**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

# Project Objectives

1. To develop a system that provides accurate movie recommendations for users, with minimal time needed to produce them.
2. Create a working prototype for our recommendation system.

# Research Questions

1. Which machine learning algorithms are used in recommendation systems and which one will work the best in the context of our system.

# Stage 1 - 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.

### Technologies

#### Proprietary Machine Learning and AI systems

* Vertex AI from Google (Google, 2024) - provides option for recommendation for media content
* Azure AI from Microsoft (Microsoft, 2024) - platform for ML and AI
* AWS ML and AI (AWS, 2024) - platform for ML and AI

Proprietary ML and AI systems are developed by excellent professionals and tend to have the highest quality among the offered resources, constant updates/improvements and plenty of support online, which would help a lot in crucial parts of this project.

However, using proprietary solutions forces us to adhere to frameworks chosen by their creators, which can impact our overall process of development. We are forced to work with “black box” solutions and frequently are unable to tell what processes or algorithms are in place.

#### Open Source systems/solutions

There are multiple solutions available for developers which hold open source licence. Solutions are created in multiple programming languages such as C++, Java and Python, with the last one being the most popular among developers and our chosen language.

The most fundamental libraries in Python for ML/AI projects are:

* NumPy
* Scikit-learn
* Py-Torch
* TensorFlow
* Pandas

Open source software presents us with opportunities to model our codebase in the way we decide, empowering innovation.

On the other hand, using open source software has potential risks associated with it. Versioning of new packages might break our codebase, packages can contain all sorts of vulnerabilities. According to Singh, Bansal e Jha(2015) security and support can be an issue because neither the environment is controlled in Open Source Development nor support is wide and active. Developers using these tools are dependable on a whole community, instead of a team that is ready to help them in case of need.

#### Our choice

We decided to choose Open-Source products, such as Python and packages NumPy, Scikit-learn and Pandas. Those are core packages used in ML/AI projects, especially Scikit-learn, which provides us with several machine learning algorithms. This presents us with the opportunity to choose the most suitable algorithm with the best score.

The most viable alternative to the Scikit-learn package would be using Google AI service, which could produce very good results. However, as mentioned above, we would have to adhere to frameworks and APIs used by Google, which could negatively influence transparency and complexity of the project.

The Proprietary alternative to NumPy and Pandas would be MATLAB, which can be incorporated to the existing Python code base. However, NumPy and Pandas are very popular libraries with excellent abilities. Both are used on both personal and professional level projects. MATLAB comes with detailed documentation, yet so does Pandas and NumPy. We believe that for complexity expected in this project, Pandas and NumPy will be sufficient.

### Legal and Ethical Issues

A Movie Recommendation System when it comes to legal and ethical issues finds some implications that we need to be mindful when involved in this project. At its initial stages of development, when dealing with datasets and the construction of the model, there are no potential legal issues since IMDB or MovieLens, as examples of sources, have obtained their data according to regulations and with all proper privacy consents etc. A discussion was raised though, with the existence of biases in demographics and other factors.

In the book entitled “Recommender Systems: Legal and Ethical Issues'' (Genovesi, Kaesling and Robbins, 2023), the authors highlighted the discrimination and bias in these sort of systems because the data gathered usually falls onto a restricting scope that misrepresent the variety of the people. Another important point of friction mentioned by the authors is: transparency and compliance with GDPR. All those points will be taken into consideration at the stage of development.

Finally, when dealing with users' data, in a way that our system improves and gives better suggestions, we will need to be inline with GDPR and the Digital Services Act (DSA). Ensuring security, transparency and an unbiased recommendation to the user must be a constant concern when developing a project of this kind.

### Data Collection

Initially, data will be obtained from datasets available on IMDB and/or MovieLens, that will give us details on catalogues and user reviews/favourite movies. Then, the model will proceed to collect data from users, which will refine and tune the system.

According to Kanoje, Girase and Mukhopadhyay (2018), user profiling is very important for providing a good web service. When we learn from our users and the model gets fed by their likes and dislikes, then we would have a competitive advantage over other companies that have this system embedded within their limiting catalogue and with no possibility of resetting data if necessary.

# Stage 2 - 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, found on <https://grouplens.org/datasets/movielens/25m/> [1].

In terms of specifics we have:

* ‘Movies’ dataset contains 62.423 rows and 3 columns;
* Columns are: ‘movieId’ (int64), ‘title’ (object) and ‘genres’ (object);
* No duplicate values;
* No null values;
* No NA values;

And:

* 'Ratings' dataset contains 25.000.095 rows and 4 columns;
* Columns are: 'userId' (int64), 'movieId' (int64), 'rating' (float64), 'timestamp' (int64);
* No duplicate values;
* No null values;
* No NA values;

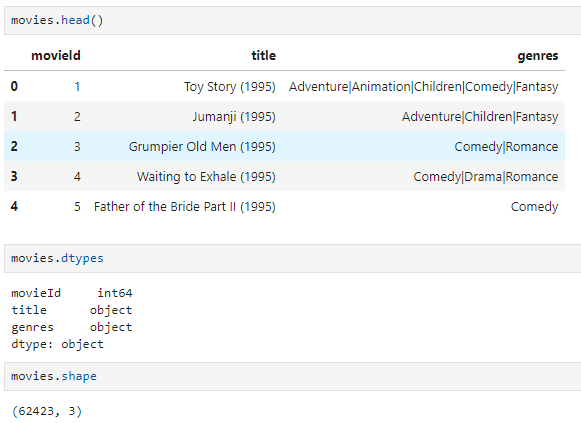


Fig 2.1 - Movies dataset basic stats

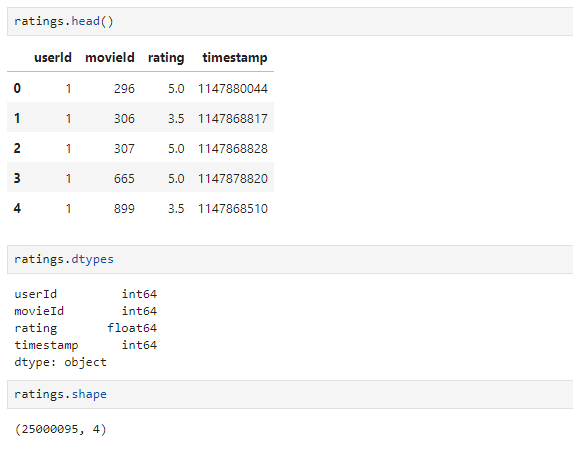


Fig 2.2 - Ratings dataset basic stats

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:

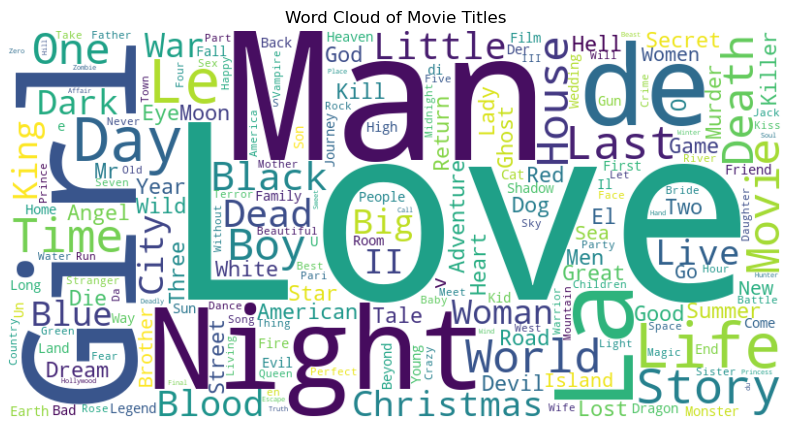


Fig 2.3 - Word Cloud of movie titles

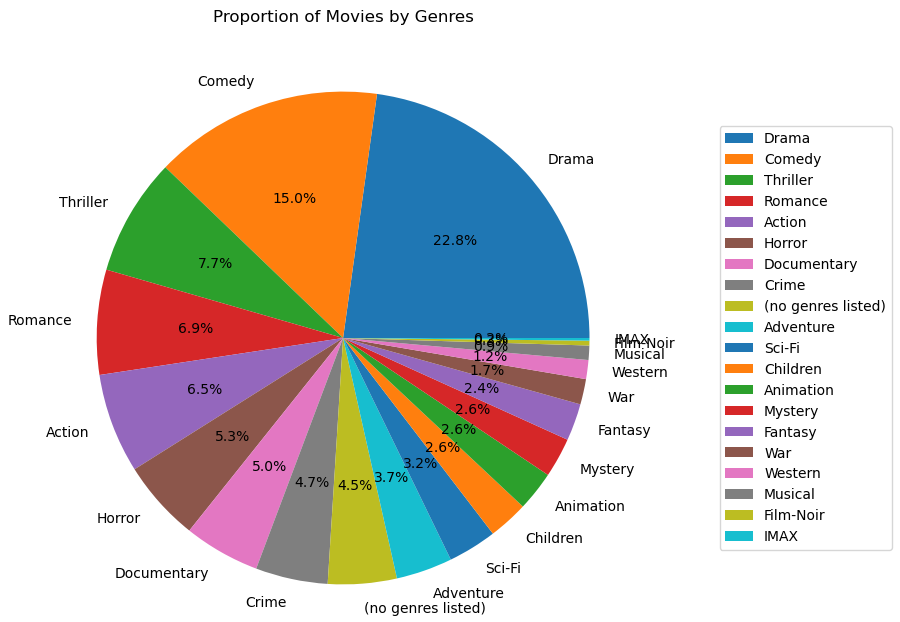


Fig 2.4 - Pie Chart of movie genres

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

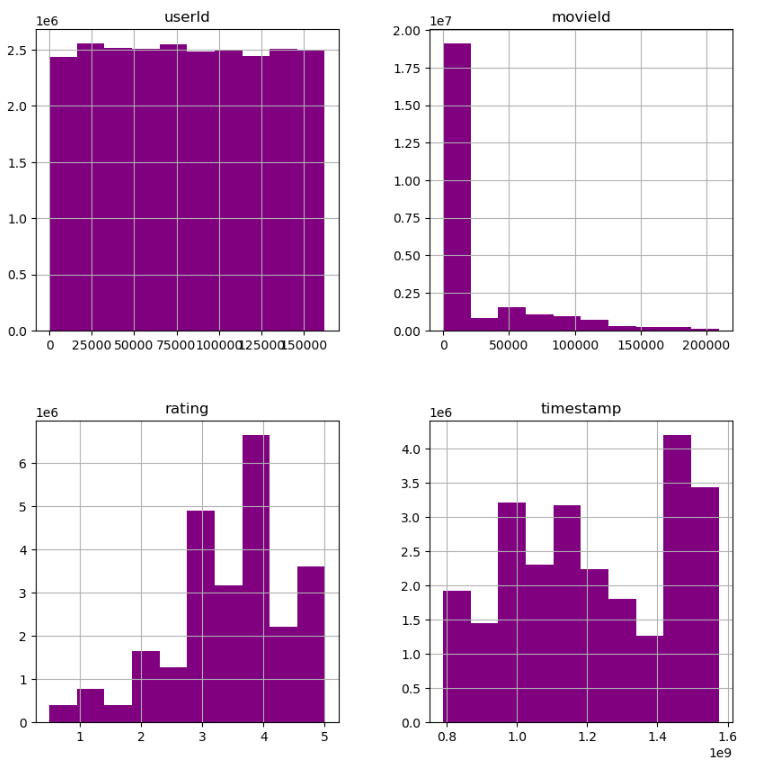


Fig 2.5 - Ratings histogram

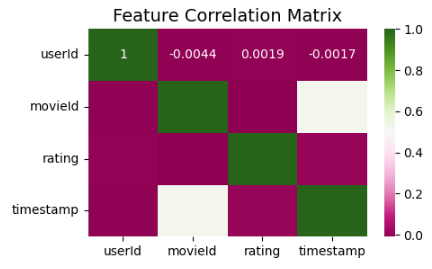


Fig 2.6 - Ratings feature correlation matrix

# Stage 3 - 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.

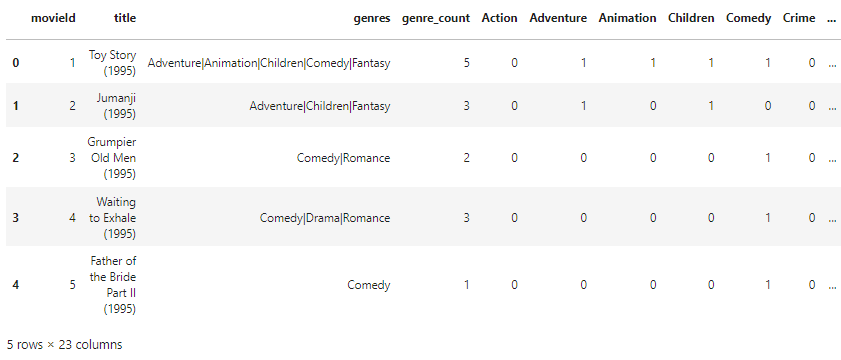


Fig 3.1 - Movies dataset with dummy encoding and inclusion of genre\_count

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.



Fig 3.2 - inclusion of the ‘year’ column.

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

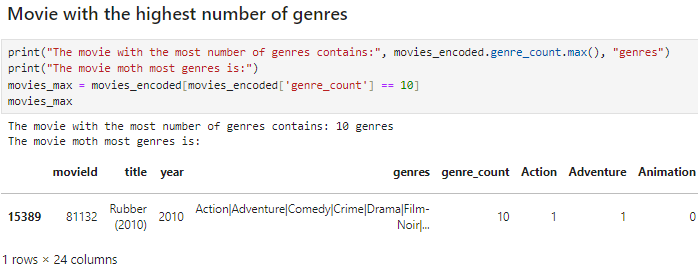


Fig 3.3 - Rubber (2010) is the movie with the highest number of genres

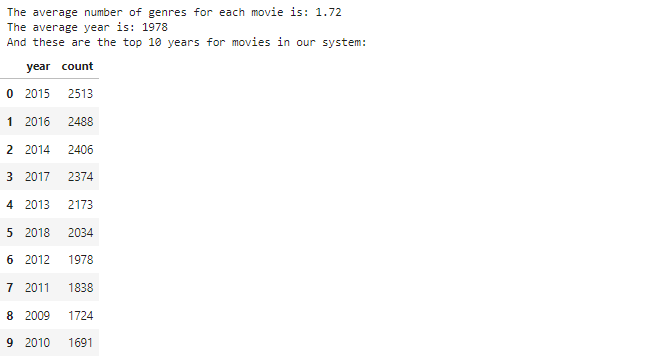


Fig 3.4 - Average number of genres, average year and top 10 years in number of movies

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

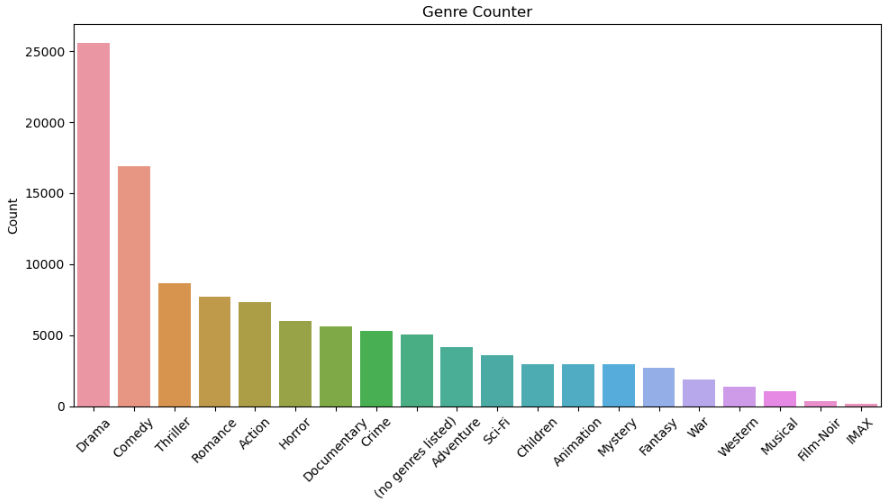


Fig 3.5 - Genre counter

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

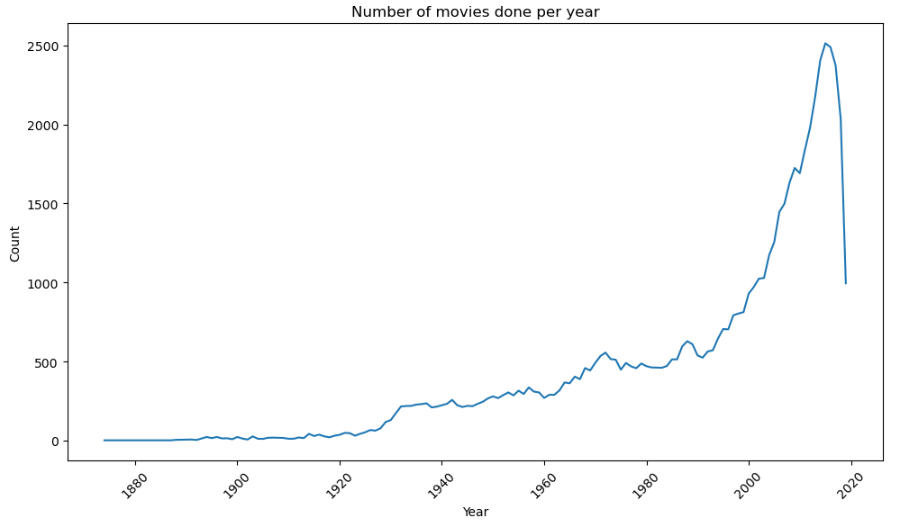


Fig 3.6 - Number of Movies by year

## 

## 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.



Fig 3.7 - ‘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:

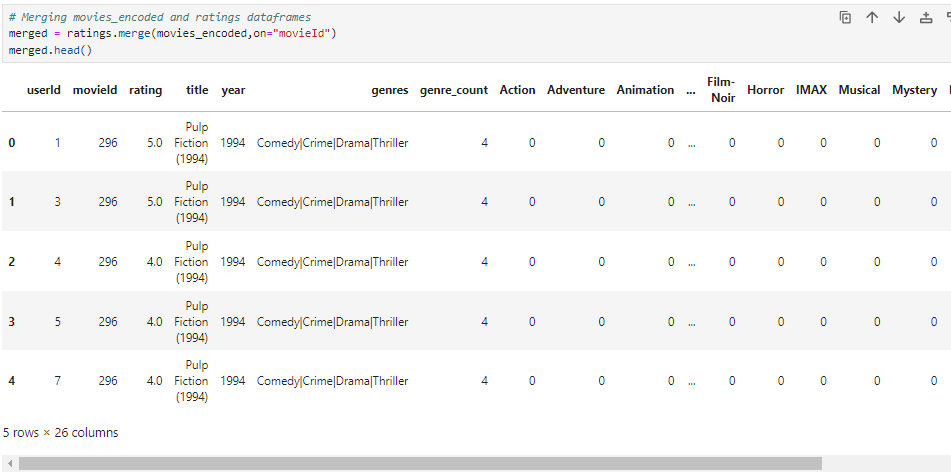


Fig 3.8 - Merge of movies\_encoded and ratings

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:



Fig 3.9 - 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

For the data preparation for modelling we need to do some actions to make data proper for model usage. We did this data prep:

* Sampled the data: we took a sample of 100 thousand rows from the merged dataset.
* Encoded rating values: because the values were float, we encoded the numbers to be integers, not going from 0.5-5, but instead, 1-10.
* Selected columns for independent variable X: columns "rating" and "title" were removed.
* Declared X and Y: We declared the dependent (y) and independent variables (X).
* Selected the models: we selected the algorithms we would use for comparison.
* Split the data: we split the data into testing and training, with test sizing being 30%.
* Scaled data: we scaled the train and test data with the StandardScaler().

To show how we encoded the values from float to integers we appended the print below:

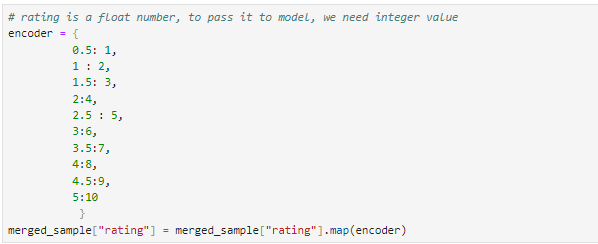


Fig 3.10 - Encoder for values float to integers

Further on we did a similar data preparation for final modelling, slicing the ratings\_final dataset onto dependent and independent variables and

# Stage 4 - 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). According to Portugal, Alencar and Cowan (Portugal, Alencar and Cowan, 2018), those algorithms are amongst the most used in the context of recommendation systems.

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.

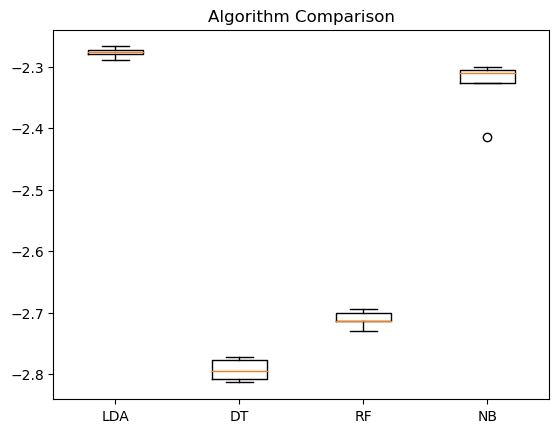


Figure 4.1 - performance comparison between Sklearn algorithms.

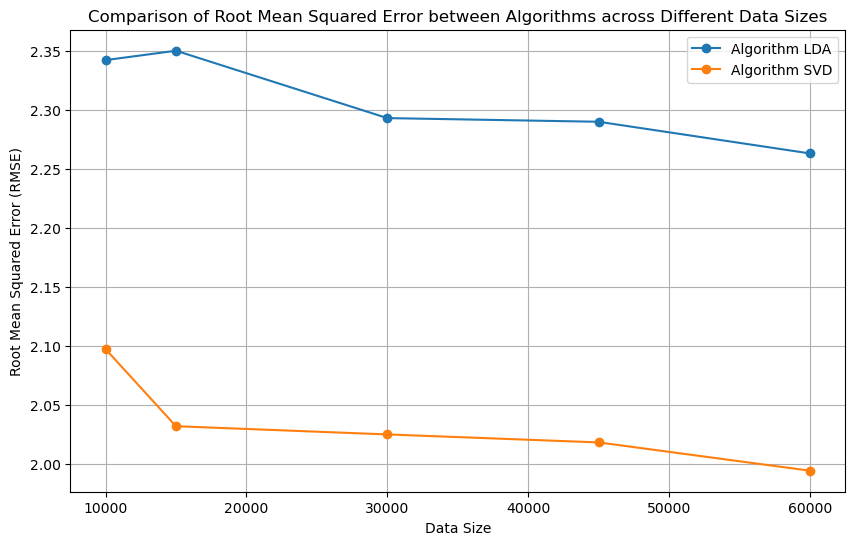
As we can observe from the figure above, the LDA algorithm performed significantly better than the rest. For that reason, we will use it for further comparison.

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 4.2 - RMSE score comparison between LDA and SVD algorithms.

As presented in figure 4.2. SVD performed significantly better than LDA, with indication that it will improve when more data is introduced. It is important to note that the scale of recommendation for those tests is 1-10 as opposed to default 1-5. The reason behind it is that the LDA algorithm does not accept floating point numbers, therefore, they have to be converted to integers.

### 

### 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.

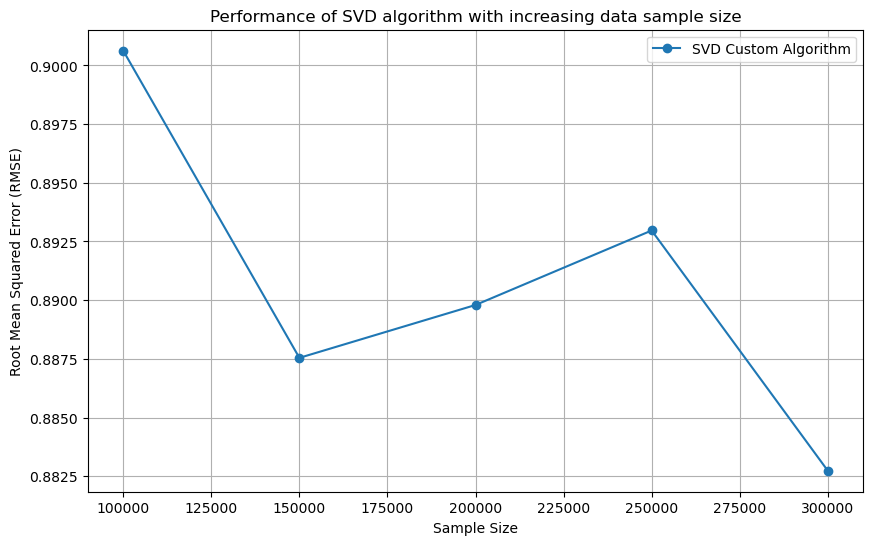


Figure 4.3 - performance of SVD with increased data sample size.

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.

As we can see, the SVD algorithm performs as expected, scoring close to 0.8 RMSE. In the next step, we trained the final model on selected data.

We decided to retain movies that have been rated over 10,000 times, this will help to reflect both movie and user biases accurately. Sufficient amount of information, in this case rating, is necessary to train the model sufficiently, not only from validation and scoring standpoint, but to recommend movies with high degree of confidence.

After selecting movies that have been rated over 10,000 times, our final dataset contains 11,877,943 entries, 162,109 users and 588 movies. We believe that this is a sufficient amount of movies to make recommendations for, however, in principle, the entire dataset could be used to train the final model. This would result in a larger movie pool to recommend from, but could be less accurate and more time consuming. Our best chance for successful prediction is to limit the amount of movies to the most popular.

The rationale behind selecting only the most relevant movie is due to the nature of the final system. The system will try to match the most similar users and average their predicted ratings for all movies that haven't been rated by our target user. If we were to use the entire dataset, the matching user pool for less popular movies would decrease, and thus the similarity score to our target user would decrease.

In the next set, we split the final dataset into independent and dependent variables X, y, and split them further to the training and testing sets. Then, we use X\_train and y\_train sets to train our final model.

# Stage 5 - Evaluation

### Testing final model

After the final model finishes training, we need to check its validity. In order to do that, we need to predict values to X\_test set and compare results to the y\_test set.

Difference between values predicted by the final model for the X\_testing set and y\_testing set will give us a score that our model has achieved. However, it is important to check whether the model is performing well over X\_training set. We will predict values for X\_train set and compare them to actual values in y\_train set as well. If final scores for training and testing set differ a lot, we potentially have a situation in which the model is under fitted or overfitted.

* Final RMSE (Root Mean Squared Error) for estimations over testing set is 0.895
* Final RMSE for estimations over training set is 0.891

As we can observe, both scores are very close to each other. This indicates that data fed into the model is sufficient and the model predicts with high degree of accuracy.

### Final model accuracy in context to the final system

As presented, the final trained model has performed very well and matched expectations. As reported by Bell (Bell et al., 2008), SVD algorithm expected accuracy is close to what we have achieved. To put it into context, movies in our dataset are rated from 0,5 - 5, with 0,5 steps. If movie A is rated 3.0, the final model will predict rating for it, anywhere from 2,2 to 3,8.

If we look at this result from a purely mathematical perspective, it might not be very impressive, however, this algorithm takes the user into account. It predicts within 0.8 mark for a particular user. That gives us confidence in the final prediction.

Moreover, we need to take into account the fact that ratings are highly subjective, and considering other tested algorithms, this one has produced the best results.

Another important factor is that the final prediction consists of an average across several users. This might potentially mitigate inaccuracies in predictions for a single user. Systems that leverage, both collaborative filtering and prediction estimation, might further benefit from the fact that users are biassed toward their preferences. With this hybrid model, we are combining the best from both approaches, we take into account what similar users like, and tilt toward their preferences and simultaneously, try to expand the pool of movies to recommend by predicting rating across the entire final set.

# Stage 6 - Deployment

In the deployment phase, we utilised the final model to predict ratings for each user for all movies in the dataset. The dataset used is the same dataset that the model has been trained on. Thus, the final rating dataset has been created and saved.

There are two part to the final system:

* Dataset with ratings for all users - the final rating dataset
* Method to select similar users

To select similar users, we need our target users to rate a few movies. They can choose any movie that is included in the final rating dataset. Then, we will search a dataset that has been used to train the final model, to select users that have given ratings scores similar to our user. In the next step, we calculate cosine similarity, and select users, lets call them a target group, with the highest score.

Next, we filtered the final rating dataset for users ids that were included in our target group, grouped movies by id and calculated average rating for each movie.

In the final step, we present the user with a dataset containing movies with estimated rating, sorted by highest rating.

As a proof of concept, we included a random user generator. It will select random movie ids from the dataset and assign it a random rating value. Then, we can n proceed and recommend it for this user. Note, that results for random users will differ each time the code runs. - Linked to objective 1.

GUI LINKED TO OBJECTIVE 2

# Conclusions

We believe that the main objective of this project has been achieved. Our hybrid implemented system recommends movies across large selection, leveraging machine learning in consequence demonstrating that such solutions are valid and accurate.

Moreover, we have learnt how recommendation systems work and how they are structured, what data is to be used and how to prepare it, so it is ready to be fed to the model. Then, we explored which model performs the best with our data and implemented it. In the final step of verification, we have achieved a solution that was adequate to our expectations, and addresses project objectives. In the deployment part, we have presented how our system would respond in context to a user, and added an element of randomness, showing that it will adapt and produce outcomes for different users. Additionally, we have implemented a simple GUI prototype to showcase a fully working use case.

In retrospect, there are multiple ways in which such a recommendation system could be implemented, and in fact, there are many such systems available. However, systems that we have seen while researching for this project, only focus on one method of recommending. This is an opportunity which we explored and we are pleased with achieved results.

# References

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