**CCT College Dublin**

**Assessment Cover Page**

| **Module Title:** | Problem Solving for Industry |
| --- | --- |
| **Assessment Title:** | Capstone Pair Project |
| **Lecturer Name:** | Dr. Muhammad Iqbal |
| **Student Full Name:** | Ingrid Menezes Castro  Robert Szlufik |
| **Student Number:** | 2020341  2020358 |
| **Assessment Due Date:** | 17/05/2024 |
| **Date of Submission:** | 17/05/2024 |

**Declaration**

| By submitting this assessment, I confirm that I have read the CCT policy on Academic Misconduct and understand the implications of submitting work that is not my own or does not appropriately reference material taken from a third party or other source. I declare it to be my own work and that all material from third parties has been appropriately referenced. I further confirm that this work has not previously been submitted for assessment by myself or someone else in CCT College Dublin or any other higher education institution. |
| --- |

# Abstract

# Objectives

— We can rearrange those and number them as well

To develop a system that provides accurate movie recommendations for users, with minimal time needed to produce them

# Research Questions

# Business Understanding

We believe that a system like this could be useful to cover a gap in the industry, since all recommendation systems are embedded within a streaming service. Because of its platform independence, many users that are not attached to specific companies might feel drawn to use it. Below we will use the business analysis canvas as a model to go through those points.

## Project Objectives

The aim of this project is to have a good working model that gives customers recommendations of movies they should and should not watch according to their taste. With the successful implementation of this as an independent platform, we would then move on to partnerships with studios and streaming services in a way to make it profitable, but with the lifetime compromise to keep the system unbiased and free of external interference.

## Stakeholders

* General public looking for a movie to watch;
* Companies that might want to integrate it as an additional feature on their website (eg. it could be featured in movie tracker apps such as TVShow or in streaming trackers like JustWatch).

## Deliverables

A model that is able to recommend movies to users based on their taste and scenarios given; A basic user interface; Supporting documentation;

## Impact to Target Operating Model

Since this is the first project of this sort, no impact to previous legacy systems will be made.

## Communication Approach

This product can be communicated through many portals so, in order of relevance, these would be the marketing approaches used:

* Social media: Instagram, TikTok, etc.;
* Content Marketing: blogs, podcasts;
* Influencers marketing: YouTubers and TikTokers that market for the movie communities;
* Partnerships with movie studios and Paid Advertisement in niche websites.

## Responsibilities

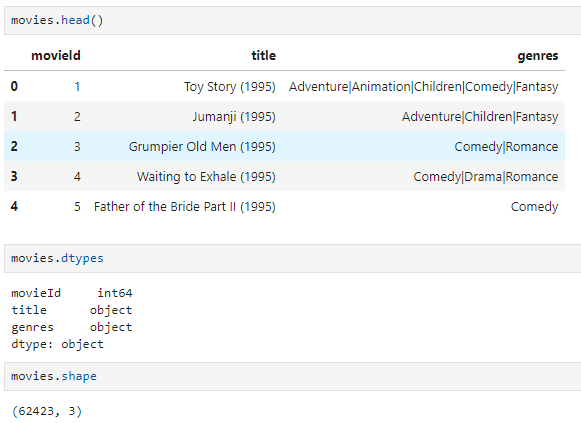
Our team is formed by Ingrid Castro and Robert Szlufik and both have equal responsibilities with the support and development of this project. They count with the technical supervision of Dr. Muhammad Iqbal and the business support of Professor Ken Healy.

## Scheduling

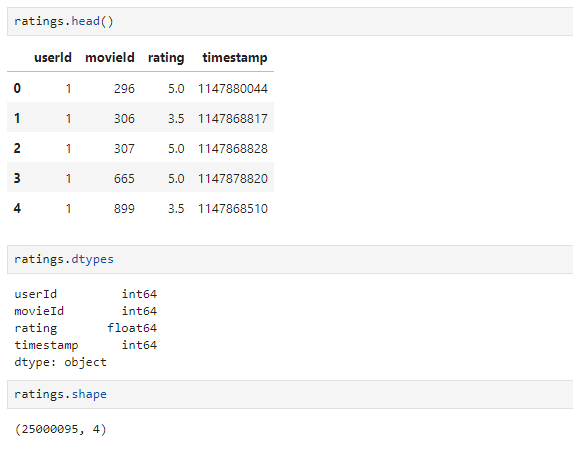
From the business analysis to the deployment of the project alongside its documentation, the team has 2 (two) months to release a working model following the timeline.

# Data Understanding

For this project we mainly used two datasets from Movie Lens 25M (Movie Lens, 2019): Movies and Ratings, both on the CSV format. In terms of shape, Movies contains 62.423 rows and 3 columns originally, while ratings has 25.000.095 rows and 4 columns, hence this dataset has collected over 25M ratings from users all over the internet.



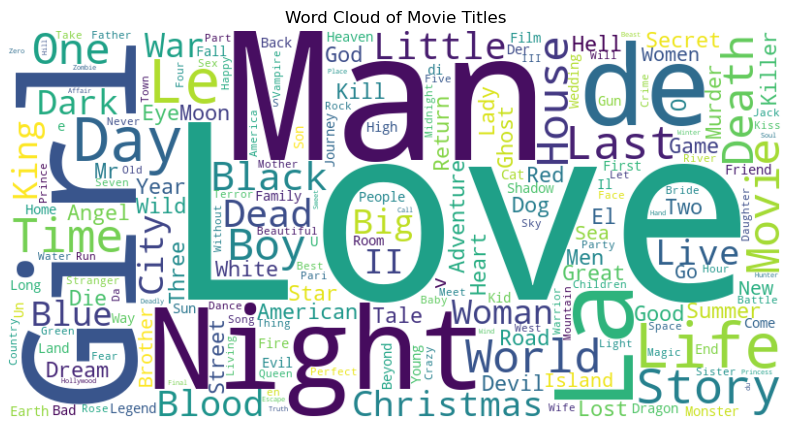
(img: Movies dataset basic stats)



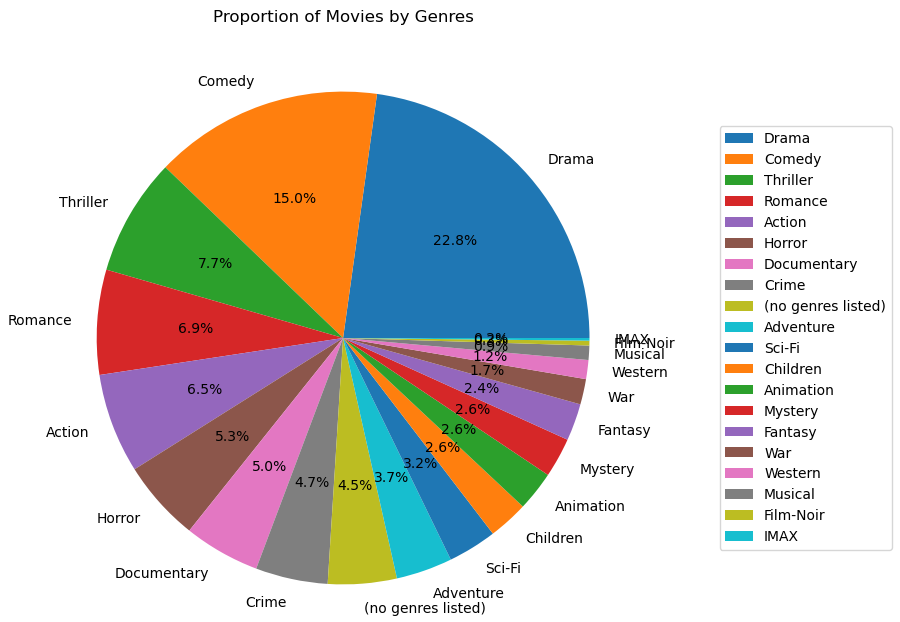
(img: Ratings dataset basic stats)

In terms of data quality, we had no duplicates or null values in both of the datasets, so no data treating will be necessary for missing values or duplicates in the next phase.

As the data still need to be prepped for more on detail EDA, we generated two visualisations of the Movies dataset:

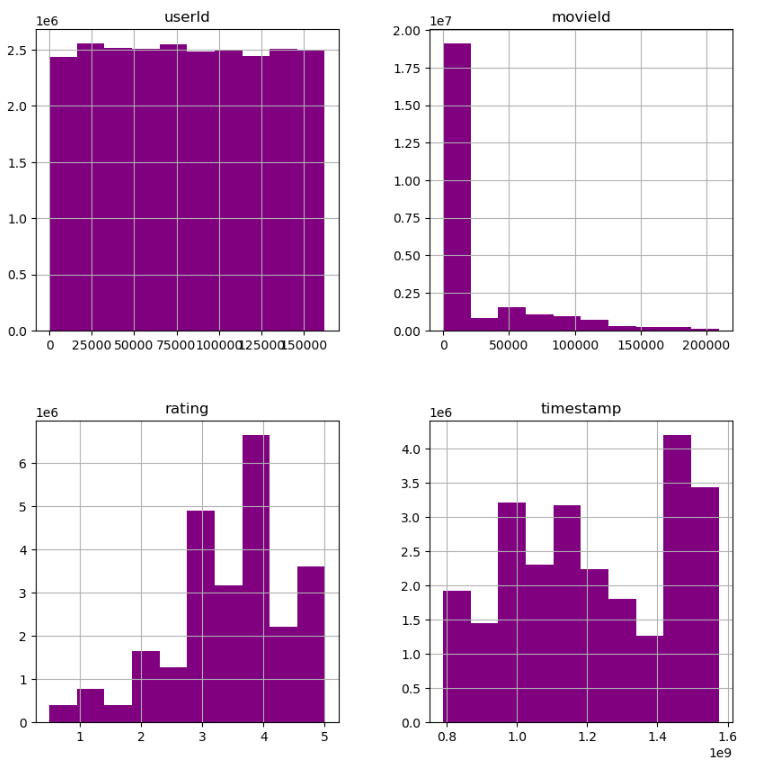


(img: Word Cloud of movie titles)

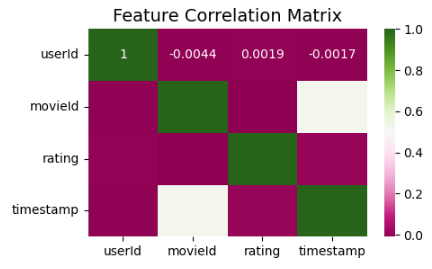


(img: Pie Chart of movie genres)

For the Ratings dataset we opted for showing the histogram and the feature correlation matrix:



(img: ratings histogram)



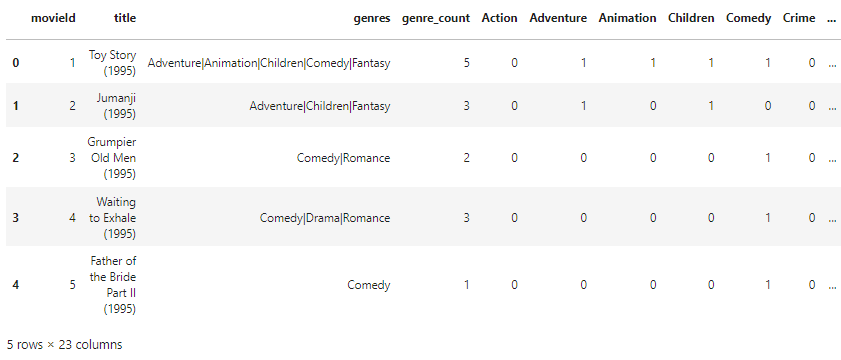
(img: ratings feature correlation matrix)

# Data Preparation

## Movies Dataset

For the data preparation phase we did some alterations on Movies Dataset. Since there were no missing values or duplicates, we started by doing dummy encoding on the ‘genres’ column, taking it from categorical to numerical. Some processes were done:

* Slicing of genres,
* Creation of genre\_count (a column that counts how many genres a movie has),
* Moving of genre\_count to the front of the new section.



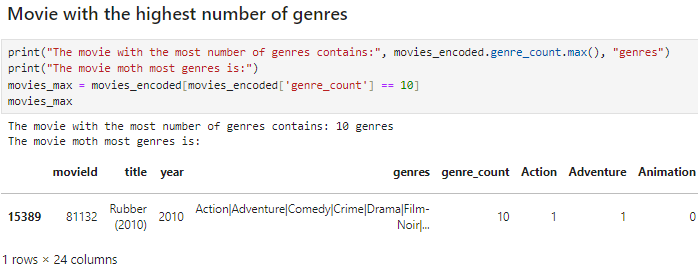
(img: Movies dataset with dummy encoding and inclusion of genre\_count)

We also included the ‘year’ column, another numerical value that was created by extracting the year contained in the ‘title’ column between parenthesis and adding just after the title.

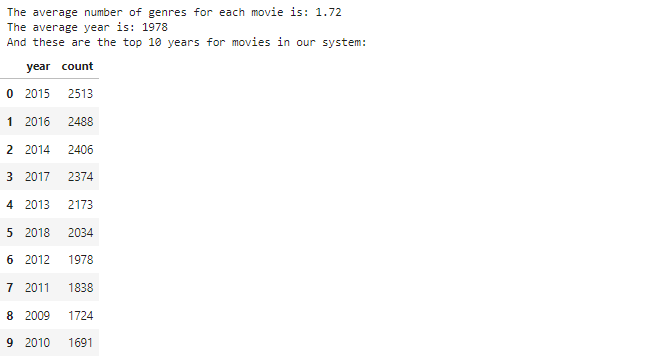


(img: inclusion of the ‘year’ column).

With that we were able to do some further data understanding:

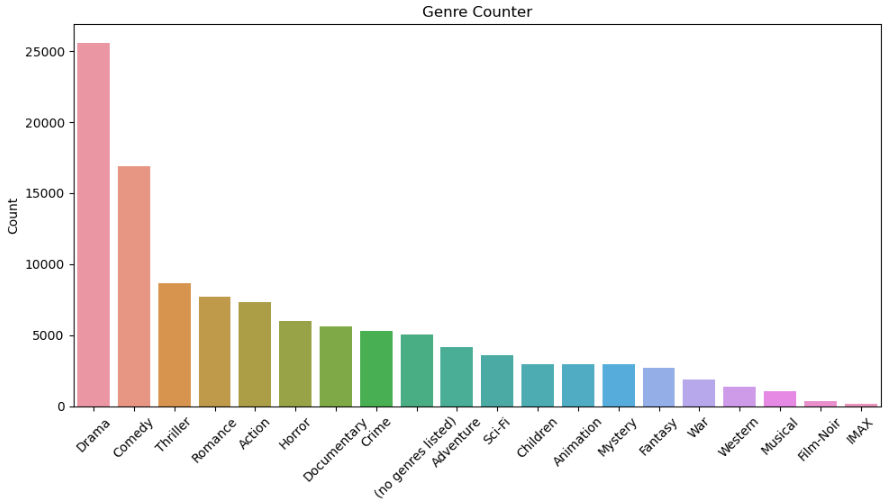


(img: Rubber (2010) is the movie with the highest number of genres)



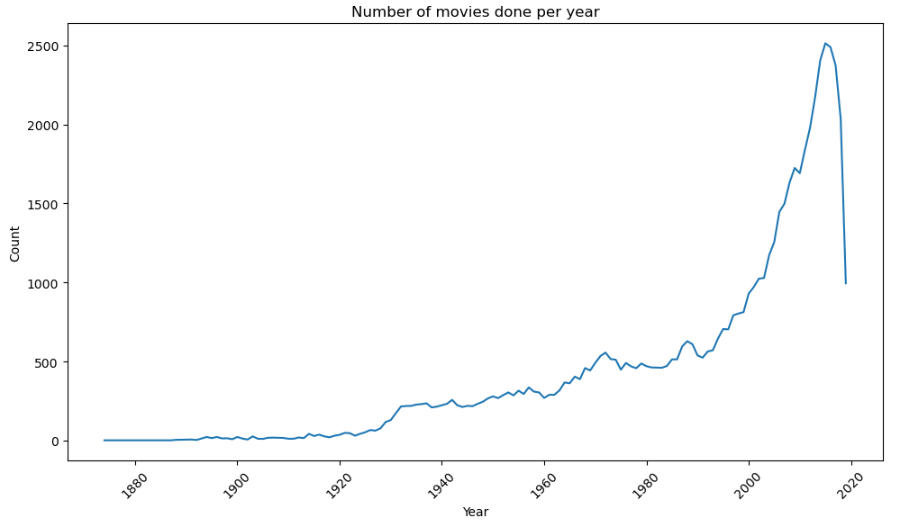
(img: Average number of genres, average year and top 10 years in number of movies)

In terms of genres, this genre counter graph tells us the distribution of genres across movies\_encoded dataset:



(img: Genre counter)

And the number of movies done per year (the dataset comprises of movies up to 2019):



(img: Number of Movies by year)

## 

## Ratings Dataset

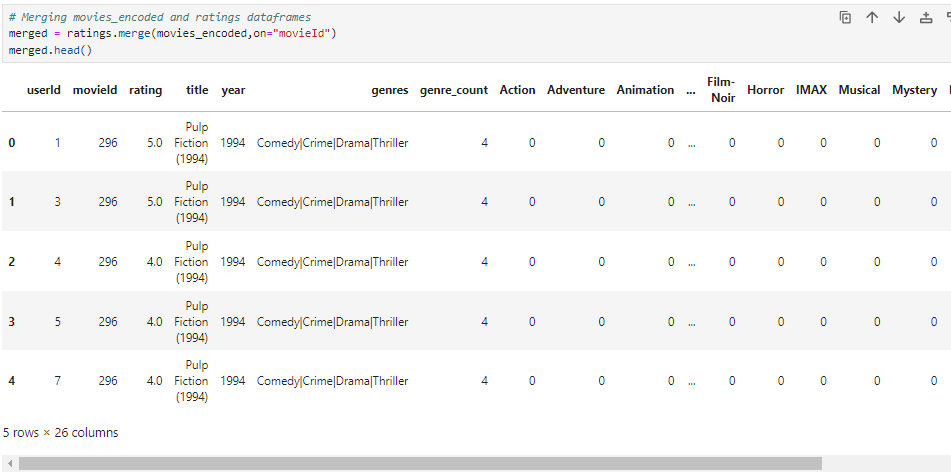
In terms of data preparation of the ratings dataset, again, no missing or duplicated values were found, so we just shaped it a bit differently to prepare it for merging. We chose to drop the ‘timestamp’ column due to it not being useful for what we would use the dataset for.



(img: ‘timestamp’ drop command for data prepping)

## Merging datasets

The last major data preparation we did before moving on for modelling preparation, was the merging of ‘movies\_encoded’ and ‘ratings’ datasets:



(img: merge of movies\_encoded and ratings)

As a last preparation step, we dropped the column ‘genres’ because that was not necessary moving forward and it was redundant information about our datasets. This new dataset that we will be working from now on was named ‘merged’ and has the following shape:



(img: 25M lines and 25 columns)

We then checked again if there were any missing/ duplicated or NA values after the merge, but there were no alterations so we moved on for modelling preparation.

## Modelling Data Preparation

# Modelling

### Overview

In the fourth phase of the crisp-dm framework, we will select a model, and train it with our data.

The very first step in the modelling phase for our recommendation system is to choose the type of the system.

There are 4 main categories of recommendation systems to choose from (NVIDIA, 2024), however, some resources might indicate there are more. This is due to the fact that some systems branch out, and become very specific. For our purposes, we can define 4 main categories:

* Collaborative filtering
* Content filtering
* Context filtering
* Hybrid models

In essence, collaborative filtering aims to find the most similar users/customers to target users and recommend based on that association. For example, we could find several similar users to our target user, and recommend them items based on rating/score provided by said similar users.

Content filtering is based on filtering and recommending items or products that our target user has interacted with. A good example might be YouTube videos recommended based on our search.

Context filtering is a method used by streaming services providers, such as Netflix. It aims to recommend based on attributes such as date, time and country of target user.

Hybrid models use multiple methods and techniques in combination, aiming to improve outcome or lower the inaccuracy.

### Alternative approach

The objective of this project is to provide users with the most accurate recommendations within a small time limit.

Through the modelling phase, we tried several different approaches, such as collaborative filtering and several machine learning models. We found that some models are very accurate but slow to train - SVD, and others that train very quickly but result in low scores.

The proposed solution is to train a large, accurate model, which provides high accuracy, but recommend movies based on predictions made for most similar users to our target user.

In other words, we will train a large model in advance, and when a user requests a recommendation, the system will ask them to rate up to 10 movies. Then, it will find the most similar users, and rate movies based on the trained model. At the end, the target user will be presented with recommendations based on the average predicted rating for the most similar users.

This approach maximises prediction accuracy, and minimises time constraint.

### SVD Algorithm

During our investigation and research into recommendation systems, we came across a python package that was created and optimised especially for recommendation systems. Upon further investigation, this package implemented a very famous algorithm called Singular Value Decomposition (SVD).

SVD is a dimensionality reduction algorithm, similar to PCA, which aims to obtain a single value from the user item matrix. It is a matrix-factorization method introduced by the BellKor’s Pragmatic Chaos team, which have won the 2009 Netflix $1,000,000 award. The competition aimed at improving Netflix's recommendation system by a substantial amount of 10%, as measured by root mean error squared (RMES). (NJIT, 2020)

There are many steps involved in implementing this algorithm exactly as presented by Bell and his team (Bell et al., 2008). It involved calculating user and item biases and calculating general error for each. Then, they iteratively adjust scores and biases, finally merging them together. When making a prediction, estimated score is calculated by adjusting obtained item and user biases.

We found a very well performing implementation of this algorithm, included in one of the packages developed for python. Package “Surprise”, developed by Nicolas Hug (Hug, 2015b), implements SVD proposed by Bell and his team. However, this package is compatible with older versions of python, and additionally comes with several different classes and algorithms. For this reason, we obtained source code for this algorithm and changed it slightly to match our needs. Source code developed by Nicolas Hug (Hug, 2015a).

### Sklearn Algorithms

Before we decide on the particular algorithm we use, we should test them initially and compare them to one another. From the “Scikit-learn” package we selected 4 algorithms for comparison, Linear Discriminant Analysis (LDA), Decision Tree Regressor (DT), Random Forest Classifier (RF) and Gaussian Classifier(NB).

In the next step, we took a dataset with dummy-encoded genres, selected independent and dependent variables, and took a 100,000 sample of that dataset.

Next, we defined our algorithms, and performed cross-validation tests for each of them. Figure 1 presents results in terms of RMSE.

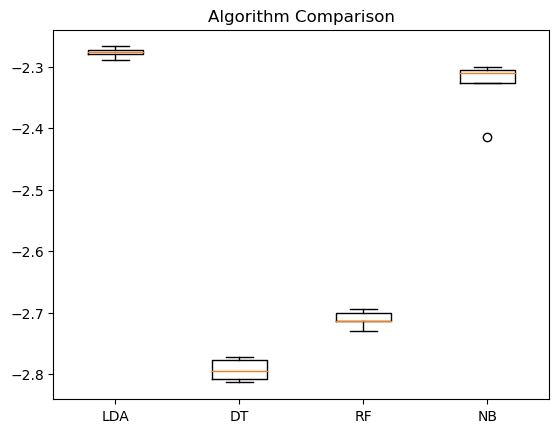


Figure 1 - performance comparison between Sklearn algorithms.

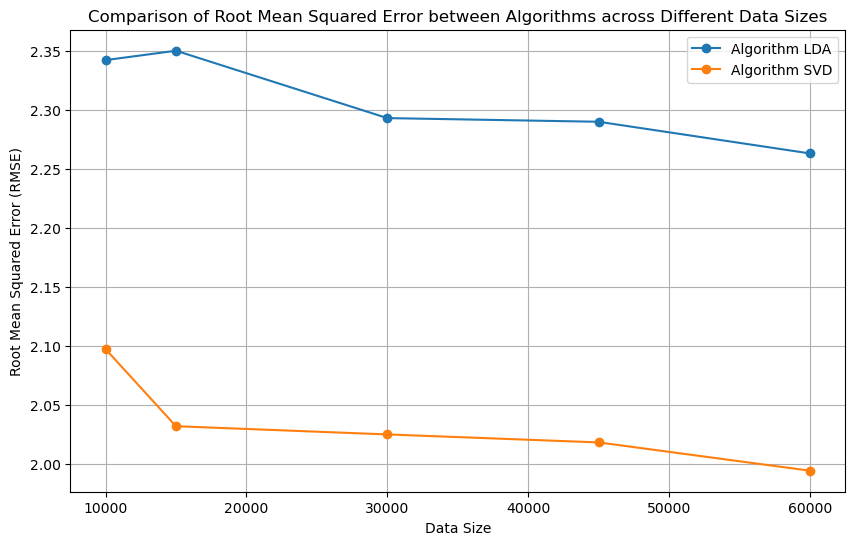
As we can observe from the figure above, the LDA algorithm performed significantly better than the rest. For that reason, we will use it for further comparison.

In the next step, we performed another grid-search on LDA, in order to tune hyperparameters.

### Comparing SVD and LDA algorithms

With SVD and LDA implemented, we can proceed to comparing two of them. Simultaneously, comparing their RMSE score and how they improve when more data is introduced.

In the following comparison, we took 5 dataset samples of sizes 10,000, 15,000, 30,000, 35,000, 60,000, respectively, and the results are presented in figure 2.

Figure 2 - RMSE score comparison between LDA and SVD algorithms.

As presented in figure 2. SVD performed significantly better than LDA, with indication that it will improve when more data is introduced.

### 

### Testing SVD’s performance with large datasets.

For the final round of modelling and testing, we selected much larger data samples of 100,000, 150,000, 200,000, 250,000, 300,000 rows. Results of the test are presented in Figure 3.

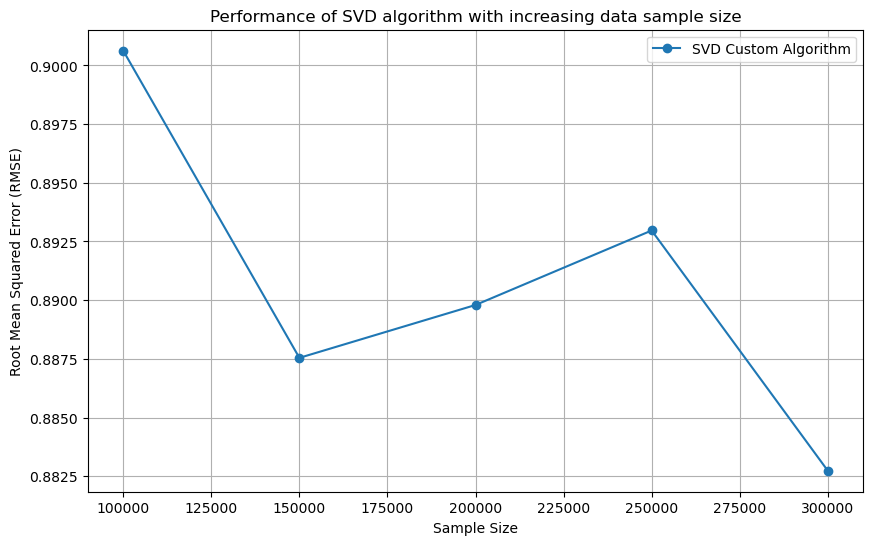


Figure 3 - performance of SVD with increased data sample size.

As presented in figure 3, we can observe improvement in algorithm performance with increased datasize. The error increases between the 2nd and the 4th sample, which might indicate data bias. Perhaps, the last bit of data introduced between sample 2 and 4 contains information about users that is not sufficient, therefore, user and item biases cannot be accurately adjusted. We can observe that with the last sample, algorithm performance improves again.

### Training final model.

# Evaluation

# Deployment

References

https://www.mybib.com/j/Old-fashionedStormyPorcupine