the assigned work is marked "Canceled".

## 5.2 Plans

As expected from having to rework our project multiple times, the work output is unsurprisingly poor compared to the time we were given. Here is our own judgments for the project by each section:

1. **Data pre-processing**. This section took us the most time, for a number of reasons:

   (a) **Poor work distribution**. As the project first started, only one person (Thiện) was assigned to this position instead of two, because we thought setup and configurating the other tools such as Redis database, ML models would take longer.

   (b) **Lack of experience.** We were used to working on pre-processing smaller datasets throughout the courses of Machine Learning. It took us by surprise at how costly it is to work around a far larger dataset, as well as the lack of tools and online community help.

   (c) **Poor planning**. In addition to point b, we had poor planning, or rather no planning at all, on how to dissect the dataset carefully, as well as miscommunications leading to mismatches between the outputs of this phase and ML model inputs, taking us a lot more additional time to revise our problem and solution.

2. **Model usages.** This section did not take us much time, however had some obstructions on our way:

   (a) **Lack of support.** Apache MLlib RS only gave us ALS, so we had to look for the other ways if we wanted to implement the use of other RS models. One we did test was a RS neural network from Keras library (/src/legacy/Keras_model.ipynb), which did not go well as the dataset was too huge. With such a large dataset, it took too much time for each experiment as well as we tried to find and test suitable methods. Finally, after comparing with SVD model, we decided to choose ALS for this project.

   (b) **Library documentation.** Apache MLlib RS does have an informative amount of documentation of the applied methods in their library. However, it was up to us to not forget spending time pursuing deeper researches and conduct experimental tests on their properties.

3. **Insights.** This section was at the very end of our project, and due to the other sections taking so much time, we had little time on this one. Still, there are things we would want to address:

(a) **Reasoning**. As our mentor has already suggested, and not limited to this section alone, the foremost thing we lack is the reasoning behind our simplest choices, as it is not our usual logicalizing process, but also lack the documentation of our reasonings to back up on how much we tried, failed and succeeded. That is a valuable life lesson on its own.

(b) **Audience.** A common practice (and problem) for college students, is how they usually have to present their work towards the audience consisting of their peers and teachers, so it is expected that the presentation of our work tends to be hard to understand while also skipping some of the most important underlying meanings of simpler, more basic things.

## 5.3 Grading

We shall self-evaluate our work based on our original proposal and final results. The most important note is that a lot of our members did have the effort, despite producing not much work in the ending results while a lot were scrapped. So we shall have two metrics to grade on: the final results and executions on how much time it took compared to our schedules.

So we would consider grading the sections as follows:

1. **Data pre-processing**. 8/10
   (a) 9/10 for the final results: The work is very detailed, but there are reasoning holes and rooms for better techniques to be used.
   (b) 6/10 for the executions: The work took very long compared to the schedule.
2. **Model usages.** 7/10
   (a) 6/10 for the final results: The work is on par, though did not consider multiple models nor explain why choosing.
   (b) 8/10 for the executions: The work did not take much time but had to wait for data pre-processing, so the worker did quite well in the remaining time.
3. **Insights.** 8/10
   (a) 8/10 for the final results: The work is quite of low quality due to lack of metrics explanation and meanings of presented data, as well as excessive and less meaningful information for the client. But at least, we think we did the necessity required for the original problem and addressed it quite neatly.
   (b) 8/10 for the executions: The work had to be done in a very short time (approximately 2 days), but at least did quite a lot.

In total, with how inexperienced we were and the resources we were given, we think a 7/10 is a fairly deserved score range for our work.

## References

[1] Yifan Hu, Yehuda Koren, and Chris Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining (ICDM)*, pages 263–272. IEEE, 2008.

[2] Chao Huang, Lianghao Xia, Yong Xu, Jiajun Liu, Yu Zhang, and Defu Lian. Vhd: Vector heat diffusion for scalable and high-quality graph-based recommendation. In *Proceedings of the 27th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 348–356. ACM, 2021.

[3] Lianghao Xia, Chao Huang, Jiao Shi, and Yong Xu. Graph-less collaborative filtering. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD)*. ACM, 2022.