Movie Recommender System

March 2022

1 Abstract

In this day of age where time keeps running shorter and people are bombarded with choices, there is a great need for good recommender engines to keep up with the fast pace of life. This research, which compares different recommender engines, particularly memory and model based ones, gives insight in different approaches to recommender systems and their implications. Even though the paper did not succeed in showing that model based methods would be better, as research suggested, some significant results still came out of the process. With a RMSE *of 0.73 (based on a rating prediction on a scale from 0 to 5), compared to the baseline:* RMSE *of 0.90 (prediction using average rating) the research did succeed in significantly improving the system and definitely gave some food for thought for further research.*

2 Introduction

In this research paper, several machine learning methods were implemented to predict user movie ratings, aiming to make more accurate movie recommendations. A Kaggle dataset with over 20 million user ratings has been used.Generally, for recommender systems, there is a clear distinction between two approaches.

On the one hand there are 'Content Based' methods, which use user/item specific information to give recommendations. On the other hand, there are 'Colaborative Filtering' (CF) methods, which use the inputs of other users to predict ratings. In more sophisticated algorithms, the two groups of methods can be combined in a hybrid method as they did in 'the Netflix Prize' competition. In this paper the choice has fallen on CF methods, due to their many use cases and versatility.

A distinction is specifically made between memory and model based techniques and the main goal of the paper is formulated in the following research question; *"Are model based methods better than memory based ones in the setting of movie recommendation?"* Research shows that it is to be expected that the model based methods should outperform the memory based ones in terms of accuracy and computation time (Aditya et al., 2016) and thus it is hypothesized that the deep learning model will be optimal. The content of the paper delves into the implementation and outcomes of the following algorithms.

1. KNN, which is a memory based method. Both user-based and item-based implementations of the algorithm are examined.
2. Singular Value Decomposition (SVD), a model based method which uses dimensionality reduction.
3. Deep learning using an embedding layer. A deep neural network tries to derive relations between users and movies using their corresponding embedding vectors .

Figure 1 provides a clear overview of the connections between the different models. In the remainder of the introduction, the basics of CF and content based method are elaborated on to establish some grounds for the reader before further proceeding with the paper.

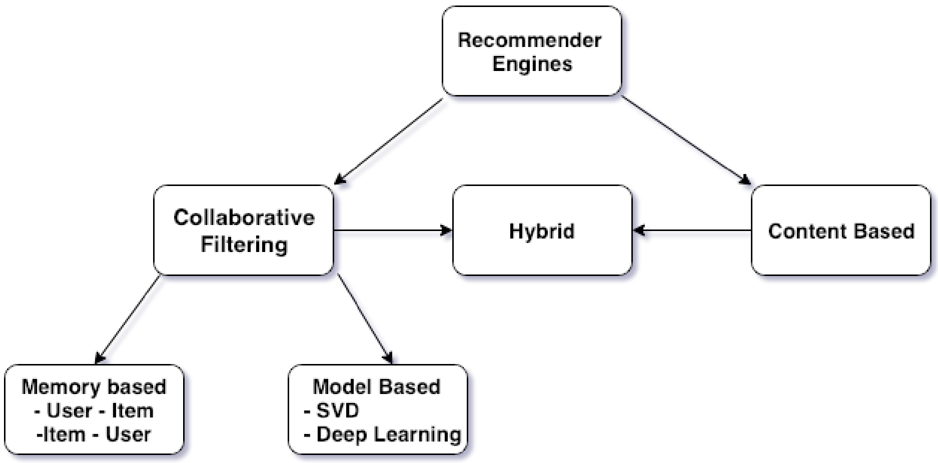


Figure 1: Visualization of the connection between the different model

2.1 Collaborative Filtering

Collaborative Filtering (CF) is a term used to refer to methods able to make item recommendations (e.g. movie recommendations) based on patterns found in user behaviour (e.g. ratings, web page visits) without the need of user/item information (Su, X et al., 2009). CF has many benefits. For instance, the method needs very little information. There is no need for any user or item analysis, which would otherwise be very complex to automatize. CF also has the ability to filter based on subjective concepts, such as taste and quality. (Herlocker, J. et al., 2009). It only requires user input making it easy to generalize. A potential downside of CF is the cold start problem. In the beginning there are no user inputs so the algorithm can't work. At this stage of the process, content based methods can give a solution.

2.2 Content Based

As opposed to CF, Content Based methods do make use of user/item specific information e.g. age, location and gender (for the user) and genre, date of release and language (if the item is a movie). Since it doesn't need any user input it can start from the get go, that's also the reason why hybrid methods are considered the best. Often in these system the cold start problem is solved by using content based methods and along the way collaborative filtering is used to improve its capabilities.

3 Data analysis

The dataset used in this research is the Movielens 20m dataset. This dataset consists of over 20 million different ratings from 138000 different users covering 27000 movies. Users are allowed to give ratings ranging from 0.5 to 5 with step size of 0.5. In the appendix, a figure can be found of the distribution of ratings within the dataset. Clearly the majority is represented by ratings of 4 and 3. Remarkable is the low amount of ratings of 3.5. Upon further analysis it was discovered that, without including the users with less than 20 rating and the movies with less then 1000 rating, 61.5% of the users rated all the movies within a single day and 74.1% of the users gave all their rating within a month. For the second figure in the appendix the difference in days between the users most recent and first given rating was calculated. These values are shown in the frequency plot.

3.1 Data pre-processing

Data preparing started with removing missing ratings; those with a value of 0. Due to computational and technical difficulties only 1 million ratings were taken into consideration. Selecting a lower bound for the amount of movies a user has watched and for the amount of ratings a movie has received will decrease user and movie bias. These lower bounds were set to 20 and 1000, respectively. The dataset that was left consisted of about 370.000 ratings. In figure 2, the density of the average rating of the movies is given. It is shown that most movies have an average rating between 3.5 and 4 and only a small proportion of the movies have an average rating below 3. This can be caused by the small amount of ratings below 3. In figure 3, a density plot is given for the amount of ratings a user has given and in figure 4, the time between the first and last rating given by a user in days.

The differences in the aforementioned models required a dataset that could suit all of them. Furthermore, because all models take user bias into consideration the data does not have to be standardized or normalize.

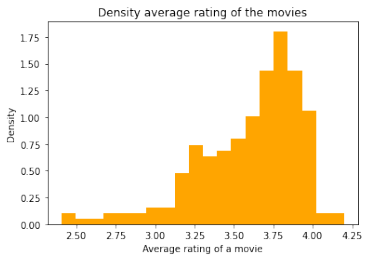


Figure 2: The Density of average ratings of movies

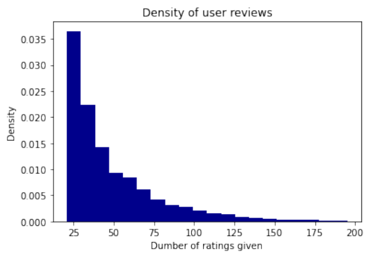


Figure 3: The density of the number of ratings per user

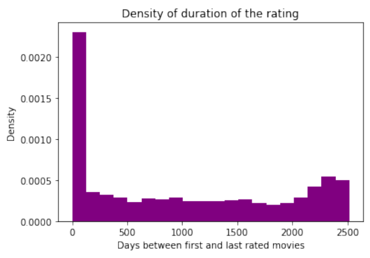


Figure 4: The density of the time between first and last rating

3.2 Splitting data

The previously prepared data needed to be split into a training, validation and test set. During the splitting data leakage and the cold start problem needed to be considered. A cold start arises when a certain user only exists in the test set and does not have any ratings to compare to other users (Vlachos et al. 2018). Data leakage arises when the global timeline is not taken into account. In the paper of Ji et al.(2021) multiple data splitting methods were discussed. The *split-by-timepoint* data split does prevent data leakage but might in turn lead to the cold start problem, as some users will only be present in the test data. Thus, the *Leave-one-out-split* split was implemented. This split places the most recent rating a user gave is in the test set, the second most recent rating a user gave in the validation set and places the remainder is in the training dataset. This data split does not prevent data leakage.

4 Memory-based model

4.1 Memory Based

This section is concerned with the two aforementioned memory based models, namely, user-based and item-based collaborative filtering. To predict a rating, the algorithm searches for similar behaviours of different users **U**, meaning, the way they rate **R** the movies **M**. Figure 5 illustrates a simple setting, to help the reader better visualise the concept. It displays two ratings for two movies, given by different users: A, B, C, and D.

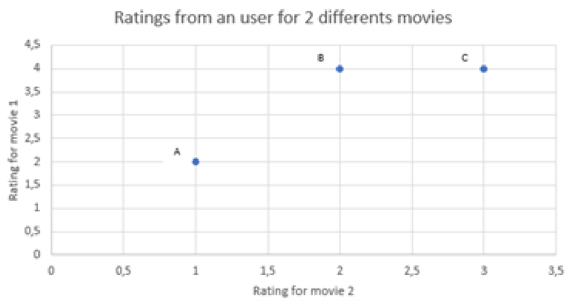


Figure 5: Example user-based similarity measure

At first glance, users B and C seem the most similar because they are closest to each other. However, this does not take into account that some users give higher ratings relative to other users. To correct for this, a line can be drawn from the origin to every user. The line from the origin to A and the one to B overlap, which indicates that these users have a similar rating pattern. When the angle increases the similarities decrease. To compute the angle the cosine is used. The higher the angle, the lower the cosine will be, thus the lower the similarity between users. The similarity factor used is $1 - cosinedistance$ (Lahitani et al. 2016). To correct for the previously mentioned rating bias, the average rating over all movies was subtracted from each rating.

The weighted average approach was used to predict the ratings. Each rating will be multiplied by a similarity factor. By doing so, it adds weights to the rating. A rating with a large weight matters more then one with a small weight. The weighted average can be calculated with the following formula, with **S** the similarity factor for each user that is similar to the target user **U**. (Sarwar et al. 2001)

$$R_U = (\sum_{u=1}^{n} R_u * S_u / \sum_{u=1}^{n} S_u) \tag{1}$$

The implementation of both user-based an item-based collaborative filtering was achieved through the use of the *KNN* algorithm, a non-parametric learning algorithm that makes no assumption on the underlying data distribution. Instead, when making an inference about a movie, it computes the 'distances' between the target movie and every other movie in the dataset, ranks the distances and returns the top k nearest movies as the most similar recommendations. The user-based approach is structured very similarly, but from the perspective of users instead of the items. Generally, the item-based approach is preferred because of the dynamic nature of the set of users, while the items remain mainly unchanged.

Several models were compared, each representing a variation of the *KNN* algorithm. The minimisation objective of the comparison were the metrics *RMSE* and *MAE*, based on the formulae below:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{N} (x_i - \hat{x_i})^2}{N}} \tag{2}$$

$$MAE = \frac{\sum_{i=1}^{n} |(y_i - x_i)|}{n} \tag{3}$$

In equation (2), x_i refers to the actual rating, $\hat{x_i}$ is the predicted rating and N is the total number of ratings. Similarly, in equation (3), y_i is the predicted rating and x_i is the actual rating.

5 Model-Based models

In the following section, two model-based collaborative filtering models are discussed. Both models make use of embedding vectors for the prediction of a specific rating for a user-item pair.

An embedding vector is a vector of dimension K (usually selected using hyperparameter selection) which can be assigned to an object or category to specify certain characteristics. In this case we use them to specify certain characteristics of users and movies, so every item and movie gets its own embedding vector dependent on its characteristics. An intuitive way to see it is that entry i of the embedding vectors of a specific movie and user give a score related to a specific characteristic of that movie and how much the user likes that characteristic. It might also be intuitive to visualise that, if we regard the embedding vectors as coordinates, similar users and movies are plotted nearby in the K dimensional plane.

The first model will be based on singular value decomposition. This is a linear approach. The second model will be a deep learning approach which will be able to see non-linear relations.

5.1 Matrix decomposition (SVD)

Singular Value Decomposition (SVD) is a popular dimensionality reduction algorithm in Machine Learning. The aim of the algorithm in this context is to decompose the large matrix with ratings (R) into two smaller matrices (U and M) [1]

$$R = U^T M \tag{4}$$

The idea behind this factorization is that the matrices U and M will contain the embedding vectors mentioned in the introduction of this section. The struggle is that there is no clear initial representation for the movies or the users. This problem is solved by firstly creating random embedding vectors, and optimizing the contents of these vectors to give us good representations. The representations are good when the dot product between U and M are approximately equal to the original matrix R

$$U^T M \approx R \tag{5}$$

To optimize these embedding vectors, there is the need for a function that takes the dot product of the two embedding vectors and outputs a high value if the user would rate the movie highly, and a lover value if the user would dislike the movie. This function is called the score.

$$score(u, m) = u^T m \tag{6}$$

[1]Derivation inspired by embedding models lecture

$$score(u, m) = u_1 m_1 + u_2 m_2 + ... + u_k m_k \tag{7}$$

Since the matrix **R** is very empty, loss function (3) is only defined for the known ratings. Otherwise the model would develop a bias by training on the zeros indicating missing ratings.

$$\text{argmin}_{U,M} \sum_{i,j \in R_{known}} (R_{i,j} - u_i^T m_j)^2 \tag{8}$$

For the optimization, stochastic gradient descent is used for versatility and scalability purposes. The goal is to minimize the difference **E** between the true **R** and the product of the embeddings.

$$E = R - U^T M \tag{9}$$

Then the partial derivatives are taken (10), (11) where $U_{k,u}$ denotes the k'th embedding for user u and $M_{k,m}$ denotes the k'th embedding of movie **m**. These partial derivatives work out to be (12), (13) and will be used in the stochastic gradient descent step.

$$\frac{\partial L}{\partial U_{k,u}} \tag{10}$$

$$\frac{\partial L}{\partial M_{k,m}} \tag{11}$$

$$-\sum_j E_{u,j} M_{kj} \tag{12}$$

$$-\sum_j U_{m,j} M_{kj} \tag{13}$$

The only thing the algorithm still needs is to declare the hyperparameters that will be tuned:

1. Learning rate η: the step size at each iteration
2. Regularization: penalty for larger models
3. Number of embedding factors K: dimension of latent factors
4. Epochs: number of passes over the training set

5.2 Deep Learning

A neural network is a series of algorithms that tries to capture the relationships in a dataset. This process is inspired by the biological neural networks that can be found in the brain. Recently, the use of neural networks with embedding layers has been researched more extensively for the prediction of user ratings. Zarzour et al. (2019) proposed a particularly nice model which was the inspiration to incorporate a deep learning into this research. Deep learning is a subset of machine learning which makes use of a neural network consisting of more than one hidden layer. Such a neural network is often referenced to as a deep neural network.

5.2.1 The Network

The Deep Learning approach to rating prediction by collaborative filtering is essentially an extension to the SVD model. The SVD model can be seen as a neural network with a single embedding layer and a linear layer (Bermeitinger et al. 2019) transforming the embedding vectors into predicted ratings by taking the dot product.

In the deep learning network the following layers were used:

- Input layer: the input layer has the user and movie combinations for which ratings need to be predicted.

- Embedding layer (First hidden layer): This layer gets the embeddings as inputs and adds a bias factor for every movie and user accordingly
- Second hidden layer: Here the dot products of the embedding vectors are calculated and activated by the ReLU function.
- Third hidden layer: Dense, sigmoid activated layer. The following formula is used:

  $$Sigmoid(W_i V_{i-1} + A_i)$$

  With A_i the weights, V_{i-1} the output from the precious layer and A_i the matrix of biases.
- Output layer: This layer is used to convert the scale of the output back to a rating between 0.5 and 5 since the sigmoid function outputs a rating between 0 and 1. Each node in this layer is connected with one unique node from the previous layer. The input is modified by the following formula:

  $$input * (rating_{max} - rating_{min}) + rating_{min}$$

  with the maximum and minimum rating taken from the whole dataset (so most certainly 5 and 0.5 respectively in this setting).

5.2.2 Methods used for optimization and hyper parameter selection

The the deep neural network was trained using a batch size of 32 and the Adam optimizer. It is a clever optimizer which is, by many, seen as the best optimizer in most situations. It is beyond the scope of this research to explain exactly how Adam works, but it makes use of a combination of gradient descent, momentum and adaptive learning rate.

For the hyper parameter selection, the Hyperband tuner was used. The Hyperband tuner is an adaptation to random search, but with the main upside that it quickly terminates unfavorable runs. Since the deep learning model is quite big and takes a lot of time to train and tune, choosing the tuner was favored.

The decision was made to tune the parameters; learn rate, regularization parameter and K which is the dimension of the embedding vectors. The optimal parameters can be found in the results section.

6 Results

A summary of all the results:

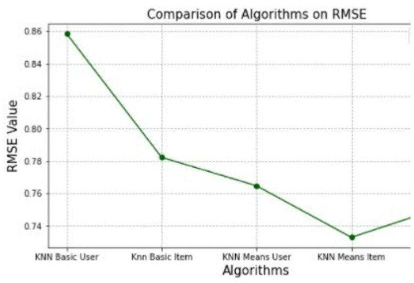
Model	RMSE
Baseline (mean)	0.90
Memory based	0.73
SVD	0.73
Deep learning	0.78

Table 1: Results

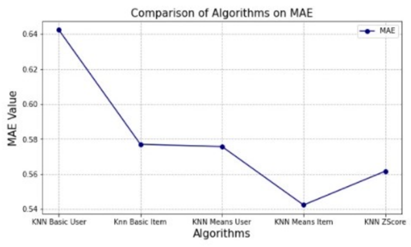
We can thus conclude that the Memory based and SVD models worked best.

6.1 Memory Based

The models used, along with their performance can be best visualised in the figures below. In order to minimize the chance element, two performance metrics were computed for each model, thus ensuring that their performances are consistent and not due to chance. Indeed, it can be noted that the 2 graphs follow a similar pattern, both pointing to the item-based method implemented with *KNNMeans* being the best one. The low values for the 2 metrics translate to high accuracy and given that the and both have range of $[0, \inf]$, it can be stated that, generally, these models all perform well, given the values below 1 for the aforementioned functions. The fact that the item based model is preferable does not come as a surprise, as it was also the optimal hyperparameter rendered by performing *GridSearch* on the *KNNBasic* and *KNNMeans* algorithms. It should be noted that *KNN Z score* is the next best performing model. This model takes into account the *z-score normalization* of each user. This approach has the potential to greatly increase accuracy by representing the original data into new data with a similar, but narrower range, thus simplyfying it so that the model can perform optimally(Muhammad et al.,2020).



(a) Performance comparison using *RMSE*

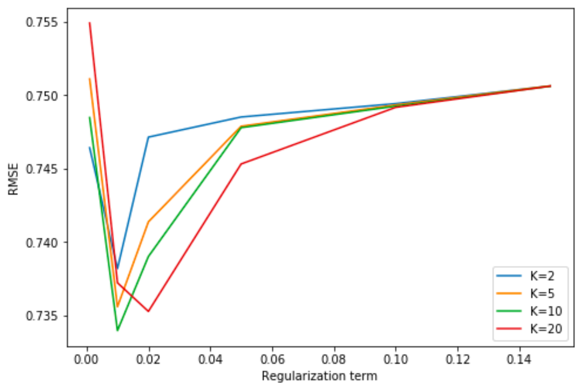


(b) Performance comparison using MA

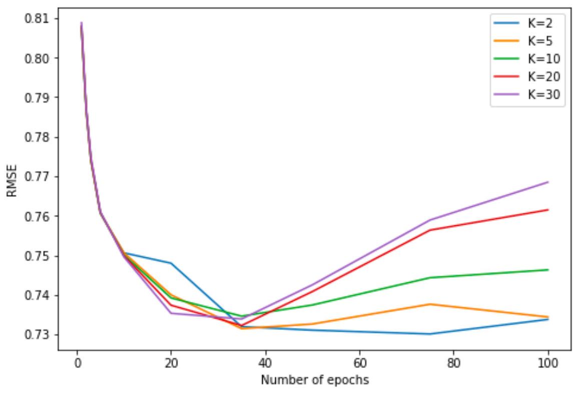
Figure 6: Algorithms' performance assesment

6.1.1 SVD

One challenging aspect was tuning the parameters, since there were many. Just using a big *Gridsearch* was not feasible since the combinations of parameters was large, while the algorithm was computation intensive, especially with the large dataset. However, by trial and error and some hyperparameters plotted against each other, as in figure 7, it is possible to obtain an understanding of the parameters' behaviour.



(a) Regularization vs *RMSE* for several K values



(b) Epochs vs *RMSE* for several K values

Figure 7: Hyper parameters plotted against each other

The minima could be used as a starting point for the *gridsearches* that would follow. Eventually after running the *gridsearch* many times for different combinations and evaluating them on the validations set, the following optimal parameters values were obtained and summarized in table 2:

Epochs	60
Learning Rate	0.02
Regularization	0.1
K	100

Table 2: Optimal parameters for SVD

This resulted in a *RMSE* of 0.650144 on the validation set and 0.730914 on the test set.

6.1.2 Deep Learning

The parameters used in the deep learning model can be found below in table 3:

Learning Rate	5.496e-04
Regularization	6.194e-05
K	33

Table 3: Optimal parameters for deep learning

The found *RMSE* is unexpected, but this will be elaborated on further in the discussion.

7 Conclusion

From the results it can be concluded that the hypothesis can be rejected. The deep learning model did not outperform the other models like expected. The SVM (model based) and KKN (memory based) performed equally well and thus it can not be concluded if either a model based approach or memory based approach is better in this specific setting.

7.1 Discussion

The first improvement could be the availability of more processing power. In this research, the models were trained on relatively small data sets to keep training times manageable.

Secondly, a better training approach could have been taken to prevent data leakage. A suitable model was proposed in the paper of Ji et al. 2021. This paper proposes a model in which the data split is kept the same, but models train from oldest to newest rating and when it encounters a test instance it uses the already trained model to predict. After that, it continues training on the training data until the next test instance arises.

Lastly, the structure of the deep neural network deviated from ones most commonly found in similar research papers in which the embedding vectors were concatenated after the embedding layer instead of dotted.

8 References

1. Aditya, P. H., Budi, I., Munajat, Q. (2016, October). A comparative analysis of memory-based and model-based collaborative filtering on the implementation of recommender system for E-commerce in Indonesia: A case study PT X. In 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS) (pp. 303-308). IEEE.

2. Bermeitinger, B., Hrycej, T., & Handschuh, S. (2019). Singular Value Decomposition and Neural Networks. Lecture Notes in Computer Science, 153–164. https://doi.org/10.1007/978-3-030-30484-3_13

3. Su, X., & Khoshgoftaar, T. M. (2009). A survey of collaborative filtering techniques. Advances in artificial intelligence, 2009.

4. Herlocker, J. L., Konstan, J. A., & Riedl, J. (2000, December). Explaining collaborative filtering recommendations. In Proceedings of the 2000 ACM conference on Computer supported cooperative work (pp. 241-250).

5. Zarzour, H., Al-Sharif, Z. A., & Jararweh, Y. (2019). RecDNNing: a recommender system using deep neural network with user and item embeddings. 2019 10th International Conference on Information and Communication Systems (ICICS). https://doi.org/10.1109/iacs.2019.8809156

6. Ji, X., Sun, A., Zhang, J. ,& Li, C.A Critical Study on Data Leakage in Recommender System Offline Evaluation.(2021)

7. Sarwar, B., Karypis, G., Konstan, J., & Riedl, J. Item-Based Collaborative Filtering Recommendation Algorithms. (2001).

8. Lahitani, A. R., Permanasari, A. E., & Setiawan, N. A. Cosine Similarity to Determine Similarity Measure: Study Case in Online Essay Assessment. Cosine Similarity to Determine Similarity Measure: Study Case in Online Essay Assessment.(2016).

9. Muhammad, A. Imron & Budi Prasetiyo, Improving Algorithm Accuracy Using Z-Score Normalization and Particle Swarm Optimization to Predict Customer Churn.(2020)

10. Vlachos, M.,Dunner, C. ,Heckel, H., Vassiliadis,V., Parnell,T. & Atasu, K., Addressing Interpretability and Cold-Start inMatrix Factorization for Recommender Systems. (2018)

9 Appendix

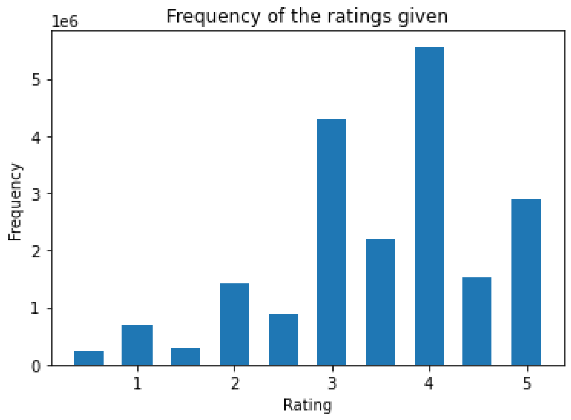


Figure 8: Distribution of the ratings

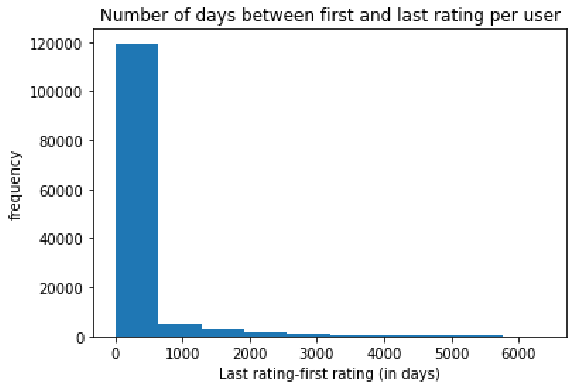


Figure 9: Shows how many days between first and last rating