Advances in Data Mining Assignment 1 - Group 80

Abhiram Elangovan (s3442403)

Nick van Vliet (s2096897)

Nicolette Duijn (s3026752)

25 October 2022

1. Introduction

Data has become the most valuable asset for companies. Some even call it the new gold. Data on user preferences helps companies to tailor their services to each individual and their needs. Profiling on preferences requires your data, this can vary from your search terms to your shopping history. Modern companies collect different types of data to improve their services. One of these companies that is an industry leader in this area is Netflix. Watching your favorite shows feeds into your profile and helps Netflix' recommendation algorithm recommend more shows that might be interesting to you. Those recommendations are not only based on what you watch, but also what others watch with similar profiles. This information is so valuable to Netflix that they decided to crowdsource the improvement of this algorithm behind the recommender system on a global scale. Netflix held a competition to improve the accuracy of the Netflix recommender system. Whoever managed to increase its accuracy with 10%, could claim a prize of 1 million dollars. This competition went on for over three years before someone claimed this prize.

In this paper we will write multiple recommender systems and test the accuracy of those systems. Firstly, we start with naive approaches, these are simplest approaches, the accuracy will not be the best, but it is a way to predict the ratings. For the naive approaches we implement the global average, user average, movie average and linear regression (with and without intercept). After that we will implement the UV matrix decomposition and we end with the matrix factorization. Once we write the matrix factorization algorithm, we will explain the concepts of PCA, T-SNE and UMAP and visualize the data using those dimension reduction techniques.

The data that is used in this paper is from the MovieLens site[1]. We use the dataset with 1 million ratings. There are around 3900 movies and 6000 users in this dataset. The dataset also provides extra information, for example which genre the movie is or the age of an user.

2. Exploratory Data Analysis

Before we delve into the problem we start with looking at the data and so get familiar with the data. Are there any missing values or how many movies does the user rate? This information is important before writing the algorithms. After doing initial analysis it can be concluded that there are no zeroes or NAs in each of the three columns and there are 6040,3706 and 5 distinct values in UserID,MovieID and ratings respectively.

From the distribution of userID (figure 1) it is shown that they are not even, and this makes sense from a data perspective as the same user need not have submitted ratings for the same number of movies. For movieID a distribution is created to see how many entries there are per movieID (see figure 1). For the movieID the same thing is true, for some movies there are many entries and for some very little.

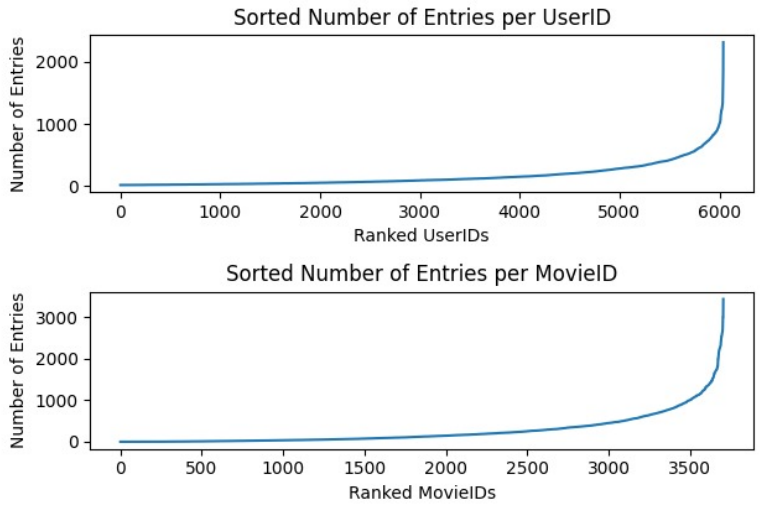


Figure 1: Number of entries per Movie and User ID's

To see what rating is mostly given to a movie, a distribution for the frequency is made (see figure 2). Number 4 is the most common rating followed by 3 and more than 1/3rd of the entire ratings is 4. The average rating for the dataset is 3.581564.

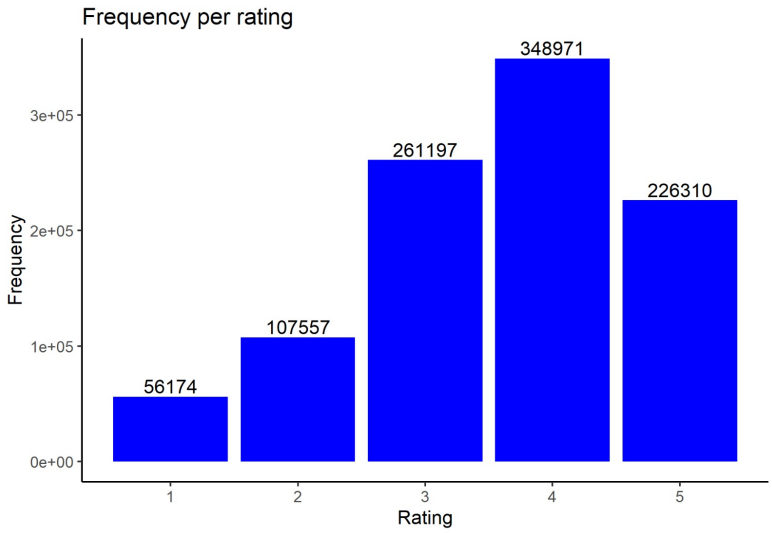


Figure 2: Frequency of each rating

3. Cross Validation Methods

One of the main pitfalls in analysis of datasets is the results' bias when training and running the model using the same data. This gives us almost always a better result and does not prepare the model for real world applications. This scenario is called overfitting.

One quick way to get around this is to split the model into 2, usually called a 80:20 split with the model being trained initially on a random 80% of the data and then tested on the rest 20% of it to ensure it can optimally perform when exposed to new data. However this also comes at a loss of not training the model with that 20% data in other words,impacting overall model accuracy as well.

3.1 K-fold cross-validation

In order to overcome this a method called K-fold cross-validation can be used. K-fold cross-validation is a superior approach because it uses different chunks of the dataset as a validation set. This works by dividing the dataset into k-folds where k is the number of parts that you want to split the data into and for each of the spit, one fold gets used for cross validation, thereby ensuring all of the data is still used to build the final model. A simple visual approach of 5-fold classification can be shown in table 1.

Table 1: 5 fold cross validation used for testing the models

Iteration 1	**Test**	Train	Train	Train	Train
Iteration 2	Train	**Test**	Train	Train	Train
Iteration 3	Train	Train	**Test**	Train	Train
Iteration 4	Train	Train	Train	**Test**	Train
Iteration 5	Train	Train	Train	Train	**Test**

4. Task 1

4.1 Performance Parameters:

The following performance parameters will primarily be used to evaluate the model fit and accuracy throughout the report.

(i) RMSE or Root Mean Squared Error can be used as a good evaluation metric.

RMSE = $\sqrt{\frac{\sum(x_i - \bar{x_i})^2}{N}}$

(ii) MAE or Mean squared error which rather calculates the average of the errors without accounting for the direction.

MAE = $\frac{\sum|y_j - \bar{y_j}|}{N}$

4.2 Naive Approaches

In this section the Naive approaches to recommender systems is written and evaluated. As known Naive approaches are a simple and a low computation intense method for building models. Naive approaches make use of aggregated means at a certain group level or linear models and use that for predictions.

In the Naive aproaches there are 5 different approaches to implement.

4.2.1 Global Mean

R_{global}(User, Item)=mean(all ratings)

Firsty, the global mean algorithm. The prediction is done for the unknown values by taking the mean of the given values. Implementing this model gives a RMSE of around 1.17 and similarly the MAE values were around 0.934. As expected this is not the best model to predict the ratings. This is one of the simplest models that can be build with this dataset and hence a better RMSE can be expected for the other models.

4.2.2 User Mean

R_{user}(User, Item)=mean(all ratings for User)

For the second model the average ratings per user is used to predict the unknown ratings for the movies the users have not rated yet. This model is again a simple model. For this model a RMSE of 1.027 is found and the MAE is 0.823. This model is already an improvement of the global mean model.

4.2.3 Movie Mean

R_{item}(User, Item)=mean(all ratings for User)

Similar to the user average, now the average per movie is taken to predict the unknown ratings. With this model a RMSE of 0.974 is received. It can concluded from those numbers that it is better to use the movie average over the user average and global average. again an improvement from the previous method.

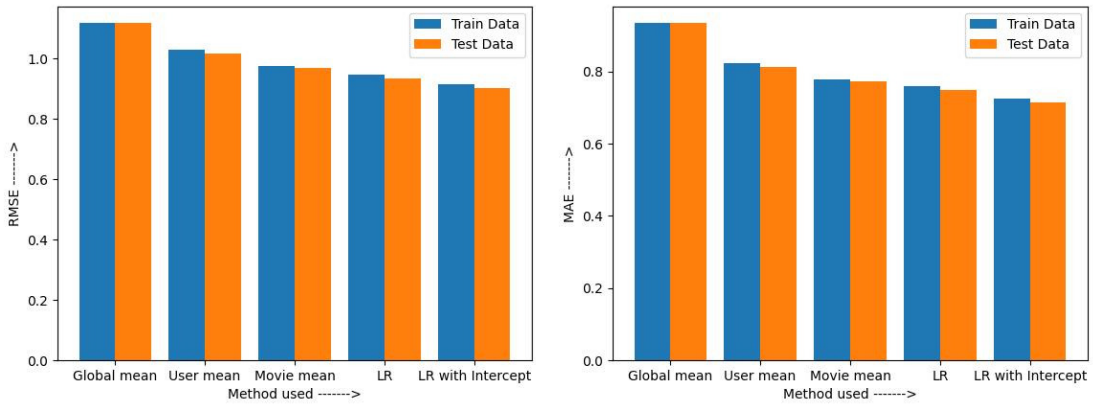
4.2.4 Linear Regression model

$pred = \alpha \cdot avg_{user} + \beta \cdot avg_{movie} + \gamma$

For linear regression the prediction is done for the ratings by taking the average user and the movie average and assign an alpha and a beta to these averages. For the linear regression there are two options: With or without the intercept γ. To see the difference of accuracy between the two models both the models are tried. The RMSE for the model without and with γ are 0.901 and 0.934 respectively. As expected, the accuracy is better for the model with γ.

4.2.5 Comparison of Train and Test data

In figure 3 the differences between the train and test of the 5 fold cross validation can be seen. The RMSE is always a little higher for the train data. The same thing is true for the MAE of the train and test data (see figure 3).



(a) RMSE test and train

(b) MAE test and train

Figure 3: An overview of the RMSE and MAE for every naive approach

In table 2 a summary of the values of RMSE and MAE are shown for the Naive Approaches, and their runtime.

Table 2: Error calculations and Runtime for each naive method

Method Used	RMSE	MAE	Runtime (s)
Global Mean	1.117	0.934	0.62
User Mean	1.038	0.829	5.01
Movie Mean	0.979	0.782	5.11
Linear Regression	0.934	0.749	1.37
Linear Regression with Intercept	0.901	0.713	1.41

It can be seen that when choosing a more complex mode with more variables here the better out resultant RMSE and MAE values get. In general a good rule when building a model is to ensure maximum accuracy (without overfitting) possible with minimal complexity of the models.

4.3 UV Matrix Decomposition

The UV Decomposition approach consists of re-imagining the utility matrix as a product of two long and thin matrices and hence the name UV decomposition. It is also an application of Singular Vector Decomposition.

(i) Consider the utility matrix with n rows - users here and m columns - movies, then there is a matrix U with n rows and d columns as well as another matrix V with d rows and m columns such that the product results in the original n x m matrix.

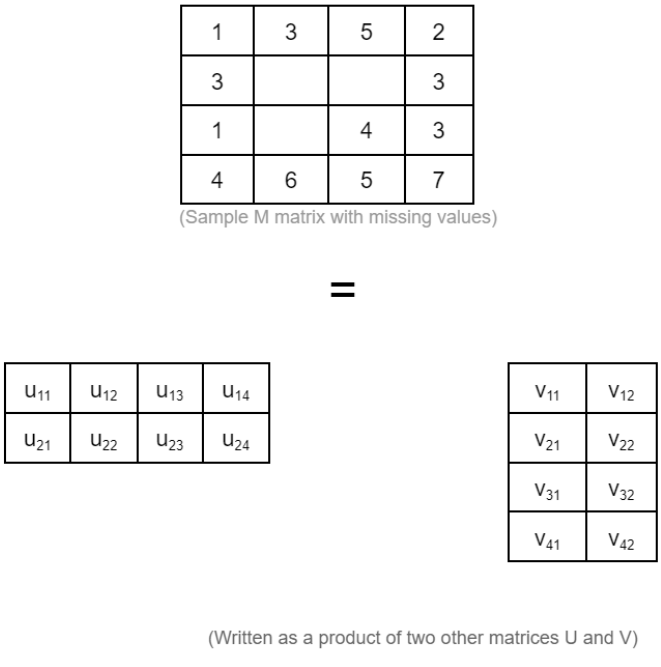


Figure 4: An example of UV multiplication

(ii) To work with the optimisation, it can be noted that there can be several values that can follow the above decomposition condition and hence the values closer to RMSE are chosen. We implement it in our code by taking the difference in squares of non- blank entries in M by the ones in the UV matrix and using the same to compute the RMSE with n being non-blank entries from matrix M.

(iii) Now in-order to get the optimal solution to get the one with the lowest RMSE it is needed to iterate with every element of U. Starting with u11, for every unknown element k, will result in the MSE computation being a polynomial with 2 degrees. This is differentiated with respect to k to obtain the point of minima. And substituting this value of k it can be seen the RMSE is getting lower. Similarly, the same process is repeated with the first element of V namely v11 and is looped over for all the elements till, it optimal RMSE is found.

4.3.1 Optimal RMSE

Since in the given example the number of blanks on the original matrix are much greater than the total number of elements in U and V together, hence a zero value for RMSE is not possible. Hence the amount of improvement in RMSE can be tracked with each round and stop the loop once RMSE falls beyond a certain threshold indicated by the variable Threshold for which is set 0.0001 as a default value.

4.3.2 Conclusion with UV Optimisation

Apart from changes in iterations, modifying the threshold is tried to make much lesser values (in terms of Geometric Progressions as recommended) and it did not impact the RMSE significantly.

This is due to the fact that for UV decomposition models the ideal RMSE is heavily reliant on the number of missing values in the original matrix vs the total number of values in U and V matrices. If the number of missing values is much greater then the total values in user and item matrices combined (which is the case here) the RMSE cannot be improved beyond a certain extent with iterations and we will have to look at advanced methods like Matrix Factorization which we will discuss in the next section. Table 3 shows the overview of the RMSE and MAE and how long it took to run. The UV matrix decomposition is with a RMSE of 0.8606 much more accurate than the naive approaches.

Table 3: Error calculations and Runtime for UV decomposition

Iteration Kfold	RMSE	MAE	Runtime (s)
1	0.8605	0.6787	83.14
2	0.8614	0.6793	62.50
3	0.8604	0.6783	44.96
4	0.8606	0.6786	52.86
5	0.8599	0.6782	38.99
Average	0.8606	0.6786	56.67

4.4 Matrix Factorization

Matrix factorisation is a famous Singular Value Decomposition algorithm originally proposed by Simon Frank during his attempt on the Netflix challenge[2]. By using SVD it helps us predict a rating for every user-item pair and since unlike other recommender systems this also outputs an RMSE or MAE and thereby enables us to understand the metric better before deploying the recommender's output in the real world. For this matrix factorization the method is based on the gravity-Tikk.pdf paper[3].

In Matrix factorisation a matrix of users by items is created and a list of ratings associated with them is computed. This is called the weight matrix which is something that is not observed in raw data but can be analyzed as a representation of ratings given by the user to the item. To begin there are three matrices, shown in tables 4, 5 and 6.

(i) Firstly, table 4 shows an example of a possible User-Movie matrix.

Table 4: Potential User-Movie matrix. Values are ratings.

	Movie 1	Movie 2	Movie 3	Movie 4
User 1	NaN	NaN	4	1
User 2	3	NaN	2	NaN
User 3	5	NaN	NaN	1
User 4	NaN	2	NaN	NaN

It is indeed worth noting that this table has a lot of empty values since all users would not have given a rating for all the movies in the list, which are replaced with 0's.

(ii) Table 5 shows an example of the M matrix, the movies matrix (initialized with random values)

Table 5: An example of the Movies matrix, initialized with random values.

	Movie 1	Movie 2	Movie 3	Movie 4
Feature 1	-1.8	1	-0.2	1
Feature 2	0.5	1.2	0.1	5
Feature 3	1.4	4	0.14	2

(iii) Table 6 shows an example of the U matrix, the User matrix (initialized with random values).

Table 6: An example of the U the User matrix, initialized with random values.

	Feature 1	Feature 2	Feature 3
User 1	0.8	1.2	-0.2
User 2	2	1.8	0.4
User 3	0.8	3	0.1
User 4	1	0.8	2.4

Multiply the matrix U with users x k features by the matrix M with k features x movies, results in the original matrix X.

(iv) Next is the computation of predicted value from the weight matrices, for every value that is not zero in the original matrix X.

Since the user and movie matrices are filled with random numbers. The multiplication of those matrices will result in a number. This new number is the prediction of the weight matrices.

(v) Now the error is the actual rating (from matrix X), minus the predicted rating calculated from the weight matrices. Again the error is only calculated for the values where the rating is nonzero. The objective is to minimize this error using gradient descent and by choosing the right number of iterations and learning rate under the variable **learn_rate**.

For U[i] and M[j] from U and M matrices respectively and using U[i] + 2 x learn_rate U[i] we can iteratively update values for U[i] and M[j] till our target MSE is achieved. In addition to the above, there is also the need of a regularization coefficient controlled by parameter regularization. The need for this regularization coefficient is to avoid overfitting of data especially in cases where the training set is relatively smaller than that of the number of model parameters. This regularization coefficient impacts the learning rate as: U[i][k] += learn_rate * (2 * ErrorValue * M[k][j] - regularization * U[i][k]) where ErrorValue is the computed error from the previous iteration. Those formulas for the gradient and the regularization are used to update the weight matrices after each iteration.

Hence summarizing this together, for the MatrixFactorisation function we have the following inputs: **dataset, num_factors, num_iter, regularization, learn_rate**.

Within the function, the test and train datasets are formed, and the RMSE is calculated by dividing the total error by the number of non zero values. Table 7 gives an overview of the outcomes with input: num factors=10, num iter=75, regularization=0.05, learn rate=0.005. The average RMSE is 1.332, which is higher than for the other algorithms. Which means the matrix factorisation is the worst predictor of all the approaches.

Table 7: Error calculations for matrix factorization for the Train and test set with runtime for each fold.

	Train		Test		
Fold	RMSE	MAE	RMSE	MAE	Runtime (s)
1	0.9101	0.8487	1.329	1.040	127.1
2	0.8981	0.8430	1.347	1.047	192.3
3	0.8986	0.8433	1.301	1.027	192.2
4	0.8998	0.8439	1.326	1.038	191.8
5	0.8961	0.8417	1.358	1.051	194.0
Average	0.9005	0.8441	1.332	1.041	179.56

From table 8 it can be seen that among all the parameters used for the Matrix Factorisation model, the learning rate seems to be a dominant contributor to the resulting RMSE for the model. This is further a good conclusion considering that other parameters iterated over like regularisation impact the model less since they are bounded by learning rate in being the major contributor to RMSE.

Table 8: Matrix factorization algorithm with different parameters.

N_factors	N_iter	Regularization	Learn_rate	RMSE_test	RMSE_train
10	75	0.05	0.005	1.319	0.898
10	25	0.5	0.005	1.335	0.953
10	25	0.05	0.05	1.508	1.069
10	25	0.05	0.003	1.326	0.896
10	25	0.05	0.001	1.320	0.889
10	25	0.3	0.003	1.300	0.891

The significant difference between RMSE_test and the RMSE_train for train datasets further adds to the importance of the need to use appropriate cross-validation methods in our models and if otherwise a model performing well with current data might not perform as expected in the real world.

4.5 Big O Calculation

We can recollect that for a Matrix multiplication of an MxN and a NxK matrix the Big O rule states that we need O(MNK) time and another O(MK) memory involving a total of N multiplications and N-1 additions.

There were two matrices in the matrix factorization one with 6040 times 10 inputs and one with 10 times 3706 inputs. For num_iter interations, where num_iter varies from 25 to 75, we get the same process iterated but over num_iter number of times we have the resulting value was product of O(MNK) x 75 in terms of time and O(MK) memory with M = 6040, N=10 and K=3706. And it is also worth noting that all these computations are for a single core processor and in modern computers will multicore processors this computing speed can hence get accelerated.

For computations in 1.2 under UV factorisation, we have similar values except that N =2 and we do not have control over the number of iterations but rather let the model choose it's own num_iter value. This makes sense since it is a significantly less complex model and hence not very computation power dependant. Hence have the resulting value was product of O(MNK) in terms of time and same O(MK) memory with M = 6040, K=3706.

5. Task 2

The importance of Data Visualization is to derive insights from data that can be actionable. And hence in this section of the assignment we will discuss how we can make use of the generated outputs from Matrix Factorization methods above to identify something meaningful.

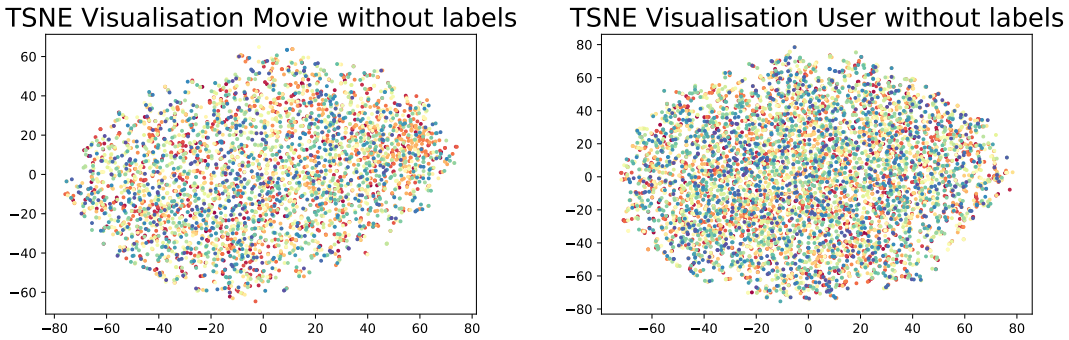
For the data visualization of the user matrix and movie matrix, 3 methods are used for dimension reduction: T-SNE, PCA, and UMAP.

5.1 Visualisation of the methods

Firstly, t-SNE stands for T-Distributed Stochastic Neighbor Embedding and it uses a probabilistic approach and it is useful for high dimensional datasets in particular. t-SNE works by creating a single map that works to reveal structure at many different scales. According to the authors of the original paper[4] on t-SNE, "T-distributed stochastic neighbor embedding (t-SNE) minimizes the divergence between two distributions: a distribution that measures pairwise similarities of the input objects and a distribution that measures pairwise similarities of the corresponding low-dimensional points in the embedding."

It is worth noting that since this technique uses heavy computations, it might have some limitations in terms of computation. So as a benchmark when t-SNE is used for datasets consisting of more than a few thousand inputs it might be slow and impractical as summarized by the original paper on t-SNE.

It is also worth noting that due to the computational intensity of tSNE it is sometimes runned with PCA which is discussed in the next section first for dimensionality reduction and later used with t-SNE. Figure 5 shows the t-SNE visualisation for the movie and user matrix, without any labels added to the points, the points are so the movieID's and userID's. As expected the t-SNE plots without labels seems random distribution of different ratings and do add value to the fact that all the ratings by itself are random and are not biased inherently.



(a) Movie Feature matrix

(b) User Feature matrix

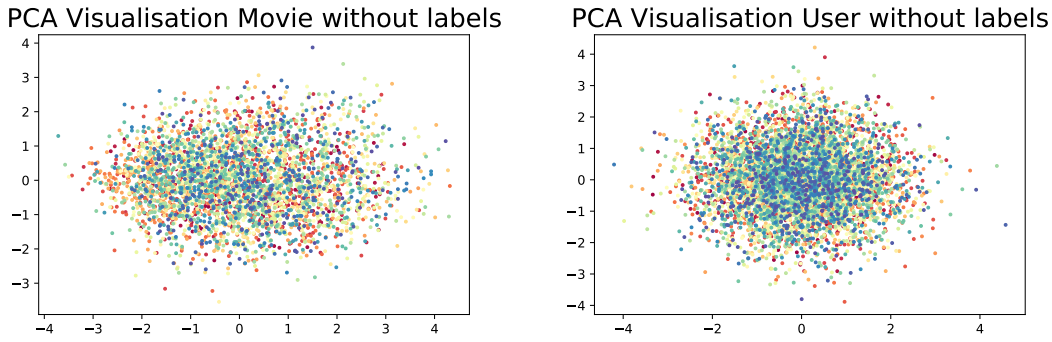
Figure 5: t-SNE visualisation for the movie and user matrices

Secondly, PCA is a mathematical technique that is used to reduce the number of dimensions in a dataset without losing out on inherent information. This achieved by orthogonal linear transforming the data. Here the variance explained is the highest for the first component and the lowest for the last component. It is asked to show the data in 2D, so only the first two compontents are used. By choosing the number of PCA components we can optimize between

reduced dimensions and the amount of information that is retained from the original data. With those new dimensions the feature vectors and there patterns can be shown.

The actual computation uses eigenvalues and eigenvectors for the resulting metric to analyze the percentage of variance explained by the new dimensions.

Figure 6 shows the PCA visualisation for the movie and user matrix, without any labels added to the points, the points are so the movieID's and userID's.



(a) Movie Feature matrix

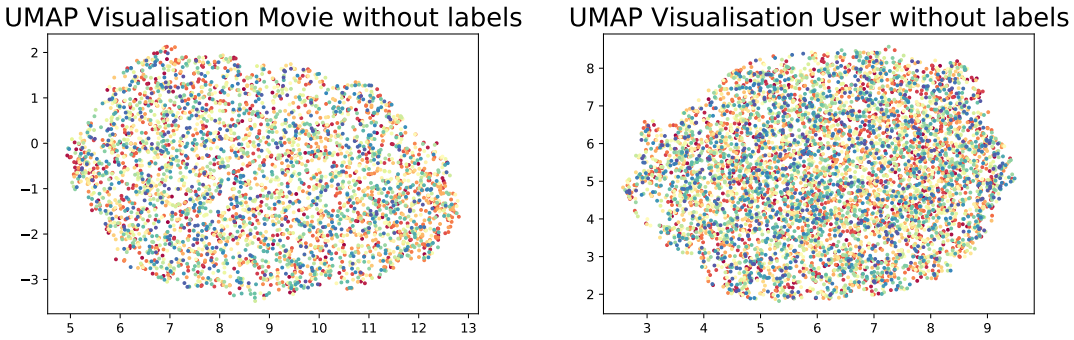
(b) User Feature matrix

Figure 6: PCA visualisation for the movie and user matrices

Lastly, UMAP which stands for Uniform Manifold Approximation and Projection, works similar to t-SNE for dimensionality reduction but it works in general for non-linear dimension reduction as well. But according to the official documentation we also need to keep in mind that the data has to be uniformly distributed on the Riemannian manifold and the Riemannian metric is a local constant. More of this is made available in the paper for UMAP[5][6] and the official documentation.

In contrast with t-SNE, while t-SNE moves the graph point-to-point from high to low dimensional space, UMAP makes a fuzzy, but topologically similar graph and compresses it into a lower dimension.

Figure 7 shows the UMAP visualisation for the movie and user matrix, without any labels added to the points, the points are so the movieID's and userID's.



(a) Movie Feature matrix

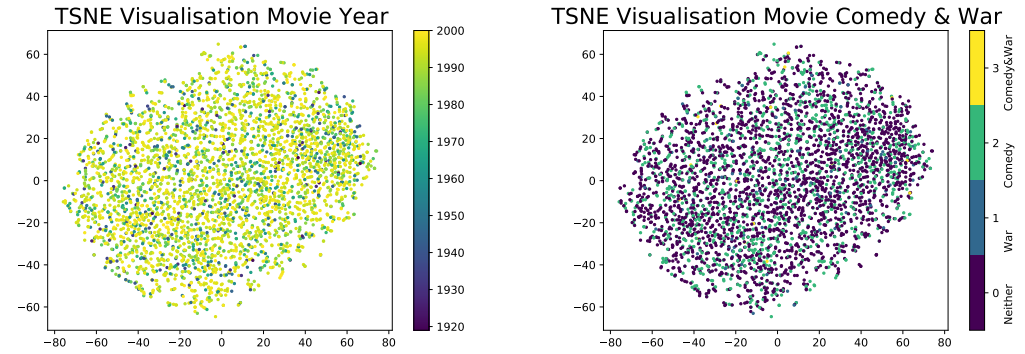
(b) User Feature matrix

Figure 7: UMAP visualisation for the movie and user matrices

5.2 Data driven inferences

To see what the dots on the visualisation represent and to see if there is any clustering present, labels of movies and users are added. For the movies those labels can be the year the movie is released or the genre the movie falls within. For the users the information about age and gender is present.

In figure 8 the t-SNE plot of movies and label years is shown. It can be seen that a major chunk of the movies are one released between 1990 to 2000s and this is a good representation of the face on how Netflix has grown and how it's library consists of increasing titles released in the recent times. There is now clustering present for the years.



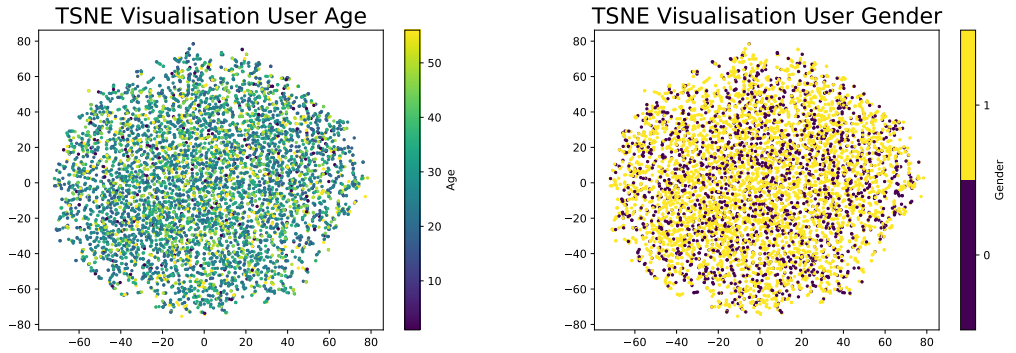
(a) Movie Feature matrix with years as label

(b) Movie Feature matrix with genre as label

Figure 8: t-SNE visualisation for the movie matrix with labels

Figure 8 also shows a t-SNE visualisation for the movie matrix with the comparison for the genre war and comedy. A comparison for all the genres is not possible since some movies have multiple genres. Therefore is chosen for a comparison between 2 genres that lie far from each other: war and comedy. It can be seen that most movies do not have the label war or comedy. There is no sign of clustering for the genres.

For the user matrix, there are two possible labels: gender and age. The t-SNE visualisation of the user matrix is shown in figure 9. The illustration gives an idea about the general population of Netlfix users and it can be seen that the target user audience for majority of users belongs to 20-30s. There is some clustering present for the user aged above 50.



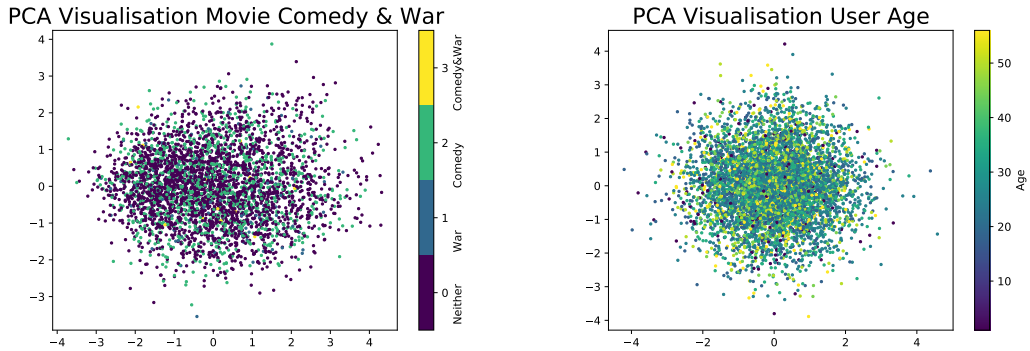
(a) User Feature matrix with age as label (b) User Feature matrix with gender as label

Figure 9: t-SNE visualisation for the user matrix with labels

The user t-SNE visualisation, shown in figure 9, reveals that majority of the users of Netflix have inputted their gender to belong to Gender 1 which represents the males. There is no clustering visible.

For PCA and UMAP the same labels as for t-SNE are added to the visualisation. Since for all the labels for the different methods there is no sign of clustering, only some of the plots are shown. For the rest of the plots see the DataVisualisation file.

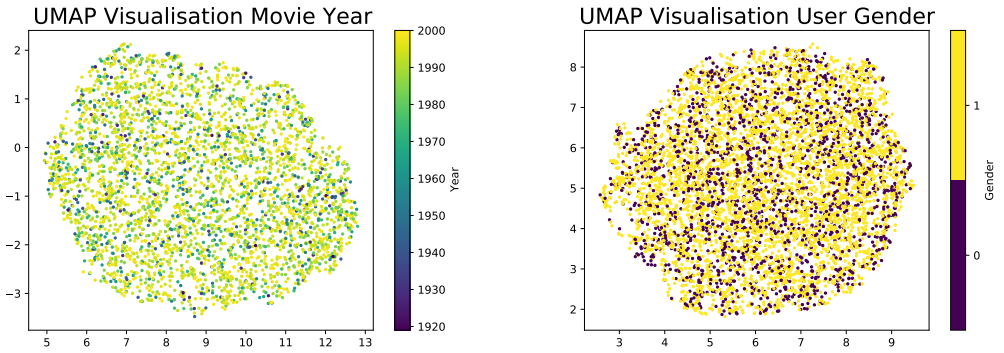
Figure 10 shows the movie feature matrix with the comparison between the genres war and comedy and the user feature matrix with the label age for the method PCA.



(a) Movie Feature matrix with genre as label (b) User Feature matrix with age as label

Figure 10: PCA visualisation for movie and user matrix with labels

Figure 11 shows the movie feature matrix with the label years and the user feature matrix with the label gender for the method UMAP.



(a) Movie Feature matrix with year as label

(b) User Feature matrix with gender as label

Figure 11: UMAP visualisation for movie and user matrix with labels

6. Discussion

In this report we discussed multiple algorithms for recommender systems. These systems, each with their own advantages and disadvantages. Naive approaches are very powerful due to their computational simplicity, this results in fast runtimes but poor predictive results. Even including more complex computational approaches like linear regression, the runtime of these simple recommender systems rarely exceeds 5 seconds. The disadvantage is a significantly higher RMSE and MAE, these are 0.901 and 0.713 respectively. However the algorithm accomplished this result in only 1.41 seconds. The naive approaches all have O(1) giving them an extra benefit over the algorithms discussed afterwards. A major advantage since this means that computational time and memory usage barely scale as the size of the dataset increases.

UV decomposition is the next algorithm discussed, this algorithm is a significant improvement over the naive approaches. The UV decomposition's calculation takes significantly longer. With an average computational time of 56.67 s the algorithm yields a decrease of the RMSE and MAE of 4.5 and 4.8 percent respectively. This approach has the additional disadvantage of scaling poorly as the size of the dataset increases, as shown by the increased computational time. It is quite important to keep this tradeoff into consideration as the increased dataset size will quickly start to mount and become hard to work with. It is important to note that an increase in size isn't limited to additional entries, but can also include more information and thus more columns.

Subsequently we used matrix factorization to predict the rating of users within the dataset. This approach should be giving even better results than the UV decomposition. The gradient descent algorithm is superior over the algorithm used in UV decomposition. The matrix produced by this method will be fed into the data visualization part on this paper. Unfortunately after repeated attempts the data factorization didn't perform optimally which influenced the data visualisation part of this assignment, perhaps with a better matrix factorization the clustering would be more present in this visualisation. This method, which should be an improvement over UV decomposition, had a RMSE and MAE of 0.9005 and 0.8441 respectively. The algorithm did this in an average time of 179.56 seconds. One of the considerations that have to be mentioned is that the dataset of matrix factorization is slightly larger than the UV decomposition dataset, as

it includes both training and test datasets. After these considerations the matrix factorization algorithm actually failed to achieve an improvement over both the RMSE and MAE of the global mean recommender.

Lastly we discussed that the best algorithm for the given application in terms of dimensionality reduction and visualization to be PCA. This also conforms to the theory that t-SNE is a probabilistic model and works poorly for data points greater than thousands and UMAP has pre-existing data distribution requirements with respect to Riemannian manifold which could not be the case for this given data. Furthermore, by testing the model for better parameters (increase in num_iter, lower learning rates which could be more hardware intensive) and implementing a better version of the Matrix Factorization algorithms one can result in better ratings predictions which consequently will give us better results/insights when subject to the above algorithms.

7. References

1. MyMediaLite: Example Experiments. (z.d.). Geraadpleegd op 26 oktober 2022, van http://mymedialite.net/examples/datasets.html

2. Netflix Update: Try This at Home. (n.d.). Retrieved October 26, 2022, from https://sifter.org/%7Esimon/journal/20061211.html

3. Pilászy, I., Németh, B. & Tikk, D. (2007). Major components of the gravity recommendation system. ACM SIGKDD Explorations Newsletter, 9(2), 80–83. https://doi.org/10.1145/1345448.1345466

4. Visualizing Data using t-SNE. (2008). Journal of Machine Learning Research, 9(86), 2579–2605. http://isplab.tudelft.nl/sites/default/files/vandermaaten08a.pdf

5. UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction — umap 0.5 documentation. (n.d.). Retrieved October 26, 2022, from https://umap-learn.readthedocs.io/en/latest/

6. McInnes, L. (2018, February 9). UMAP: Uniform Manifold Approximation and Projection for Dimension Reduction. arXiv.org. https://arxiv.org/abs/1802.03426