UNIVERSITY OF MACEDONIA
POSTGRADUATE PROGRAMME
DEPARTMENT OF APPLIED INFORMATICS

A RECOMMENDER SYSTEM TO PREDICT THE BEHAVIOUR OF AN E-COMMERCE PAGE VISITOR

Msc Thesis

by

© Angelos Martidis

Thessaloniki, February 2023

A RECOMMENDER SYSTEM TO PREDICT
THE BEHAVIOUR OF AN E-COMMERCE PAGE VISITOR

Angelos Martidis

Diploma in Civil Engineering, UTh, 2020

Msc Thesis

A thesis submitted in partial fulfillment of the requirements for the degree of

MASTER DEGREE IN APPLIED INFORMATICS

Supervisor
Dr. Ioannis Refanidis

Approved by the three-member committee on 01/03/2023

Dr. Ioannis Refanidis, Professor	Dr. Dimitrios Hristu-Varsakelis, Professor	Dr. Eftychios Protopapadakis, Assistant Professor
.....	.....	.....

Angelos Martidis

.....

Περίληψη

Η ανάπτυξη της μηχανικής μάθησης ως κλάδος της Τεχνητής νοημοσύνης αυξάνεται ραγδαία τις τελευταίες δεκαετίες λόγω της επέκτασης των μεγάλων δεδομένων. Η αυξανόμενη διαθεσιμότητα μεγάλων ποσοτήτων δεδομένων έχει επίσης δημιουργήσει απαιτήσεις για πιο αποτελεσματική ανάλυση δεδομένων. Οι αλγόριθμοι βασίζονται σε στατιστικά μοντέλα, έχουν τη δυνατότητα να καταλαβαίνουν μοτίβα και να κάνουν προβλέψεις για τη βελτίωση ποικίλων εφαρμογών. Ένα σύστημα συστάσεων διερευνά την υπολογιστική προσέγγιση που έχει σχεδιαστεί για την πρόβλεψη των επιλογών ενός χρήστη προς ένα αντικείμενο, με βάση την εξέταση των προηγούμενων προτιμήσεων και ενεργειών του χρήστη. Η τεχνική που είναι γνωστή ως συνεργατικό φιλτράρισμα, ανήκει στα συστήματα συστάσεων και στοχεύει να κάνει προτάσεις προτιμήσεων για τις άγνωστες προτιμήσεις, ενός νέου συνόλου χρηστών αναλύοντας τις προτιμήσεις ενός γνωστού συνόλου. Για τη μελέτη του συνεργατικού φιλτραρίσματος, είναι απαραίτητο να προσδιοριστεί η ομοιότητα μεταξύ μιας ομάδας χρηστών και των αντικειμένων, η οποία συνδέεται συχνά με τις συμπεριφορές ενός χρήστη και τον τύπο του αντικειμένου, προκειμένου να γίνουν προτάσεις με βάση τις προτιμήσεις παρόμοιων χρηστών. Αυτή η μεταπτυχιακή διπλωματική εργασία υλοποιεί ένα σύστημα συστάσεων σε γλώσσα προγραμματισμού Python για την πρόβλεψη των κλικ, προσθηκών καλαθιού και παραγγελιών, ενώ ερευνά τα συστήματα συστάσεων και ιδιαίτερα το συλλογικό φιλτράρισμα, για τη βελτίωση των αποτελεσμάτων.

Keywords: Μηχανική Μάθηση, Προγνωστική Μοντελοποίηση, Επιστήμη Δεδομένων, Συστήματα συστάσεων, Συνεργατικό φιλτράρισμα, Σύστημα Συστάσεων Πολλαπλών Στόχων, Ανάκληση, Python

Abstract

The development of machine learning as a branch of Artificial intelligence has been rapidly increasing in recent decades owing to the expansion of big data. The increasing availability of large amounts of data has also created demands for more efficient data analysis. Algorithms that are based on statistical models, can learn patterns and make predictions to improve a variety of applications. A recommender system explores the computational approach designed to predict the choices of a user toward an item, based on an examination of the user's prior preferences and actions. The technique known as collaborative filtering belongs to recommender systems and aims to make preference recommendations for the unknown preferences of a new set of users by analysing the preferences of a known set. For the study of collaborative filtering, it is necessary to determine the similarity between a group of users and items, which is often associated with a user's behaviour and the type of the item, in order to make suggestions based on the preferences of similar users. This master's thesis uses a Python recommender system to predict clicks, cart adds, and orders while researching recommender systems, in particular collaborative filtering, to enhance the outcomes.

Keywords: Machine Learning, Predictive Modelling, Data Science, Recommender Systems, Collaborative Filtering, Multi-Objective Recommender System, Weighted Recall, Python

Acknowledgments

First of all, I would like to express my deepest gratitude to my Professor and thesis supervisor Mr. Ioannis Refanidis, for his guidance, support, and encouragement throughout the entire process. His insightful comments and constructive feedback helped me to shape my research and improve my knowledge and skills in the field of Artificial Intelligence. I am truly grateful for his dedication and enthusiasm for my project.

I would also like to thank my family and my friends for their continuous support and encouragement. A special acknowledgment goes to my brother Vasilis, who offered exceptional guidance for my thesis implementation. Their love and understanding have been a source of motivation throughout my academic path.

Angelos Martidis

Table of Contents

List of Figures

List of Tables

Symbols

RS Recommendation systems, Recommender systems

CF Collaborative filtering

CB Content-based

UBCF User-Based Collaborative Filtering

IBCF Item-based Collaborative Filtering

1 Introduction

1.1 The rise of data and its role in E-Commerce

Customers today have a variety of needs, they each have their own set of priorities and preferences that can affect their choices. Finances, personal interests, lifestyle, social status and cultural background are some of the main factors that might significantly impact consumer behaviour and decisions. Figure 1.1 [1]

However, the time when it is necessary to accomplish a goal, address a particular issue, or try to solve a problem, it is also crucial for what a customer could need or remember to get from a store or website online.

Understanding and addressing the numerous needs of customers is vital for companies in order to be able to satisfy them and keep their trust. Meeting multiple needs, instead of only spreading their product variety, allows companies to approach a wider market and increase their potential by offering products and services to new customers outside their standard customer base.

E-commerce or electronic commerce has allowed them to do that. Businesses of any size that have their base or storage in one or multiple places perform shipments to selected regions or even worldwide. As the investment in physical storefronts for operations is unnecessary, overhead costs become lower. As a result, many small firms expand and develop faster than otherwise. Companies sell their services and products online to many people without them having to move to physical stores. Online shopping improved the process of making purchases by allowing customers to make purchases at any time without restrictions from anywhere and by evaluating the costs and goods offered by various merchants. Transactions are faster and more efficient, and many corporate activities, such as inventory management and customer support, may be automated. [2]

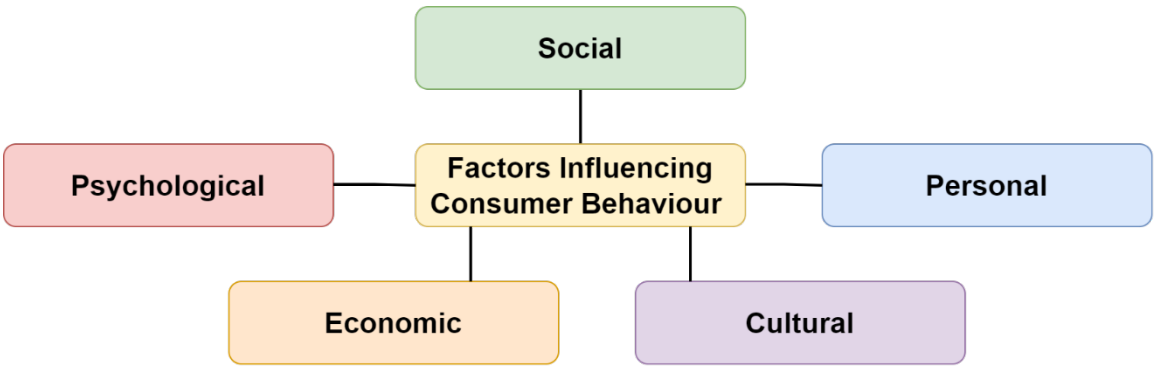


Figure 1.1 Main factors influencing consumer behaviour

1.2 Effective Marketing through Digital Means

Marketing aims to reach customers and increase their awareness about an option and influence them to take a decision. The concept of affecting this decision has changed over the years. Nowadays marketing became cost-effective, works mainly through emails and social media, providing flexibility to reach their target audience way quicker.[3] Also, it enables the ability to collect consumer data and analytics to track the effectiveness of their advertising efforts. The acquired data can be utilized to examine various areas of their marketing performance and product development, gain insights and create tactics for the optimization of their sales and campaigns for better results. The user's experience can also be improved by personalizing their preferences to the businesses' communication, offers, products, and services. These developments have resulted in upgrading the items with massive amounts of data that consumers must be aware of, to make their decisions, depending on their necessities.

1.3 Optimizing the E-Commerce Experience through Recommendation Systems

For a customer, managing all this information in order to purchase a product or decide among various possibilities could be an immensely challenging procedure. [4]Recommendation systems are methods that use machine learning algorithms and mathematical formulas to contribute to the issue of managing all this data for the user, providing a better experience for them.

Recommender systems (RS) are programming tools that offer recommendations depending on data such as user behaviour and preferences. New recommendations are produced, even if it is an item to purchase, music to listen to, or food to order. The term item is typically used to describe the predicted suggestions. Depending on the procedure for the decision-making the algorithm tries to find new items for the users. The structure and the way a recommender system is designed to perform predictions varies, and is defined by the exact type of item it tries to predict. The approach utilized to create the suggestion for the consumer is adjusted to deliver successful choices for that particular item and does not apply the same way to every type of item.[5, pp. 1–35]

Suggestions that are not based on the user's personal data are also considered useful and successful options for an RS if the user’s data are not enough. The reason is that these

suggestions can be the best-selling products on a selling website, the most recommended movies by popular directors on a streaming platform, or popular books of the year. However, an RS does not focus on these options and scopes on better solutions.[4]

The impact and the actual number of practical use cases of recommended systems in the commercial sector are very substantial. An e-commerce recommender engine assists users in selecting items to purchase. Their use in e-commerce websites is very common as they increase the marketing of the company. A popular website such as Amazon.com processes a per-customer personalized store with suggestions based on their data using a RS.[6] For many businesses, a big percentage of the profits is attributable to the systems they use.

1.4 Benefits of using recommender systems in E-Commerce

- Businesses identify cross-selling and upselling opportunities.
- Increasing conversion and click-through rates.
- Assisting in discoverability of new items (or totally hidden)
- Reduce the rate of bounce and cart abandonment rate.
- Improving decision making and increasing client satisfaction.
- Personalizing offers which affect the time and the quality of the visit.

1.5 The purpose of the research

The purpose of the research of this master thesis is to present a comprehensive overview of recommender systems, explain the basic categories used in collaborative filtering, and create a multi-objective recommendation system given the data of an enormous number of previous buyers to predict the choices of the next ones. More specifically, the purpose of the study is to develop a recommendation system that can anticipate the next moves of a new set of buyers, having available a set with the clicks, additions to the cart, and purchases of previous users as a starting point. This new set of users contains fewer data meaning fewer options than those on the first provided for data analysis. The data utilized for this project is offered by OTTO online shop on a Kaggle's featured prediction competition. The goal is to help online retailers find more relevant items to recommend to their clients based on their real-time behaviour. An implementation

of a collaborative filtering recommendation algorithm was created using the Python programming language and attended the competition.

1.6 Basic Machine Learning Terms

The definition of some basic machine learning terms is given bellow:

1.6.1 Machine Learning:

Machine learning is a sub-field of computer science and more specifically Artificial Intelligence. This field focuses on producing algorithms that improve the accuracy of their operation and results by mimicking the human learning method. Data is collected and used to develop statistical based models or algorithms for the purpose of analysing and solving practical problems. Systems that use machine learning develop the capacity to autonomously learn and improve from experience without being explicitly coded. [7] Input data train the model and after the trained model makes predictions for new data.

1.6.2 Data (or samples/examples) :

Labelled data comprise a target variable (also known as dependent variable or label) which is the variable that the model attempts to predict.

Unlabelled data, on the other hand, does not comprise the requested predicted value.

The dataset's examples are often grouped into the following three categories: training set, validation set, test set

1.6.2.1 Training Data(training set):

Data used to train the model. During the training process, the ideal weights are determined for a better prediction result.

1.6.2.2 Validation Data(validation set):

Data used for the evaluation of the model, to fit on the training dataset or to improve parameters. Usually evaluation of the trained model is checked several times against the validation set.

1.6.2.3 Testing Data(test set):

Testing data ensures a neutral evaluation after the completion of training the model. It enables comparison of predicted outputs by using this set. This assesses the model's ability to generalize to new, unseen data.

1.6.3 Categories of learning models:

1.6.3.1 Supervised:

These models use as input a known dataset (training data set) that is labelled, and produce the predictions for new, unprocessed data (test data set) using a selected function. Targets are predicted for the input data set in the model. Any errors in the model's learning process are corrected through a procedure called back-propagation, which means that the output of the model is getting compared to the ground truth label or the actual expected output. Supervised models are frequently separated into Regression and Classification problems. In Regression, the aim is for the relationship between independent variables and a dependent variable to be modelled and predict the dependent variable. In Classification, the purpose is to assign new data to one of several predefined categories or classes based on the input features.

1.6.3.2 Semi-supervised:

These models use as input both labelled and unlabelled training data together in order to produce better learning accuracy.

1.6.3.3 Unsupervised:

Unsupervised models use as input an unlabelled data set to find a hidden structure or commonalities in them. A function that describes these common forms in the unlabelled data can find conclusions for them. Unsupervised models are frequently separated into Clustering and Association problems. In Clustering, data is separated into a number of clusters and grouped based on similarities without a sufficient meaning for each cluster. In Association, patterns and relationships are found in the data by defining certain rules.

1.6.3.4 Reinforcement:

Reinforcement learning explores how software agents can perform actions in an environment to maximize a reward. The environment is described with states that the agent can receive as input while performing various of actions. The decisions made can lead to more or fewer rewards per step, but can also change the location of the agent which will affect the full reward. A function known as the policy is takes as input the feature vector of a state and produces an ideal move to perform in that phase. This function is the result of the reinforcement learning algorithm. Reinforcement learning methods do not require a perfect mathematical formula and are effective in real-world applications such as autonomous vehicles and games. They can function even when exact models are not feasible.

1.7 Thesis Structure

Chapter 1 A brief overview of the importance of recommended systems in e-commerce and their impact on the customer experience. The objectives, the problem that the thesis is trying to solve, and some key terms for ML and E-Commerce are presented, along with the outline of the thesis.

Chapter 2 The theoretical background of recommender systems, including their history, development, and various types, is presented. The concept of collaborative filtering, its application in recommender systems, and the basic approaches are described.

Chapter 3 The definition of the problem that the thesis aims to solve in detail, including background, purpose, dataset description and its characteristics, evaluation method, and timeline.

Chapter 4 Data analysis, methodology used in the implementation including data pre-processing steps, similarity measurement, model selection, and both comparison and discussion of the results.

Chapter 5 Summary and conclusion, discussion of the limitations and challenges of the study, recommendations for future work, and improvement.

2 Theoretical Background.

2.1 Recommendation systems

Recommendation systems (or recommender systems or RS), can be applied in many areas and have very good results and this is also the reason that contributes to their rapid development. As it was mentioned in the first chapter, recommendation systems are commonly used in e-commerce websites but have also been established in a wide range of other sectors, such as, media sharing or streaming, video games or even location-based applications. By accessing a service-related website, a user and potential customer understands their data is used. The algorithm uses these data, in order to present them suggestions closer to their interests. In other words it is an information filtering system that its purpose is to deal with input and deliver interesting suggestions as output.

The factors that will determine the final result vary and play important role on the outcome. These factors depending on the problem being addressed can be: existing information (data), algorithms, methods, database sparsity and scalability, system efficiency and quality (evaluation metric).[8] A classic form of a recommendation system is illustrated in the figure. Figure 2.1.

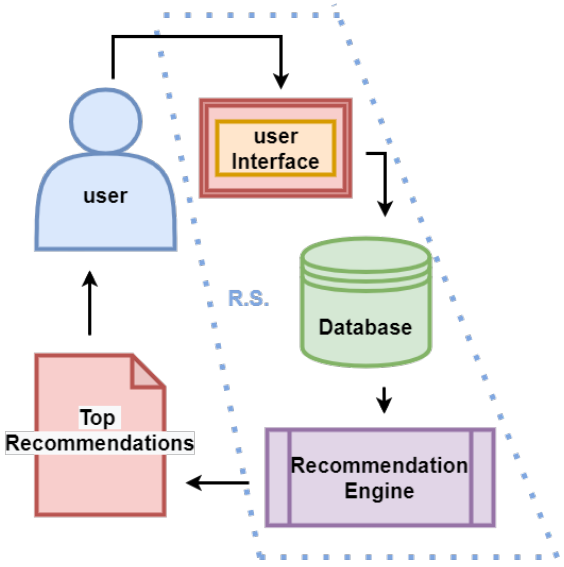


Figure 2.1 Standard recommender system execution

There are two basic approaches on the concept of filtering the information into getting expected results, Content-based filtering and Collaborative filtering, while hybrid is a combination of the two of them. In the next figure recommender systems and their categories are shown Figure 2.2.

- A content-based filtering system chooses items based on the user's preferences and the items' content.
- A collaborative filtering system is based on the association between individuals with common interests.
- A hybrid system, as its name suggests, combines these different aspects to create a more wide variety of settings in order to solve the problems.

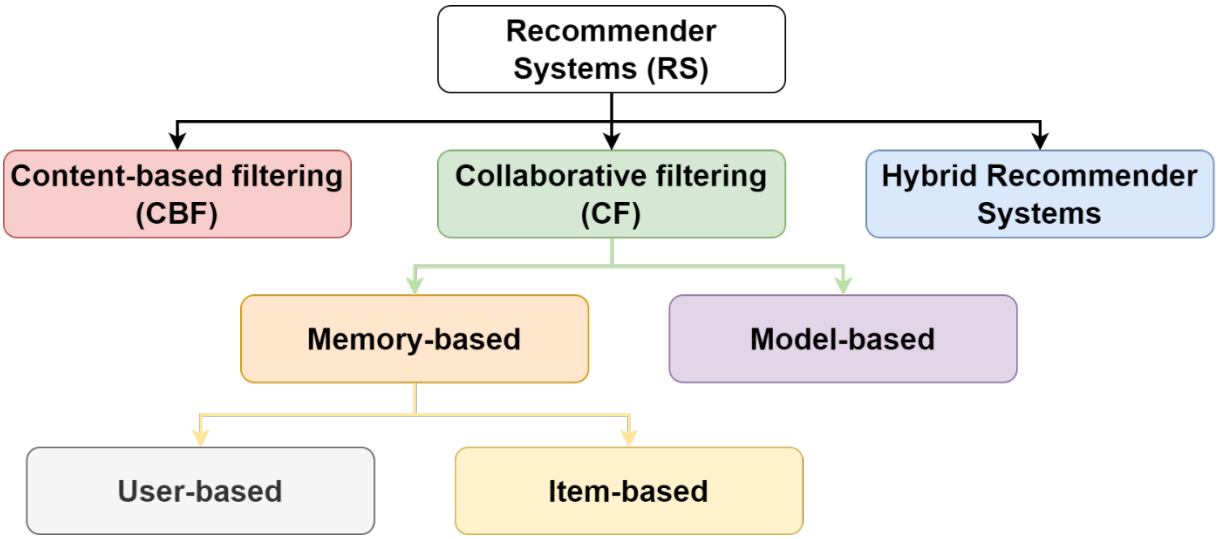


Figure 2.2 Types of Recommender Systems

2.2 History

In Duke University around the 1970s, while sharing text was possible it could be classified in subcategories without involving the preferences of the individuals.[9, pp. 175–186]. In 1979 Elaine Rich managed to accomplish a book recommendation approach that provided recommendations by categorizing a person’s interest to a specific stereotype. [10, pp. 231–259]Another remarkable approach is “Digital bookshelf” of Jussi Karlgren in Columbia University, who was also the first to define them as RS.[11]

2.3 Explicit and Implicit data

Data on user preferences for specific items are processed by recommender systems. This information can be gathered either explicitly, through user evaluations of products, or implicitly. Recommender systems employ this data to forecast and recommend things to users that they are likely to want. Explicit data is called the type of data that describes something that is clearly stated in a direct way. This type of information cannot be misinterpreted. For example, when a user clicks the like button or the heart respectively on YouTube and Spotify platforms, or has added the track to a playlist, this type is explicit. Implicit data is called the type of data that describes something that is not clearly expressed but can be implied. An example of implicit data is when Spotify constructs and suggests to the user a playlist with similar songs based on his listening behaviour. Spotify has no clear statement of the user's choices but it suggests songs based on previous choices.

2.4 Utility Matrix, Statement Of the problem

This table is constructed during the recommendation system's design to distribute data while developing a solution to the problem. Typically, a standard recommendation system model consists of two groups users and items, where users have particular preferences for some of them. In this table users' values correspond to the objects they use or like, thus obtaining a measure of liking of each object by each user. It is assumed that the data of preference for the users are not known so the table is sporadic. The goal of this process is not to fill out all the missing data but to find high values - meaning high liking in order to make the appropriate recommendations.

On the following table, assume that columns are the items represented by capital I and a lower letter, while rows are the users that rated these items, represented by numbers. Ratings are taking values form 1 to 5. Table 1 Not all items are rated and this is quite normal for a dataset.

	Utility Matrix					
	I-a	**I-b**	**I-c**	**I-d**	**I-e**	**I-f**
user 1	3		1	5		
user 2	4	4			5	
user 3	1		3			2
user 4	3			4	5	
user 5		3			3	
user 6	1		2			

Table 1 Example of Utility Matrix

Using this table, the goal is the prediction for non-rated value. Table 1 This can happen by taking in consideration other attributes of an item such as the creator/brand name, the material of the item, or the price. Basically a similarity over a characteristic of items can be the key measure or even similar users' behaviour. i.e. if user 3, user 6 rated both I-a and I-c with a low value, and if I-b has similar characteristics chances are that both users will rate I-b in a similar way.

2.5 Long-Tail

In an e-commerce company, the term long-tail refers to the distribution of the items with the most demand. This term reveals how important the recommendation systems are because simply the availability of some products, is not enough in physical stores instead of multiple available options in online ones. Recommendation systems help the user to discover options of items that would not have found otherwise. The x-axis on the graph demonstrates the items purchased due to popularity whereas the y-axis demonstrates popularity in regards to ratings. Figure 2.3 The items that may not be as popular or are new can be found in the "Long Tail" section of the graph, whereas those in high demand are typically located on the right side and are deemed to be efficient and profitable. Many successful companies, such as Google, eBay, Yahoo, and Amazon, have utilized the long tail concept in their business strategies to target specific audiences and deliver a wide range of products and services to improve their profits.[12]

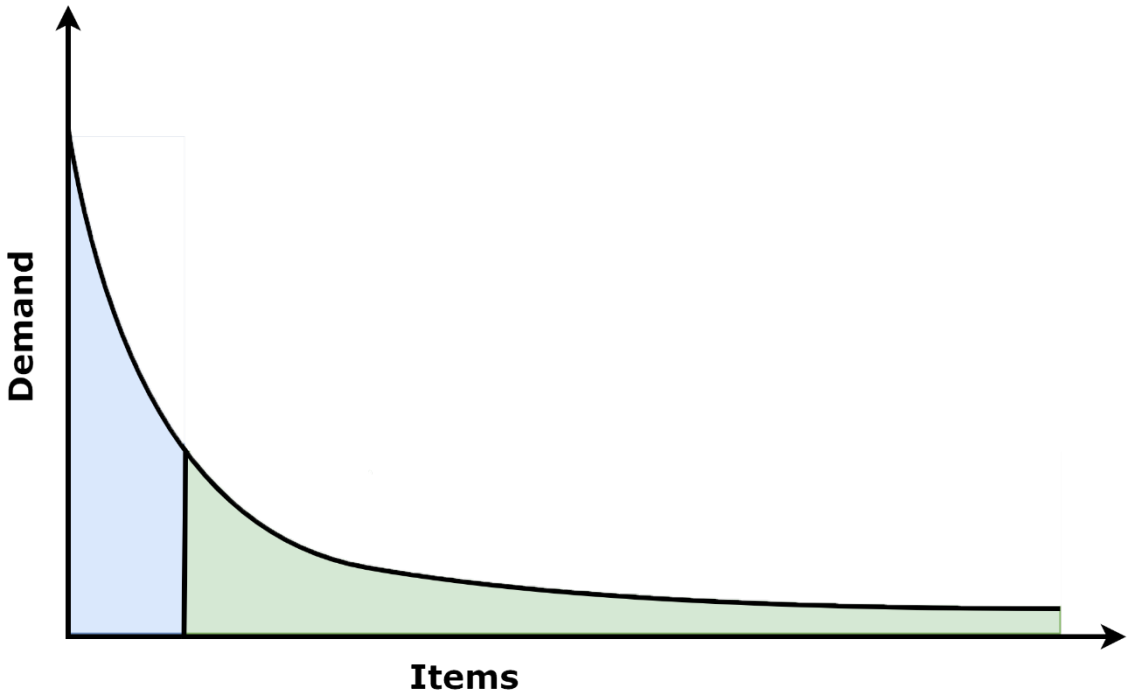


Figure 2.3 Long Tail

To the left The cutoff is selected in this illustration such that the volumes of both regions are equal. (blue) are the few that dominate, and to the right (green) is the long tail.

It is important to take under consideration the place of the starting point or threshold of the diagram while researching a recommendation system algorithm. By shifting this point towards the right on the graph, there can be an increase in the diversity of recommendations that the system will provide. This point can also affect the sparsity of the algorithm which is another issue, as the items on the tail are not rated by many and this can result in bad predictions.

2.6 Content-based filtering

When the user's profile is unknown and there is no data specified for the user, but the only data available on the problem are those of the items, then content-based filtering suits the most. In these systems, the term content is referring to the description of the items. Despite the fact that user data are unknown, this method can suggest items by creating a user profile with old liked items and suggesting similar options.

More specifically old options of the user are compared with all possible picks to produce the recommendations. The profile is generated based on the preference of the user and on his interaction with the RS. The words that describe the specific item are of considerable importance in the method because it is recommended depending on the count of similarity [13].

The individually rated content vectors can create a vector consisting of item features which after that is used but weighted to create the profile of users. In case user profiles and item data are known in order to determine if a user prefers an item both heuristic methods or classification algorithms are welcomed. The mechanism of a CBF system is shown in the figure. Figure 2.4 [14, pp. 163–170] [15, pp. 714–720]

Figure 2.4 The Content-Based Filtering Process

To demonstrate content-based filtering, a simple example follows. Assume that features are engineered for a movie database and the items are the movies. A feature matrix where each row represents a movie and each column represents a feature is shown in the figure. Figure 2.5 Features in this case could be a genre for the description of the movie or a set of keywords(such as horror, romance, comedy), the movie director. When a feature exists on the matrix, then its value is non-zero (binary matrix). Some user-related features may be offered directly by the user, for example, if they watch the filmography of a certain actor. Furthermore, a similarity metric must be used to recommend items relevant to this user. The system should score each feasible prediction item according to the metric. The

recommendations are specific to this user, as the model did not use any information about other users.[16]

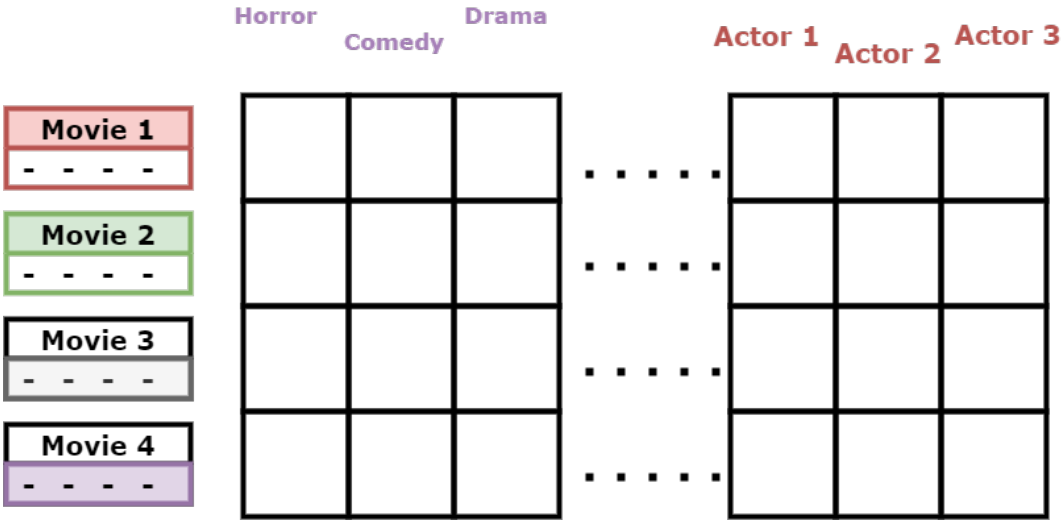


Figure 2.5 Movie example CBF

2.7 Collaborative filtering

Collaborative filtering is a method of filtering information concerning preferences of an individual user from the common set of users (based on similar profiles). The data are obtained from many profiles but the predicted results are about the particular user. Traditionally CBF algorithms are used on big data sets, and may require users' active engagement, way to define users' interests, and algorithms capable of matching persons with similar preferences.[17, pp. 291–324], [18, pp. 154–164]

It is related to individual's choices but, it is based on the idea that if a different person has shared choices, then in another situation the latter person's choices will be shared same with the former.

Typically, the application of collaborative filtering is achieved in three steps. Figure 2.6 First, the importance of each user in relation to the active user is determined. A suitable selection of users is then chosen for the purpose of prediction. Finally, a combination of the ratings of these users is used to calculate the new prediction after the ratings normalization of the selected group of users. [18, pp. 154–164], [19]

Normalization is accomplished by taking a user's average score and then subtracting it from each of the provided scores. As a result, the scores are further divided into groups that are similar according to their scores. By standardizing the data, clusters of users with same scores to similar items are formed, which are then utilized to recommend items to users.

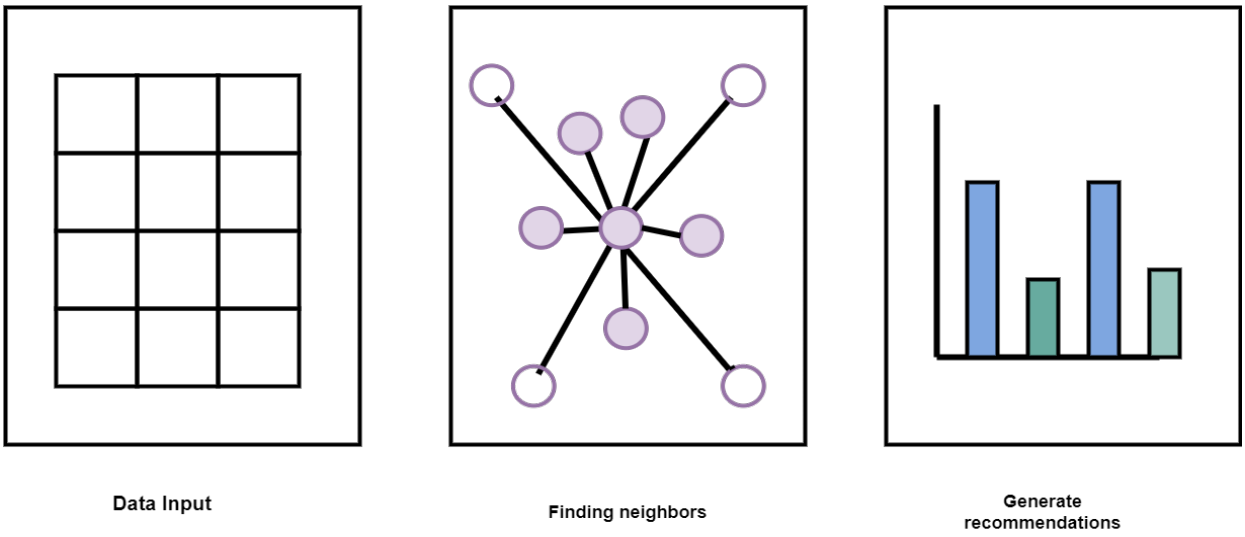


Figure 2.6 The Collaborative Filtering Process

The basic approaches are of two kinds: User-Based Collaborative Filtering(UBCF) and Item-based collaborative filtering(IBCF).

2.7.1 User-Based Collaborative Filtering (UBCF)

User-Based Collaborative Filtering is called the type of collaborative filtering in which a target user is compared with others to find similar behaviours, in order to provide recommendations. A classic example is that of Netflix or You-tube when the history of two users is similar, then the system may recommend media that the one has viewed or liked to the other. The figure below visualizes the user-based collaborative filtering Figure 2.7 [20, pp. 153–158]

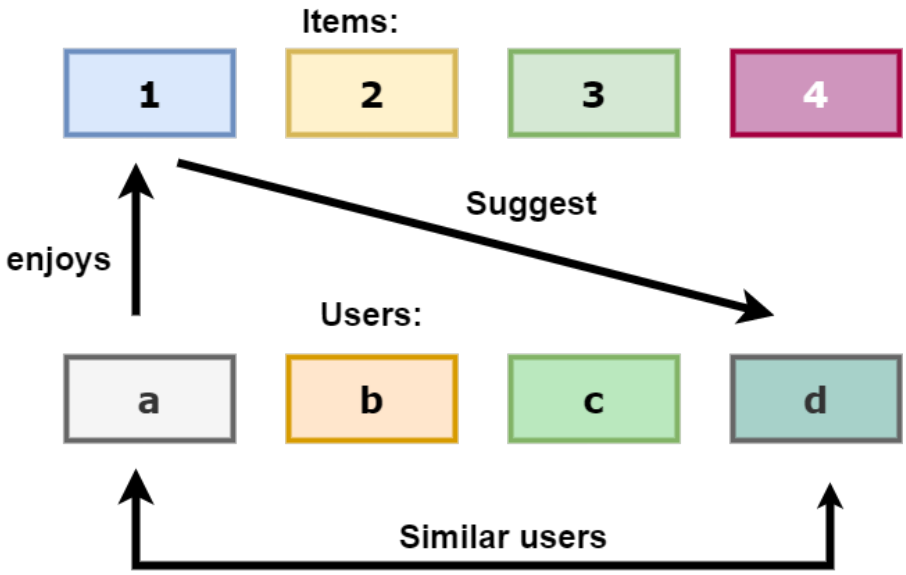


Figure 2.7 User-Based Collaborative Filtering

2.7.2 Item-based Collaborative Filtering (IBCF)

Item-based Collaborative Filtering is called the type of collaborative filtering in which items that are frequently consumed or purchased together are identified, by analyzing the actions of the users. For example, if a user purchased a book, then the system knows the user's purchases and identifies other items that are frequently purchased along with the specific book. It may notice that many readers chose to purchase a magazine too, and then make that suggestion to this user. Item-based Collaborative Filtering is illustrated in the figure. Figure 2.8 [20, pp. 153–158]

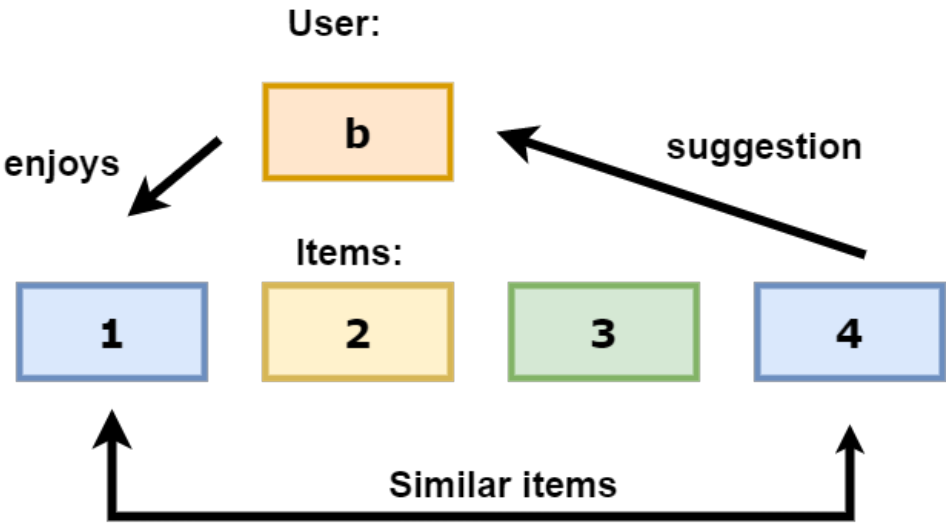


Figure 2.8 Item-based Collaborative Filtering

To demonstrate collaborative filtering, a simple example follows. Assume that a recommendation system is built for movie database as before. In this scenario, rows represent viewers and columns represent the movies. Whenever a user enters the database, he must receive a recommendations. These suggestions should be based on similarities to movies previously liked by the viewer as well as movies that similar viewers have enjoyed. Note that the quantification of a viewer's liking of a movie is defined using numerical values and the preference is defined by watching it. Features in this case could be the description of each movie in the database, as shown in the figure. Figure 2.9

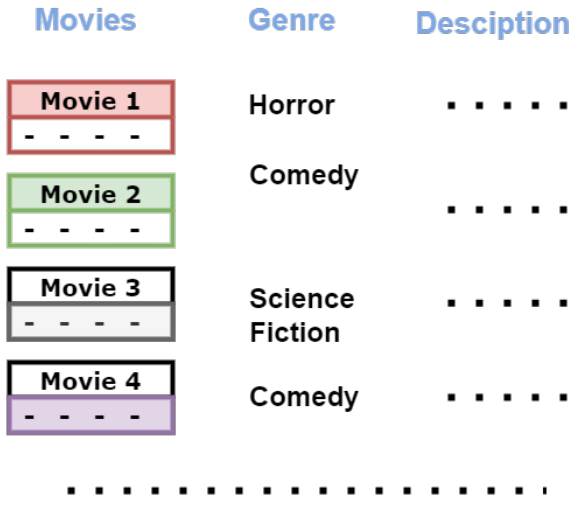


Figure 2.9 Movie database

Considering whether a movie is a comedy or not, an indicator -1 or 1 can be relatively assigned. The same goes for the viewers and their interest in this genre of movies. So -1 for not enjoying it and 1 for enjoying the movie. So in order to recommend a movie to the viewer, the movie must also be of the comedy genre and to their liking, i.e. (it must have a value of 1 on both user and type indicator). Visually, this example is shown in the following figure. Figure 2.10

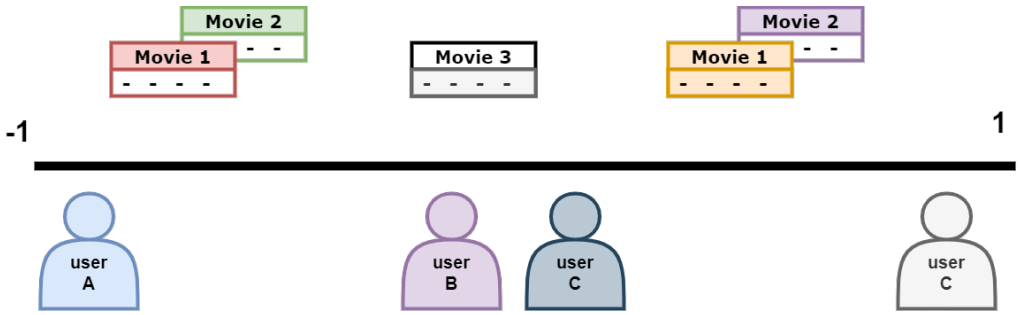


Figure 2.10 Movie example for CF

In practice, more features are applied in order to represent and explain various of characteristics for items and users and also to achieve different results concerning our predictions. The advantage of collaborative filtering models is that the extracted features can also be acquired automatically.[16]

2.7.3 Model-based collaborative filtering:

Model-based methods' goal is to predict the ratings on unseen items by modeling past ratings of the users. To achieve this, the system builds a model of the user's preferences based on their past ratings and uses that model to make predictions for new items. This model is built using machine learning algorithms.[21] The concept is that the patterns and associations found in previous ratings can be utilized to make predictions about the user's preferences for new items they have not yet seen. As new data enter the system, it can be used to retrain it the model. Some algorithms that are used in these methods are matrix factorization algorithms, singular value decomposition, principal component analysis, and neural networks[22].

2.7.4 Memory-based collaborative filtering:

This is a subcategory of user-based and item-based collaborative filtering. These methods are using historical data to find the recommendation meaning that the data of user-item interactions are stored in the system's memory. A similarity measure between users or items is calculated in real-time, to find most similar to recommend. There are many ways to represent the similarity of a user or an item, with cosine-similarity and correlation similarity being the most popular. Memory-based algorithms are not always efficient and fast, thus they are commonly replaced by model-based. [23]

2.7.5 Neighborhood-based collaborative filtering:

This is a subcategory of user-based and item-based collaborative filtering. These methods are concerned with the relationships that exist between items or, between users. Based on ratings of similar items by the same user, an item-based method predicts a person's preference for an item. More specifically, they are using a subset of similar users or items that is similar to a target user to create recommendations. This approach determines how similar two users or items are.

The similarity is measured with Pearson correlation and vector cosine based similarity. [24]

Pearson correlation:

$$\text{simil}\,(x,y) = \frac{\sum_{i\in I_{xy}} \left(r_{x,i} - \bar{r_x}\right)\left(r_{y,i} - \bar{r_y}\right)}{\sqrt{\sum_{i\in I_{xy}} \left(r_{x,i} - \bar{r_x}\right)^2}\sqrt{\sum_{i\in I_{xy}} \left(r_{y,i} - \bar{r_y}\right)^2}}$$

Vector cosine based similarity:

$$\text{simil}\,(x,y) = \cos\,(\vec{x},\vec{y}) = \frac{\vec{x}\cdot\vec{y}}{\|\,\vec{x}\,\|\times\|\,\vec{y}\,\|} = \frac{\sum_{i\in I_{xy}} r_{x,i}r_{y,i}}{\sqrt{\sum_{i\in I_x} r_{x,i}^2}\sqrt{\sum_{i\in I_y} r_{y,i}^2}}$$

where I_{xy} is the set of items rated by both user *x* and user *y*.

Because of their simplicity, efficiency, and capacity to deliver accurate and individualized recommendations, neighborhood-based approaches have grown in popularity. [5] [22]

2.7.6 Hybrid collaborative filtering:

Companies like Amazon, Netflix, and Google use a combination of various technologies and techniques to drive their results in a better outcome. This sort of recommendation system combines content and collaborative recommendation system algorithms of both memory-based or the model-based approaches. These models have the ability to perform better as they overcome some of the weaknesses of the previous ones, but have higher difficulty to be applied. For example Netflix does that by integrating the viewer's viewing and searching data contexts with existing records(past data). Google on the other hand builds user profiles based on their click behaviour to generate news feed. [25][26, pp. 81–104] [27, p. 271] [27]

2.8 Common Problems

For recommender systems and specifically collaborative filtering algorithms, the desired recommendation is the outcome of a good performance, which can be affected by other factors as well. In this subsection common problems of collaborative filtering algorithms are presented.

2.8.1 Cold start - New User or New Item

The user's activity must include many ratings for the recommendation system to understand a profile of a user, otherwise if the ratings are not enough the recommendations will be incorrect. Correspondingly on the object side if a new object is not rated by a specified number of users to be considered sufficient then the recommendation system does not have the ability to put it as a suggestion to the user. [28], [29]

2.8.2 Data sparsity

A similar problem is created when the volume of data managed by the recommendation system is very large. There must be an availability to a number of users who have participated in item ratings in order for the system to function properly. It is also important that there are many users who rate similar items otherwise they cannot be recommended. [28]

2.8.3 Scalability

Another major challenge collaborative filtering algorithms face is the handling of massive datasets, which results in a correspondingly large number of users and items. Considering that such a system must act in seconds by making access of all these entries, this also means that the complexity of this task is also massive and requires improved resources to solve.[30]

2.9 Key differences

Key differences of both Content-Based and Collaborative filtering are following. An illustration of each procedure for both Content-Based Filtering and Collaborative Filtering method is presented. Figure 2.11 Also a table describing the main differences (on focus, targets, requirements, data, results and additional problems) for these two methods is demonstrated. Table 2

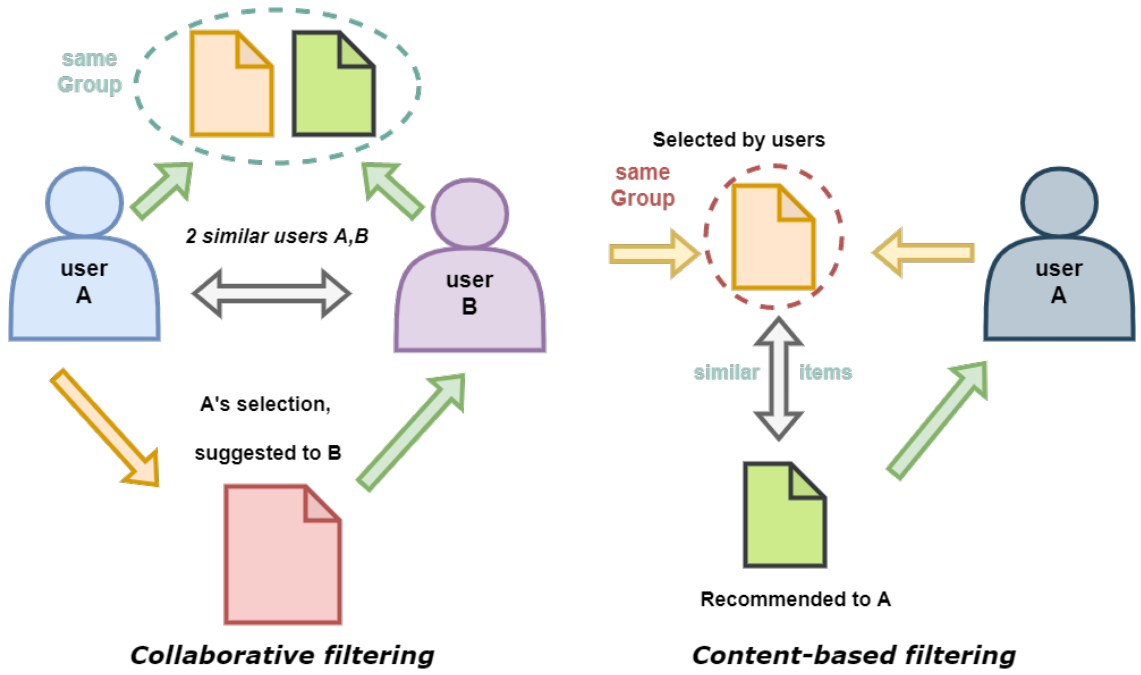


Figure 2.11 CF and CBF processes

Content-Based Filtering	Collaborative Filtering
What is the focus?	
Focuses on the content features	Focuses on the user's preference
Targets	
Types or characteristics of items a user selects	Patterns of users selecting items
Requirements	
User's profile and content profile is also required	User's profile is required
How it works?	
Description and attributes of items	Users that interracted with an item
Data	
Needs less amount of data Existing data knowledge is required	Needs large amounts of data Embedding data are automatically learned - data knowledge not required
Results	
Results based on a user's existing choices Captures the specific interests of a user More limited on recommendations More predictable (since it is based on specific attributes)	Results based on ratings of many users Can provide users new interests Possibility to get recommendations of unseen items More unpredictable
Problems	
Difficulty on content analysis and extracting data Biased towards popular items - more data	Even users with same interests can not necessarily prefer the same item. Hard to predict recommendation for unusual behaviours' of users

Table 2 Differences of Content-Based Filtering and Collaborative Filtering

3 Problem

3.1 About OTTO

OTTO (GmbH & Co KG) is a successful e-commerce company and belongs to multi-national Otto Group, which also subsidizes Crate & Barrel (USA) and 3 Suisses (France), but at the beginning started as a retailer based in Hamburg of Germany. The company was established in 1949 when the business owner launched a catalogue with 28 pairs of shoes and made ordering from home available. Over the years, OTTO expanded, added more products, and became the largest mail-order company worldwide by the mid-1980s. In 1995 was one of the first companies that decided to adapt to the digital age and made a difference by launching a website that supported fashion and lifestyle.[31]

Nowadays, a diverse selection of products is available on the website, including those for clothing, accessories, technology, home and living, sports, and leisure equipment. The company is committed to sustainability, with a focus on environmentally friendly and socially responsible practices in its areas. OTTO employs more than 7,000 employees and offers services to more than 30 million individuals in Germany and other nations. It has over 4000 OTTO Market Partners and provides over 10 million products online to more than 11,5 million active customers. Daily the average number of qualified visits per day corresponds to 2.89 million.[32]

One of the company's distinguishing qualities is its user-friendly website, which makes it simple for clients to navigate and purchase products online. Free delivery and returns are also offered, making the shopping experience convenient and faster.

In addition, OTTO as mentioned provides the service of recommending products to its customers. However, these recommendations are not random but targeted. OTTO makes use of advanced data analysis and algorithms to deliver its clients, personalized recommendations based on their preferences and previous purchases. OTTO has a huge item database and gathers data from its clients every day which is used to recommend different items to them or to new customers. The company has a related group that focuses on investments that can be made in the technology sector to increase profit. AI being of the most sought-after areas which the group applies to various business sectors. Bonprix which is an individual company of the Otto Group has created its own AI prediction model that orders items into categories and offers better quality service by displaying at the top

of the store page, the most popular items which are currently available. This model can also detect which combinations of different attributes are most promising and provides the company the ability to understand new insights into their customer needs. [33]

3.2 The Competition

Kaggle is an online famous competition platform that offers data scientists and machine learning practitioners the opportunity to find and publish datasets of problems allowing them to build models, and solve but also participate in competitions for money rewards. Kaggle platform launched in 2010 and was bought by Google in 2017.

The Otto Group's challenge on Kaggle is about the development of a recommendation system algorithm for their customers which will help to increase their sales and improve the shopping experience. As there is no model in e-commerce that can optimize numerous objectives at the same time. Their goal is to achieve that. More specifically the model should be able to predict the next move of a customer, either it is a click, an add to cart or an order prediction. The objective is the creation of a multi-objective recommender system based on past events of a user session for the prediction of e-commerce clicks, cart additions, and orders.

This competition started in November 2022 and ended at the end of January 2023. It was open to anyone from any place that complied with the given rules. To enter the competition registration was required prior to the entry deadline. The aim was the promotion of data science and the type was clearly skill-based. Also in its terms were included awards of $ 30,000. The prizes were awarded to whichever system performed the best score based on the evaluation metric and the merits of the data science models submitted. The first award was $15,000, followed by $ 10,000 for second place, and $ 5,000 for third.

The public leaderboard on the competition website displayed the current position throughout the competition period. Subject to adherence to the rules, the possible winners were chosen purely based on leaderboard positioning on the private leaderboard. Both the public and private leaderboards were built using test sets that were made available to the general audience. Participants could participate either individually or in teams. The maximum team size was five participants per team. Daily entries allowed were 5, while at the end of the event duration, each contestant could choose two to be judged.

In total 3,350 people in 2,615 teams took participation in the event representing different countries.

3.3 Data

Three data files were provided: two (.jsonl) files with test and train data, and one (.csv) file for the sample submission.

- Test data: test.jsonl - 402.09MB: Contains the sessions for which the predictions will be made.
- Train data: train.jsonl - 11.31GB: The training data includes e-commerce session details.
- sample submission: A sample file in the required format.

For data comprehension, the task of the required prediction should be stated. Sessions truncated in time order are included in the test data set. Our goal is to predict the next moves after the point of truncation. For each test session and each event type our task is to predict the aid values involved on the next move.

The submission file sent to Kaggle should be a (.csv) file, has the following format where session is the label of the session in the test set, type is the type of the event and labels are the predicted product codes which should be space delimited. We can predict up to 20 product codes per row.

session_type,labels
12906577_clicks,135193 129431 119318 ...
12906577_carts,135193 129431 119318 ...
12906577_orders,135193 129431 119318 ...
12906578_clicks, 135193 129431 119318 ...
etc.

Train data has the following format:

{"session": SESSION_ID, "events": [{ "aid": PRODUCT_CODE, "ts": TIMESTAMP, "type": "clicks/carts/orders" },…}]}

- session - the unique session id (number of session)
- events - the time ordered sequence of events in the session
    - aid - the article id (product code) of the associated event
    - ts - the Unix timestamp of the event
    - type - the event type, (clicks, added to cart or ordered during the session)

Each session contains a random number of events of different type in chronological order Figure 3.1 .

Test data **test.jsonl** contains truncated session data in the same format.


```
{"session":12899779,"events":[{"aid":59625,"ts":1661724000278,"type":"clicks"}]}
{"session":12899780,"events":[{"aid":1142000,"ts":1661724000378,"type":"clicks"},{"aid":582732,"ts":1661
{"session":12899781,"events":[{"aid":141736,"ts":1661724000559,"type":"clicks"},{"aid":199008,"ts":16617
{"session":12899782,"events":[{"aid":1669402,"ts":1661724000568,"type":"clicks"},{"aid":1494780,"ts":166
{"session":12899783,"events":[{"aid":255297,"ts":1661724000572,"type":"clicks"},{"aid":1114789,"ts":1661
{"session":12899784,"events":[{"aid":1036375,"ts":1661724000604,"type":"clicks"},{"aid":1269952,"ts":166
{"session":12899785,"events":[{"aid":1784451,"ts":1661724000809,"type":"clicks"},{"aid":1169631,"ts":166
{"session":12899786,"events":[{"aid":955252,"ts":1661724001174,"type":"clicks"},{"aid":955252,"ts":16617
{"session":12899787,"events":[{"aid":1682750,"ts":1661724001617,"type":"clicks"},{"aid":1682750,"ts":166
{"session":12899788,"events":[{"aid":245131,"ts":1661724001619,"type":"clicks"},{"aid":39846,"ts":166172
{"session":12899789,"events":[{"aid":525156,"ts":1661724002025,"type":"clicks"},{"aid":631398,"ts":16617
{"session":12899790,"events":[{"aid":1219653,"ts":1661724002843,"type":"carts"},{"aid":1830166,"ts":1661
{"session":12899791,"events":[{"aid":915175,"ts":1661724002984,"type":"clicks"},{"aid":915175,"ts":16617
{"session":12899792,"events":[{"aid":1537160,"ts":1661724003149,"type":"clicks"},{"aid":1537160,"ts":166
{"session":12899793,"events":[{"aid":1181781,"ts":1661724003444,"type":"clicks"},{"aid":1585431,"ts":166
{"session":12899794,"events":[{"aid":1746099,"ts":1661724003737,"type":"clicks"},{"aid":1242729,"ts":166
{"session":12899795,"events":[{"aid":207754,"ts":1661724003946,"type":"clicks"},{"aid":641250,"ts":16618
{"session":12899796,"events":[{"aid":4503,"ts":1661724004002,"type":"clicks"}]}
{"session":12899797,"events":[{"aid":1335784,"ts":1661724004003,"type":"clicks"},{"aid":156020,"ts":1661
{"session":12899798,"events":[{"aid":379440,"ts":1661724004013,"type":"clicks"},{"aid":1441157,"ts":1661
```


Train set:

	session	aid	ts	type
0	0	1517085	1659304800	0
1	0	1563459	1659304904	0
2	0	1309446	1659367439	0
3	0	16246	1659367719	0
4	0	1781822	1659367871	0

Test set:

	session	aid	ts	type
0	12899779	59625	1661724000	0
1	12899780	1142000	1661724000	0
2	12899780	582732	1661724058	0
3	12899780	973453	1661724109	0
4	12899780	736515	1661724136	0

Figure 3.1 Structure of train and test data

The test set includes data from the sessions that happened the week after the 4 weeks included in the train set. In the Figure 3.2 it is shown how the data was extracted from real sessions and the train sessions that overlapped with the test were trimmed.[34]

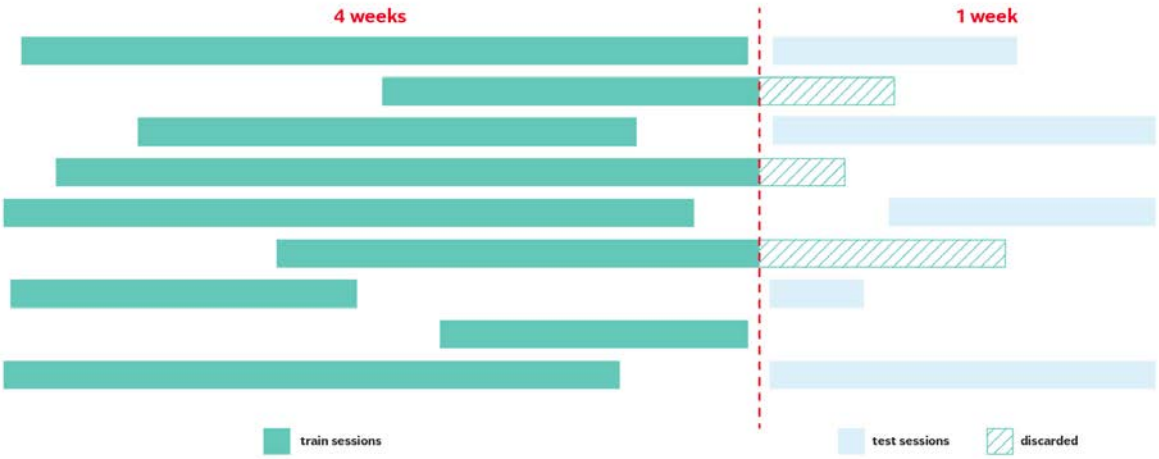


Figure 3.2 Data extraction from real user sessions and trimming. **[34]**

3.4 Evaluation

3.4.1 Recall (or True Positive Rate)

Recall is the evaluation performance metric that the submission file is evaluated with. A confusion matrix shows counts between expected and observed values. Table **3** Recall quantifies the number of positive predicted results made out of all positive data in the set. Precision is defined as the division of true positive values by all values that were predicted as positive while recall (or True Positive Rate) is the division of true positives by all values that should have been predicted as positive. Figure 3.3.The table is frequently used while solving classification problems. If the recall value is high, this means that our model is able to identify correctly a large percentage of positive occurrences as opposed to when its value is low where it means that the resulting predictions are quite dissimilar to the train set.[35, p. 138]

$$Recall = \frac{tp}{tp + fn} \quad , \quad Precision = \frac{tp}{tp + fp}$$

In e-commerce, an example would be if a recommended system provides 20 things to a user who looked for a white keyboard, 10 of which are genuinely white keyboards, 6 are black keyboards, 2 are white mousepads, and 2 black mousepads. Precision is 10 out of 20

or 50%. Recall 10 / 10 = 100%. In an example where the search algorithm missed some of the relevant items and 7 out of 10 were retrieved, then recall would be 7/10 = 0.7 .

		actual	
		positive	negative
predicted	positive	true positives	false positives
	negative	false negatives	true negatives

Table 3 Confusion Matrix

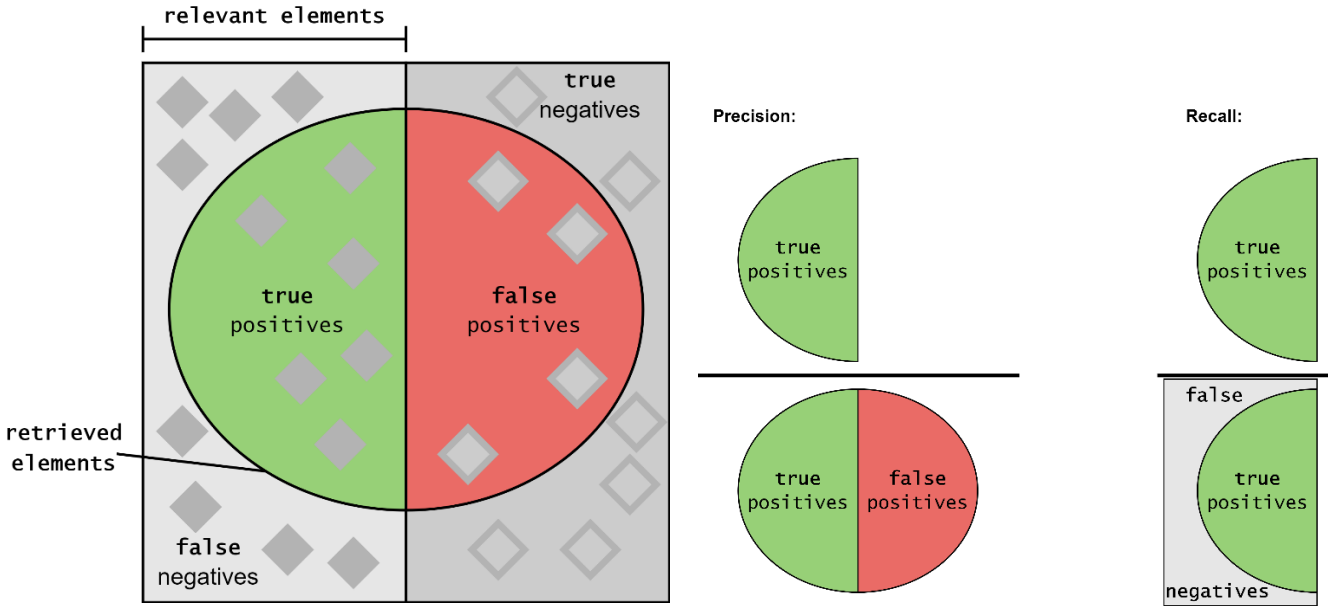


Figure 3.3 Precision and Recall

3.4.2 Weighted recall (wR)

Weighted recall (wR) is used on multiclass classification. Weighted recall assures that the performance on the minority class is taken into consideration by assigning weights

to each one of the classes. It is defined exactly as the recall but is calculated with weights.[36, pp. 37–63] [37, pp. 97–119]

In this occasion each action type (click, add to cart, or order) the three recall values are calculated weight-averaged:

$$\text{score} = 0.10 \cdot R_{\text{clicks}} + 0.30 \cdot R_{\text{carts}} + 0.60 \cdot R_{\text{orders}}$$

Where R_{type} is:

$$R_{\text{type}} = \frac{\sum_i^N \left| \{ \text{predicted aids} \}_{i,\,\text{type}} \cap \{ \text{ground truth aids} \}_{i,type} \right|}{\sum_i^N min\left(20, \left| \{ \text{ground truth aids} \}_{i,t_{ype}} \right|\right)}$$

In the above formula, N is the total number of sessions in the test set. Predicted aids are the predictions for each session-type. The ground truth value clicks is just one value for each test session. For carts and orders the ground truth includes all aid values that were added to cart and ordered respectively in the duration of the session.

3.5 Data analysis

Here are some stats about the data provided by OTTO already known from the documentation of the dataset, but can be easily extracted. [34] The following statistics can give us a good picture of the dispersion of the data but also can inform us to avoid incorrect conclusions.

The total number of events, unique products and sessions in both data sets is shown on the first table. Table 4 On the following table are shown the timestamps of first and last sessions. Table 5

	train dataset:	test dataset
unique sessions	12899779~12.9M	1671803~1.67M
events	216716096~217M	6928123~6.9M
unique product	1855603~1.85M	783486~783K

Table 4 Unique sessions/ events/ products

Unique number of items in each set:

Number of unique products in train dataset - 1855603
Number of unique product in test dataset - 783486
Number of common products in the sets – 783486

	session time	
	train dataset:	test dataset
First session	2022-07-31T22:00:00	2022-08-28T22:00:00
Last session	2022-08-28T21:59:59	2022-09-04T21:59:51

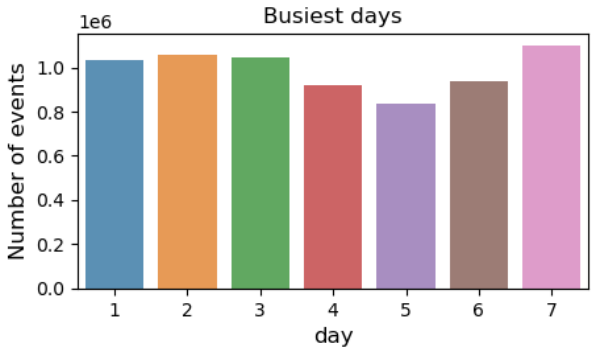
Table 5 First and last sessions timestamps.

Session with the longest time duration:

training set: 27days, 23 hours,59 minutes
test set: 6 days, 23 hours, 59 minutes

On the first graph the days with the most traffic/busiest days in terms of the amount of actions per day in each set are displayed Figure 3.4 . It is assumed the week starts on Monday, which is denoted by 1 and ends on Sunday which is denoted by 7. On the second the graphical correlation of the actions that occurred according to their type in each data set is Figure 3.5 .

Train set



Test set:

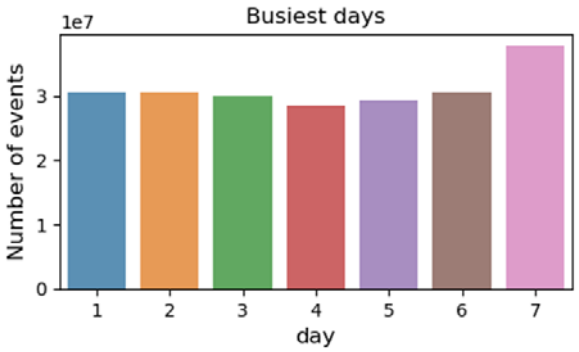
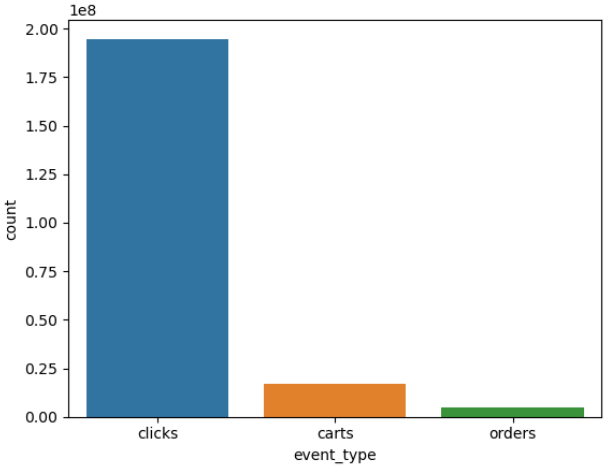


Figure 3.4 Graphical correlation of the busiest days

Train set:



Test set:

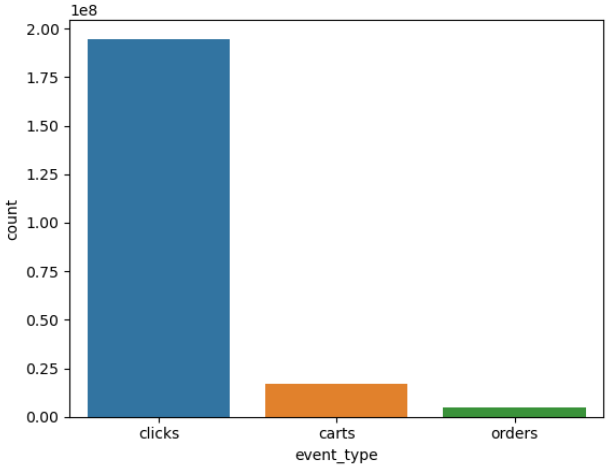


Figure 3.5 Graphical correlation of the number events by type.

On the next page are statistics of user movements in each set of clicks, additions to the card, and orders of the items recorded in each data set. Also statistics from the movements of the sessions per month are exported.

Train set:

Total number of each type of action in train set:

sessions[train_set] contains :

194720954 number of "clicks" in total.

16896191 number of "carts" in total.

5098951 number of "orders" in total.

Total actions of each type during 7,8 month:

During the 7 month,

320632 number of type "clicks"

22963 number of type "carts"

5517 number of type "orders"

During the 8 month,

194400322 number of type "clicks”

16873228 number of type "carts"

5093434 number of type "orders"

Test set:

Total number of each type of action in test set:

sessions[test_set] contain :

6292632 number of "clicks" in total.

570011 number of "carts" in total.

65480 number of "orders" in total.

Total actions of each type during each month:

During the 8 month,

2868445 number of type "clicks"

262328 number of type "carts"

35504 number of type "orders"

During the 9 month,

3424187 number of type "clicks"

307683 number of type "carts"

29976 number of type "orders"

On the table below the product clicked cart or order the most in the event s of the training set Table 6 . Also two diagrams with graphical correlation of the distribution of the number of actions taken in each session and the length of each session were produced Figure 3.6, Figure 3.7 .

aid	count
1460571	129004
485256	126836
108125	118524
29735	113279
1733943	105091

Table 6 Top 5 aids and number of occurrences in train set

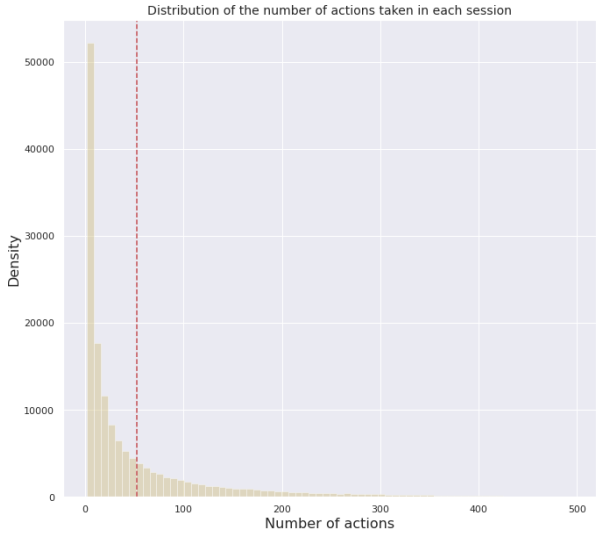


Figure 3.6 distribution of the number of actions in each session

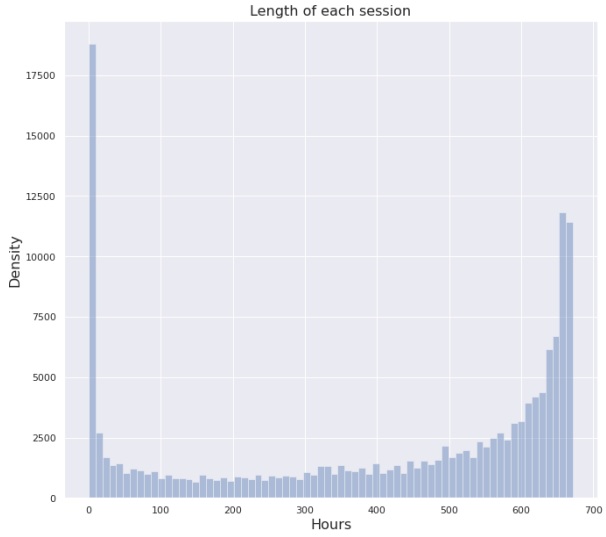


Figure 3.7 Length of each session

4 Methodology

In this Chapter, the planning followed for the creation of a recommendation system approach in the context of this diploma thesis is described in detail. First, reference is made to the available tools, and libraries used in most machine learning problems and the reasons that led to the selection of some of them are documented. Next, the format, specifics, and challenges of the "OTTO – Multi-Objective Recommender System" dataset are described, in order to create a basis for a better understanding for the attempts. The parameters that had to be decided for the execution of each approach are then analyzed. Also with the help of the evaluation system of the competition, the comparison of the methods that will be used will be done. However, in addition to the methods that will be applied, some improvements will also be proposed.

4.1 Libraries and Tools

4.1.1 Python

For this task the programming language that was selected was Python, as it is commonly used in machine learning problems. Python is used for systems currently under development around the world, and it also offers the most efficient interpreted language. Python is totally open source and there is a huge community of programmers providing solutions for common problems, coding is simpler than other languages and there are a lot of libraries containing functions and methods that everyone can use on their projects.

4.1.2 Pandas

Pandas is an open source library for python and it is used a lot in machine learning problems because it offers tremendously simplified data representation. This can significantly improve data analysis and comprehension resulting on better results. It provides features to filter and handle data faster while it makes analyzing data more flexible.[38]

4.1.3 Numpy

NumPy or Numerical Python, is also an open source library, that many scientist and engineers tend to use when working on numerical processes with a lot of data because it provides object structures and many methods to deal with complex problems. One example of a commonly used object structure is NumPy arrays which are faster and more efficient to create than standard python arrays.

4.1.4 Pickle

Pickle can serialize and deserialize a Python object structure. In other words, it is the act of transforming a Python object into a byte stream in order to save it to a database or a file. By unpickling the pickled byte stream, the original object hierarchy can be recreated, resulting in the original object structure. In this way someone can maintain a program state across sessions, or transport data over a network.[39]

When a byte stream is unpickled, the pickle module first makes an instance of the original object before populating it with the right data. To accomplish this, the byte stream only carries data relevant to the original object instance. However, simply having the data may not be enough. To effectively unpickle an object, the pickled byte stream contains instructions to the unpickler for rebuilding the original object structure, as well as instruction operands that contribute in populating the object structure.

In the following figure the main differences between the pickle protocols and JSON are listed. Figure 4.1 The screenshot was taken from the Python 3.11.1 documentation, Lib/pickle.py [39]

- JSON is a text serialization format (it outputs unicode text, although most of the time it is then encoded to `utf-8`), while pickle is a binary serialization format;
- JSON is human-readable, while pickle is not;
- JSON is interoperable and widely used outside of the Python ecosystem, while pickle is Python-specific;
- JSON, by default, can only represent a subset of the Python built-in types, and no custom classes; pickle can represent an extremely large number of Python types (many of them automatically, by clever usage of Python's introspection facilities; complex cases can be tackled by implementing specific object APIs);
- Unlike pickle, deserializing untrusted JSON does not in itself create an arbitrary code execution vulnerability.

Figure 4.1 Differences between the pickle protocols and JSON (JavaScript Object Notation)

4.1.5 Matplotlib

Matplotlib is a Python package that allows the user to create static, dynamic, and interactive plots or graphs that are suitable for publication. Exports can be produced in many formats, as there is a wide diversity available, and the style can also be customized and adjusted to the user's preference.

4.1.6 Seaborn

Seaborn is an open source library that is dataset-oriented and uses matplotlib for creating plots. It enables the user to manipulate data with panda dataframes, plot functions, and perform statistical aggregation to develop the graphs he want.

4.2 Understanding the data

In this subsection, additional information is provided to better understand the results, which are closely tied to the type and size of the data set in any problem.

It should be emphasized that the set of "OTTO - Multi-Objective Recommender System" data was published for the first time in order to create this competition. Nonetheless, this corporation is active in data technology, analysis, and machine learning, as seen by their 2015 "Otto Group Product Classification Challenge" competition on the Kaggle platform. This was a classification challenge, and the goal was to create a predictive model that could classify between the main product categories.

As mentioned in the introduction, the "OTTO - Multi-Objective Recommender System" data set is used. Here it should be noted that this is real data from real consumers, as well as implicit data. A detailed report on the structure of these data and data analysis are made in subsection 3.3 .

The sequence of data that were used had as mentioned a session id with the actions that happened in this session. The event in each session has three attributes which are aid(product id), ts(timestamp), type(clicks/ carts/ orders). The most common type of action is click, followed by adding to cart and order. The orders are the actions that interest us the most because they have a great influence on the evaluation process. Actions in each session are chronologically ordered by the way they were performed. The sessions, aids, and ts have integer values while the types contain strings with the type of actions subsection 3.3 .

The asked task is to predict what each of the test users, (test) sessions will do in the future. For each test session it is needed to the predict the following click, cart and order items, after the last record.

4.3 Connection to the server

The amount of training data is enormous, it has 12,899,778 sessions with 216,716,095 actions. To begin with, the ability to manage all of this information would be impossible if the institution did not have a machine to which a remote connection was made to run the code. The connection was achieved with Visual Studio Code Remote, an SSH extension which allows the user to open a remote folder on a computer with a running an SSH server. PuTTY was also used in order to check the performance of the tasks and the ram available on the machine.

4.4 Dealing with the datasets

Both of the datasets are in in the same format and of .jsonl or JSON Lines text format. Due to the size of the sets it was necessary to convert them to another object structure which will be easier to read and use for processing. The problem of this dataset was the RAM it required to run code for the training set. Depending on the strategy that is adopted to face the competition, there is a need for a certain data structure for handling the data in an efficient way. This means that performance of the model should be good in terms of speed and keep RAM to a minimum(memory-efficient). Furthermore, processing the data could take a lot of time, for each attempt to perform even a small test for a certain action in a subset. While running code, it was required to keep checking if the code was doing what was requested for each action and for this reason at the beginning for the first testing a subset of the data was actually used.

4.5 Creating a model

By studying the structure of the data, using some of the tools mentioned above for the analysis presented, followed the study of the model to create the algorithm of the recommendation system, and then some variations of it to improve the results. After the identification of the requested problem and the understanding of the data, some goals were set for the initial approach.

The original model was an improvised approach based on the nearest neighbor method. In this particular method, all sessions of the test set and a subset of the training set took part. The purpose was to find the distance between two sessions, a test session with a training session. This distance is supposed to be determined by comparing the similarity of the two sets in the model. In this situation, the objects that took part in the comparison were are the items of each session, and because each session contains three different types of actions, this should also be taken into account.

The calculation of these R distances is done using the evaluation type formula and calculating the R_{score} , between the train session and all tests for each type of Click, Cart, Order action. This requires that three R_{type} must be calculated for each session before the R_{score} can be calculated. But this comparison had to be done for all the test sessions and for each of them with all the training sessions.

4.6 First attempts

While reading the data, a structure was created which for each session contained the click cart and order items separately. The above procedure took place for both sets. This structure was saved using the aforementioned pickle library, to save the time of reading it each time. Then followed the calculation of the closest R sessions. Since the closest of a session were known the items of these could be extracted and enter into the corresponding position as predictions for the appropriate session. The original model was intended to find the distance R between two sessions, a test session with a training session. But this comparison had to be made for all test sessions and for each of them with all training sessions. This means that the comparison for each test session with every train had to be completed three times

The process took a long time to complete since the sets were two big and the result was that there were many empty rows of predictions in the output submission file, so necessarily the structures had to be changed. This happened probably because it was taken under consideration only a very small subset of the train set and because no extra calculations were done to add extra items to the predictions.

4.7 The model

The next steps to create the model were as follows:

The data structures that held the session items were changed and adjusted to contain as little information as possible. An index of all items was made for the train set in order to know in which sessions an item is contained. This action also could happen with only one passing of all train set.

Then, the test was scanned for each session, but only those sessions of the train set that had a common item were checked (through the indexes), so it was not necessary to go through the whole train dataset again but only access those sessions that have a common item.

Knowing in which sessions of the train set the items of a test appear, the code calculates the R_{scores} and keeps a number of the best scores. The result of the calculation of all R_{scores} for the sessions was saved in a 45GB .txt file where it was then passed and read to get the appropriate information. More specifically, during the construction of the file, the concerning train's session id, R_{types} and R_{score} is recorded in a row which is made for every test session, i.e. a row contains test session id and then many train referring numbers including the train session id the, R_{click}, R_{cart}, R_{order} and the final R_{score} of the best scores. The size of R_{score} i.e. closest sessions instead of 20 was chosen to be 1000 after some more tests. Since the R_{scores} were now saved the time spent comparing all these tests was significantly reduced (under 50%), as the calculation for the next moves was done while parsing the R_{score} file.

Later, from these nearest sessions common items are selected and added as suggestions for the corresponding test session. The common items are ranked in order based on the number of their occurrences in the nearest sessions. As it is already perceived

the items the criminal to the close sessions means that they are preferred by more users, therefore that they are of high preference among similar users.

If the number of selected items is less than 20, then a list is added to them according to the type mentioned with the top(clicks, carts, orders). The top are three lists with a size of at least 20 items containing the top_clicks top_carts top_orders of the train meaning items with the more appearances at events of each type. The addition is made after the suggestions from nearby train sessions to fill the requested size of 20 for the submit.

4.8 Adding weights

Another factor that played a significant role in the final results was the addition of weighting factors to the common items selected for the prediction. Weights can be used in recommender systems and assist the algorithm on delivering accurate and individualized recommendations to users. The system is assigning different values of weights to each factor for determining a better suggestion to the user. The outcomes attach a level of value to various features, characteristics and quality measures of the items suggested, allowing the system to personalize offerings to the needs and tastes of the individual user, resulting in a better user experience. The optimal values for the weights are chosen after running various trials by changing the function that generates them.

For example a recommender system creates a list with the preferences of a user based on past behaviour. In case the user’s habit is to choose more items on the section of gaming items then the system will suggest him items of that category but in this case it might add a higher value on this recommendations. The ranking of the items influences the determination of the items, which is affected by the weights applied. Without the weights, the system would not be able to effectively take into account the user's preferences. This would result in a less accurate recommendation.

4.8.1 Radial Basis Function

The initial function that calculates the weights was chosen as the Radial Basis Function. Radial basis functions act as a foundation for many machine learning subjects. Radial bias networks are used for both learning and data classification, and consist of an input layer, a processing layer (or hidden) and one or more outputs Figure 4.2. The parametric extension of Gaussian function was chosen as it is the one usually used on these types of problems. RBF calculates the similarity by taking as input the distance of the given points from some fixed point. The value of the function is calculated with providing a distance(Euclidean distance)from a point x_c . RBF with KNN or NN is used to give a larger value on weight to nearby points and less weight to distant points, resulting in a smoother classification than NN and KNN. The value of this function decreases as the magnitude of the distance given by the formula $||x - x_c||$ rises.

In our case it is assumed that R takes values from 0 to 1. R[0,1]. So in this case to calculate this distance we assumed that X takes the value 1 and xc is the value of each R_{score} in the training set that we examine each time.

$$\varphi(x, x_c) = \exp\left(-\frac{\|x - x_x\|^2}{2\sigma^2}\right)$$

The denominator of the fraction took the form of a constant α for simplicity. Values of a were changed during the tests to improve the results. [40]

$$\varphi(\mathrm{d}) = \exp\left(-\frac{(1-d)^2}{\alpha}\right)$$

Where, $d = R_{score}$

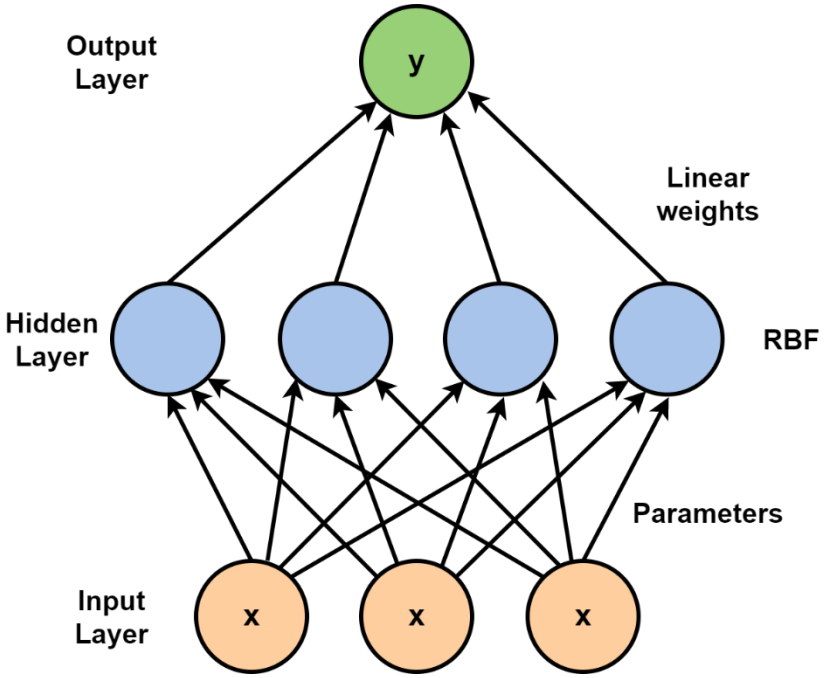


Figure 4.2 Structure of a radial basis function network.

The image illustrates the architecture of an RBF Network, including an input vector x, a RBF layer, and an output layer.

4.8.2 Inverse multiquadric kernel

Another mathematical function that was also used was the inverse multiquadric kernel (IMQ) which is used for approximation of multivariate data,

with a constant c > 0

$$k(\mathrm{x},\mathrm{y}) = \frac{1}{\sqrt{\| \mathrm{x} - \mathrm{y} \|^2 + c}}$$

In our case it is assumed that by converting the standard form of the multiquadric kernel to this:

$$k(\mathrm{x},\mathrm{y}) = \sqrt{\| \mathrm{x} - \mathrm{y} \|^2 + c}$$

And then by replacing k (x,y) with k (d) the x and y terms of the Euclidean distance ||x-y|| with d then :

$$k(\mathrm{d}) = \sqrt{d^2 + c^2}$$

Where, $d = R_{score}$

The constant c in the Multiquadric kernel function is a hyperparameter that affects the kernel width. Tests were performed to choose the right value for the constant c, and those with the best performance were selected. [41]

4.9 Optimizing accuracy through user actions and weights

The next step in modifying the recommendation algorithm of the model involved adjusting the predictions according to the user's actions. This involved incorporating the train sessions' choices into the prediction model in order to improve the accuracy of the recommendations. By saving the values of the product ids that each event included in a separate list, for each type of action in the test session it was now known which products the user was even slightly interested in to click, cart or order. After these three lists were completed, it became possible to include these options of items into future predictions. This could be performed in several ways by either including the same types of alternative items into the predictions or by combining options and providing additional flexibility in terms of effectiveness. Furthermore weights was also used on these relevant on the test user options on delivering accurate recommendation. The system is assigning different values of weights for each type to each test user's list for determining a more powerful option, providing better suggestion to the user. For example, if the clicks list was chosen to go to the predictions of the click action, this implies that the list of clicks made by the user was given a higher priority in the recommendation process for the clicks. By giving a higher value to the weights for the actions performed by each individual then these items could go first on recommendation priority, thus elevating the importance of related items.

Also combining these items allows for the consideration of multiple aspects and patterns of user's behaviour. Meaning for example, items that have been clicked on can determine items added to a cart, items added to a cart can determine items placed in an order, and items ordered can determine items placed in an order with a different value of weight. This played an important role in the final results.

Finally, the recommendation system then implements the established procedure of assigning weights to common items of a test session with the nearest training sessions and providing these options in addition to the choices of the test user.

4.10 Final modification of the model

In this subsection, in order effectively evaluate the results of a recommendation algorithm, the modifications of the model and the parameters that were altered during the testing and the evaluation

The final step in the construction of the model involved selecting the nearest training sessions for each test session not based on R_{scores}, but rather on a different calculation for each type of prediction. More specifically, the calculation used was determined based again on the evaluation formula for the prediction task. This time the code keeps a number of the best scores in two extra files and the nearest training sessions are chosen from each file for click, cart and order predictions. The best scores for each type of prediction were calculated as follows, $(0.1*R_{\text{click}})$ for clicks predictions, $(0.1*R_{\text{click}}+0.3*R_{\text{cart}})$ for additions to cart predictions and R_{score} for order predictions. These two additional files, with a file size of 46GB and 47GB respectively, are holding the results of the calculations and keep the same format as the initial R_{scores} .

This approach was performed to improve the accuracy of the predictions, as an alternative method by selecting the most relevant training sessions for each test session's type of action.

4.11 Review of tests

In this subsection, in order effectively evaluate the results of a recommendation algorithm, the modifications of the model and the parameters that were altered during the testing and the evaluation are reviewed. This information is presented through the use of the following tables and diagrams, providing a comprehensive overview. The impact of each modification is identified by the result of the evaluation process. Finally, the impact of various approaches followed has the potential to inform decisions about future improvements of this algorithm or other strategies.

First Model:

- determines **N** the nearest training sessions for each test session, selects shared items, and, if the quantity falls below 20
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

Attempt	*Number **N** of nearest sessions*	Final Score
1	***N** = 5*	0.2417
2	***N** = 20*	0.3331
3	***N** = 1000*	0.4625

Table 7 Evaluations of the Model #1

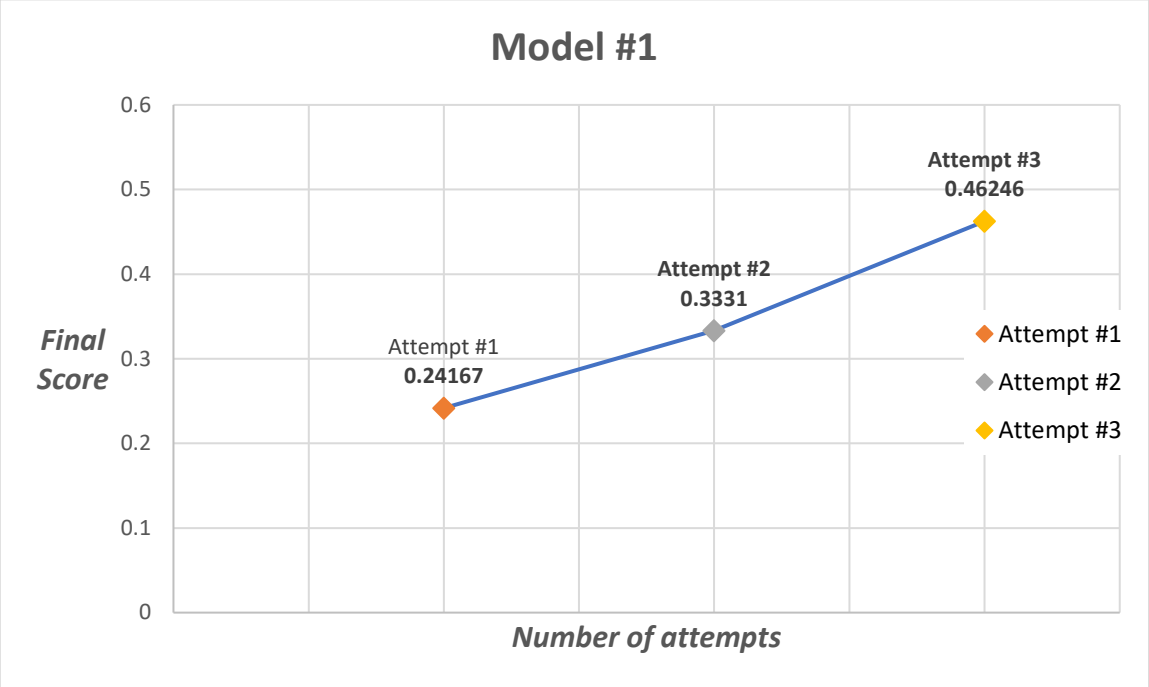


Figure 4.3 Model 1 - Results

From these results, it is visible that increasing the number N of nearest sessions used to make suggestions boosts the system's accuracy.

Second Model:

- determines **N =1000** (based on the results of the first model) the nearest training sessions for each test session, selects shared items,
- assigning weights based on a **RBF** radial basis function with constant **a**, and, if the quantity falls below 20
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

On this model, the third attempt was performed with the inverse multiquadric kernel (IMQ) and a=0.10 (based on 1,2)

Attempt	Weight function - Constant Value	Final Score
1	*RBF - a=0.05*	0.4549
2	*RBF - a=0.10*	0.4699
3	*IMQ - a=0.10*	0.4781

Table 8 Evaluations of the Model #2

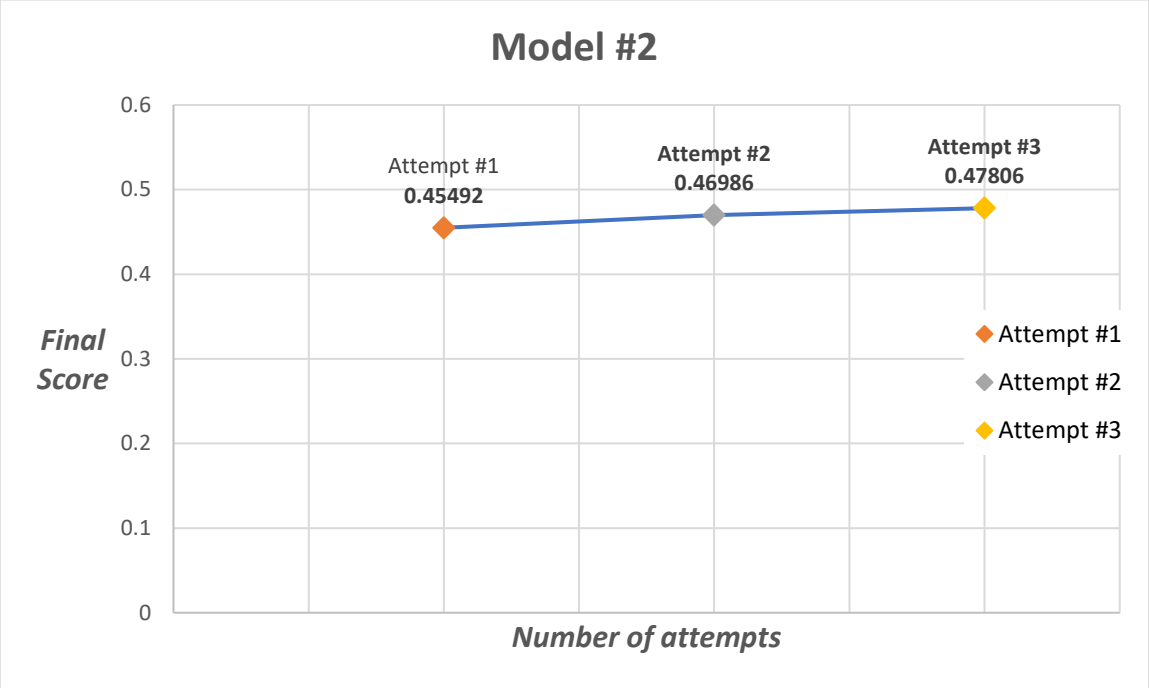


Figure 4.4 Model 2 - Results

In this modification of the model, by increasing the constant a to 0.10, and trying a different mathematical formula for the weights the outcome is considerably affected.

Third Model:

- determines **N =1000** (based on the results of the first model) the nearest training sessions for each test session, selects shared items,
- assigning weights based on a **RBF** radial basis function with constant **a**, and, if the quantity falls below 20
- Items that have been **clicked** on (by the test user) can determine **cart predictions**
- Items that have been **clicked** on or added to a **cart** (by the test user) can determine **order predictions**
- Note that on this model weights for test user choices, were adjusted with **low** values, meaning that other option could be high on priority if they appear many times.
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

On this model, the third and fourth attempt were performed with these changes:

- Items that have been **clicked** on (by the test user) can determine **cart predictions**
- Items that have been added to a **cart** (by the test user) can determine **order predictions**

Attempt	Weight function - Constant Value	Final Score
1	*RBF - a=0.50*	0.5545
2	*RBF - a=0.10*	0.5562
3	*RBF - a=0.10*	0.5312
4	*IMQ - a=0.10*	0.5561

Table 9 Evaluations of the Model #3

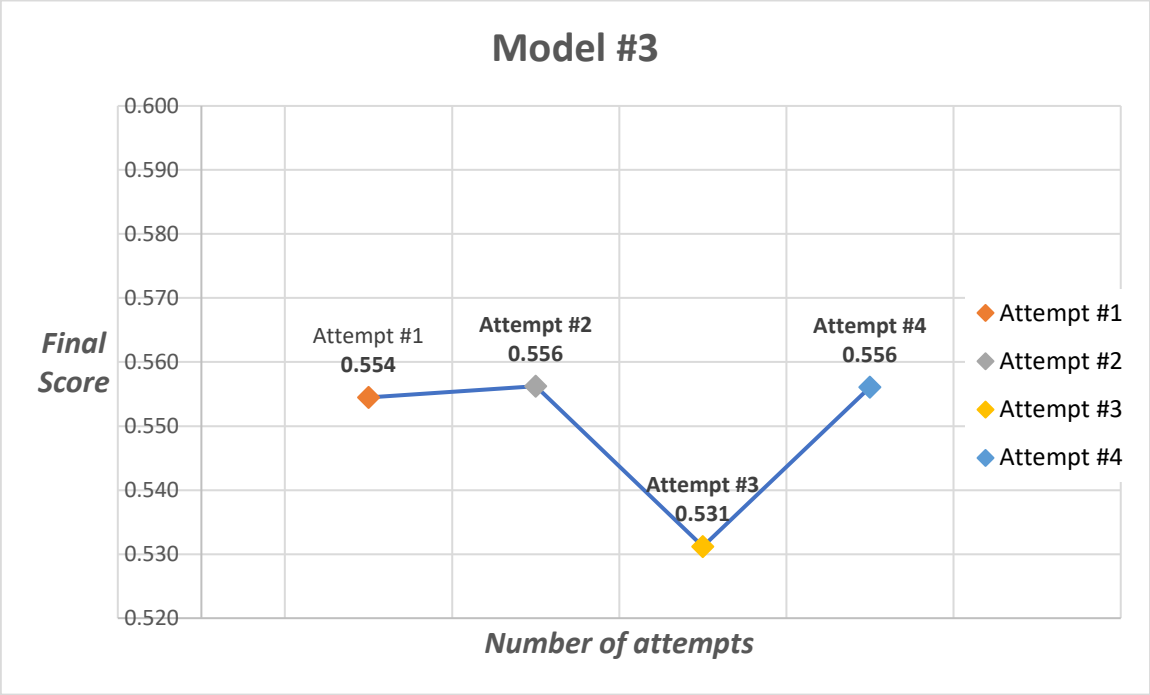


Figure 4.5 Model 3 - Results

Here, the data points of #1 and #2 attempts that used both clicks and carts on order predictions, show a better result than #3 which refers to the same function. The #4 attempt was done using IMQ but the score was also lower than RBF.

Fourth Model:

- determines **N =1000** (based on the results of the first model) the nearest training sessions for each test session, selects shared items,
- assigning weights based on a **RBF** radial basis function with constant **a**, and, if the quantity falls below 20
- Items that have been **clicked** on (by the test user) can determine **cart predictions**
- Items that have been **clicked** on or added to a **cart** (by the test user) can determine **order predictions**
- Note that on this model weights for test user choices, were adjusted with **high** values, meaning that other option could be high on priority if they appear many times.
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

Attempt	Weight function - Constant Value	Final Score
1	*RBF - a=0.50*	0.5589
2	*RBF - a=0.30*	0.5588
3	*RBF - a=0.10*	0.5562
4	*RBF - a=0.02*	0.5440

Table 10 Evaluations of the Model #4

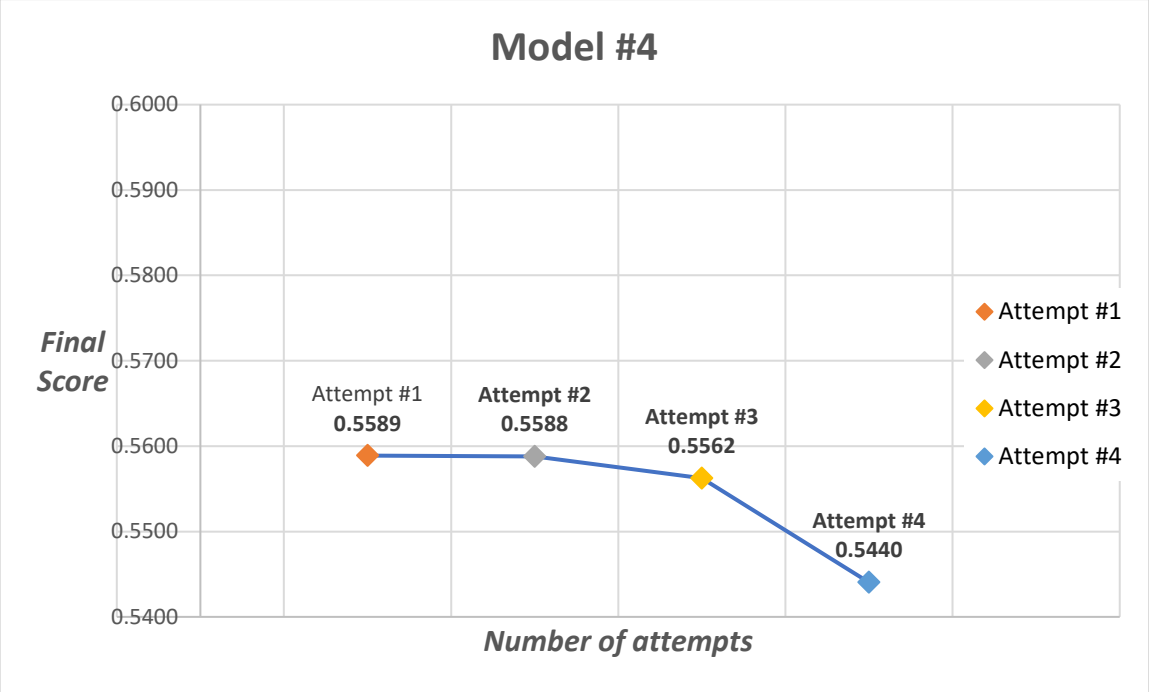


Figure 4.6 Model 4 - Results

Here, the graphical correlation shows that the efficiency of the predictions increases the same way as the value of constant a does from 00.2 to 0.5.

Fifth Model:

- determines **N =1000** (based on the results of the first model) the nearest training sessions for each test session, selects shared items,
- assigning weights based on a **RBF** radial basis function with constant **a**, and, if the quantity falls below 20.
- Note that on this model weights for common items changed depending on clicks(0.1* R_{click}), carts(0.1* R_{click} +0.3* R_{cart}), orders(R_{score})

- Items that have been **clicked** on (by the test user) can determine **cart predictions**
- Items that have been **clicked** on or added to a **cart** (by the test user) can determine **order predictions**
- Note that on this model weights for test user choices, were adjusted with **high** values as before.
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

Attempt	Weight function - Constant Value	Final Score
1	*RBF - a=2.50*	0.5590
2	*RBF - a=1.00*	0.5590
3	*RBF - a=0.50*	0.5589
4	*RBF - a=0.1*	0.5569
5	*RBF - a=0.004*	0.5355
6	*RBF - a=0.002*	0.5454

Table 11 Evaluations of the Model #5

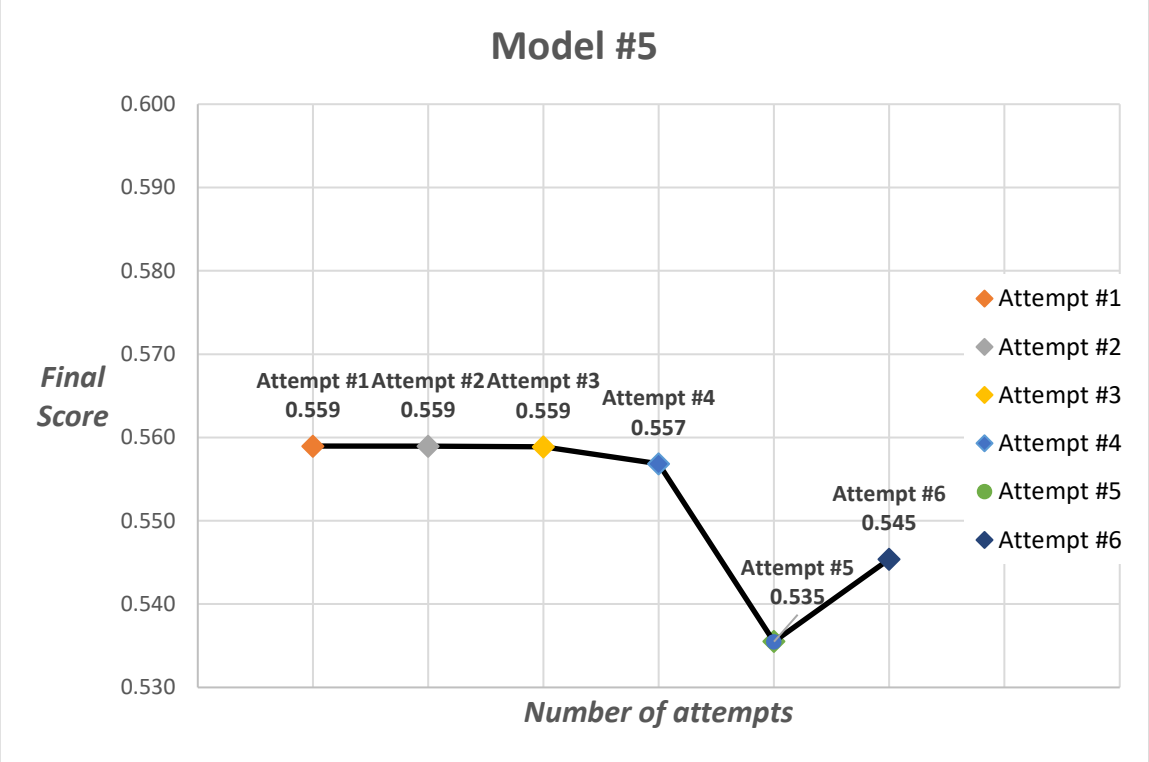


Figure 4.7 Model 5 - Results

This plot depicts the relationship between the value of constant a with the evaluation result on this model showing that while the value of a is high the evaluation is better.

Sixth Model:

- In this and last model 3 different files are accessed for each test session, highest $\boldsymbol{R}_{\mathbf{click}}$, highest **0.1*** $\boldsymbol{R}_{\mathbf{click}}$ **+0.3*** $\boldsymbol{R}_{\mathbf{cart}}$ and the initial $\boldsymbol{R}_{\mathbf{scores}}$ file , for **N** nearest sessions of each type of prediction. **N =1000** the nearest training sessions for each test session, selects shared items based on the type,
- assigning weights based on a **RBF** radial basis function with constant **a**, and, if the quantity falls below 20.
- Items that have been **clicked** on (by the test user) can determine **cart predictions**
- Items that have been **clicked** on or added to a **cart** (by the test user) can determine **order predictions**
- Note that on this model weights for test user choices, were adjusted with **high** values as before.
- augments with the **top items** (most popular) from clicks, carts, and orders from the training set.

Attempt	Weight function - Constant Value	Final Score
1	*RBF - a=2.50*	0.4709
2	*RBF - a=1.00*	0.5568

Table 12 Evaluations of the Model #6

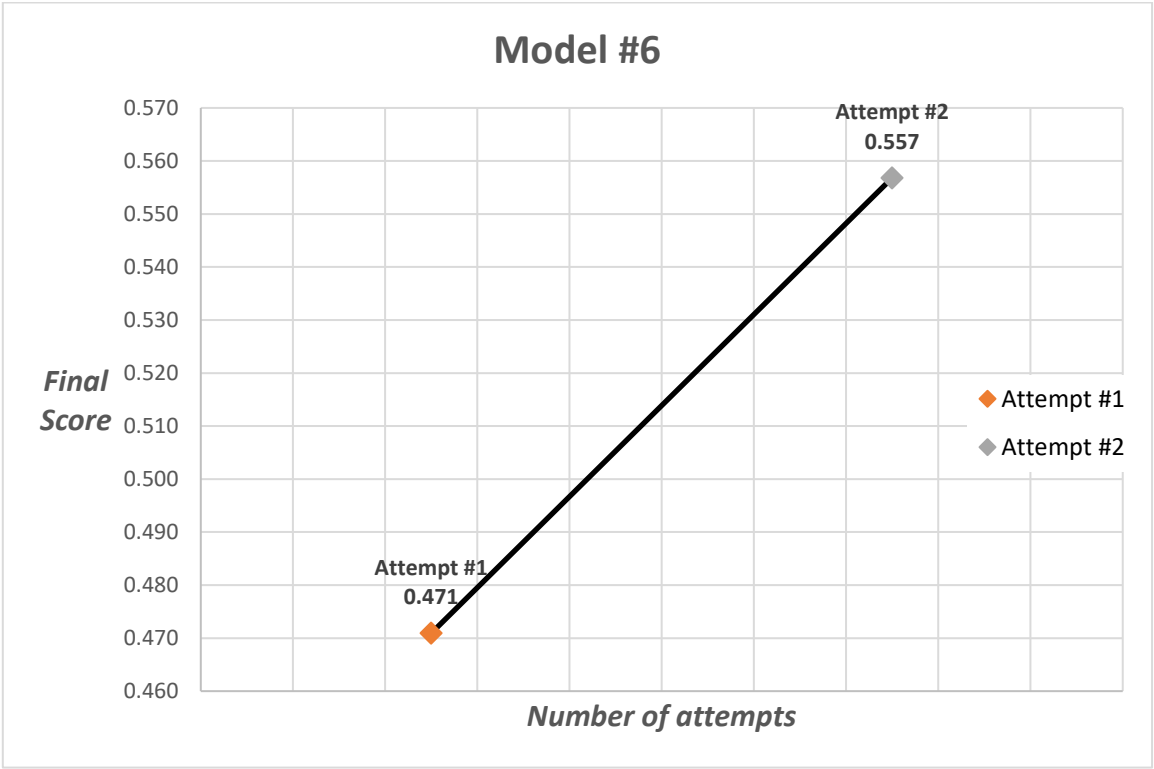


Figure 4.8 Model 6 - Results

On this last model the result is higher while the value of a is lower, but the results are not better that the previous model was (#6).

The following graph displays the results of 6 models.

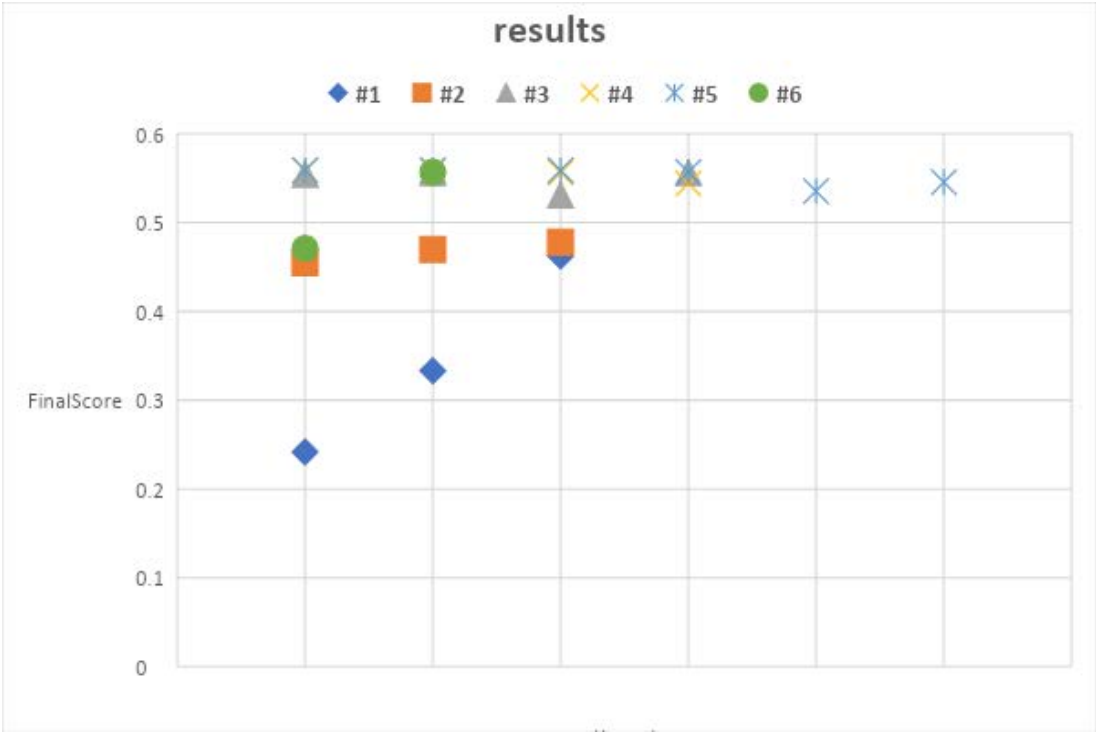


Figure 4.9 Results of all the models

5 Summary and conclusions

This thesis, aimed to highlight the use of recommender systems in e-commerce, by providing their importance on enhancing the shopping experience for both consumers and businesses. The thesis also focuses on collaborative filtering methods. After examining these methods a recommender system algorithm was implemented based on data of e-commerce sessions.

The approach proposed for this dataset, is an user-based collaborative filtering model, implemented by finding the nearest neighbors (or most similar users-training sessions) of the test user (session), and then recommending the common items in these sessions.

To improve the predictive accuracy the algorithm is modified many times by adding weights on the common items. Additional modification included adjusting the predictions into test user's preference and even by adding weights on these preferences of the test user. The results obtained from the tests show that even small adjustments to the parameters can significantly affect the result, therefore the programmer can achieve optima results. Finally the similarity measure for test and train users for each prediction type was also tested to check if the result would be improved.

5.1 Suggested Improvements

Considering the difficulties presented by this data set as well as its volume, the predictive ability of the model could potentially be greater depending on the strategy followed to tackle this problem.

Also, for the improvement of the accuracy on the model, it is necessary to use the exact moment in time (timestamp) at which each action occurred. Time may require running the model, using another data structure that will store per training period the product IDs ordered in time order. Thus, during the execution of the prediction model by type, different weights are added additionally based on the time series of the compared user to each object. Depending on one more factor for the execution, the model may result on giving greater options.

In addition, perhaps a pre-processing of the aggregate data could be applied, this would remove data that are incomplete or deviant, to make the algorithm produce more accurate recommendations.

Bibliography

[1] ‘Dr. Nilesh B. Gajjar / International Journal of Research In Humanities and Social Sciences’, *Humanities and Social Sciences*, vol. 1, no. 2, 2013.

[2] J. B. Schafer, J. Konstan, and J. Riedl, ‘Recommender Systems in E-Commerce’.

[3] A. Stankevich, ‘Explaining the Consumer Decision-Making Process: Critical Literature Review’, *JIBRM*, vol. 2, no. 6, pp. 7–14, 2017, doi: 10.18775/jibrm.1849-8558.2015.26.3001.

[4] P. Resnick and H. R. Varian, ‘Recommender systems’, *COMMUNICATIONS OF THE ACM*, vol. 40, no. 3.

[5] F. Ricci, L. Rokach, and B. Shapira, Eds., *Recommender Systems Handbook*. New York, NY: Springer US, 2022. doi: 10.1007/978-1-0716-2197-4.

[6] G. Linden, B. Smith, and J. York, ‘Amazon.com recommendations: item-to-item collaborative filtering’, *IEEE Internet Comput.*, vol. 7, no. 1, pp. 76–80, Jan. 2003, doi: 10.1109/MIC.2003.1167344.

[7] ‘Andriy Burkov - The Hundred-Page Machine Learning Book-Andriy Burkov (2019).epub’.

[8] R. Logesh, V. Subramaniyaswamy, D. Malathi, N. Sivaramakrishnan, and V. Vijayakumar, ‘Enhancing recommendation stability of collaborative filtering recommender system through bio-inspired clustering ensemble method’, *Neural Comput & Applic*, vol. 32, no. 7, pp. 2141–2164, Apr. 2020, doi: 10.1007/s00521-018-3891-5.

[9] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl, ‘GroupLens: an open architecture for collaborative filtering of netnews’, in *Proceedings of the 1994 ACM conference on Computer supported cooperative work - CSCW ’94*, Chapel Hill, North Carolina, United States, 1994, pp. 175–186. doi: 10.1145/192844.192905.

[10] *R. E. Nisbett – T. D. Wilson (1977): Telling more than we can know: Verbal reports on mental processes, Psychological Review, Vol. 84, No. 3.*

[11] akvileja, ‘A digital bookshelf: original work on recommender systems’, *Jussi Karlgren*, Oct. 01, 2017. https://jussikarlgren.wordpress.com/2017/10/01/a-digital-bookshelf-original-work-on-recommender-systems/ (accessed Feb. 05, 2023).

[12] C. Anderson, ‘Long Tail : Why the Future of Business Is Selling Less of More’.

[13] P. B.Thorat, R. M. Goudar, and S. Barve, ‘Survey on Collaborative Filtering, Content-based Filtering and Hybrid Recommendation System’, *IJCA*, vol. 110, no. 4, pp. 31–36, Jan. 2015, doi: 10.5120/19308-0760.

[14] M. de Gemmis, P. Lops, G. Semeraro, and P. Basile, ‘Integrating tags in a semantic content-based recommender’, in *Proceedings of the 2008 ACM conference on Recommender systems*, Lausanne Switzerland, Oct. 2008, pp. 163–170. doi: 10.1145/1454008.1454036.

[15] C. Basu, H. Hirsh, and W. Cohen, ‘Using Social and Content-Based Information in Recommendation’.

[16] ‘Introduction | Machine Learning’, *Google Developers*. https://developers.google.com/machine-learning/recommendation (accessed Feb. 06, 2023).

[17] J. B. Schafer, D. Frankowski, J. Herlocker, and S. Sen, ‘9 Collaborative Filtering Recommender Systems’.

[18] X. Luo, Y. Xia, and Q. Zhu, ‘Applying the learning rate adaptation to the matrix factorization based collaborative filtering’, *Knowledge-Based Systems*, vol. 37, pp. 154–164, Jan. 2013, doi: 10.1016/j.knosys.2012.07.016.

[19] M. Balabanović and Y. Shoham, ‘Fab: content-based, collaborative recommendation’, *Commun. ACM*, vol. 40, no. 3, pp. 66–72, Mar. 1997, doi: 10.1145/245108.245124.

[20] J. L. Herlocker, J. A. Konstan, A. Borchers, and J. Riedl, ‘An algorithmic framework for performing collaborative filtering’, in *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval*, Berkeley California USA, Aug. 1999, pp. 230–237. doi: 10.1145/312624.312682.

[21] X. Su and T. M. Khoshgoftaar, ‘A Survey of Collaborative Filtering Techniques’, *Advances in Artificial Intelligence*, vol. 2009, pp. 1–19, Oct. 2009, doi: 10.1155/2009/421425.

[22] C. C. Aggarwal, *Recommender Systems*. Cham: Springer International Publishing, 2016. doi: 10.1007/978-3-319-29659-3.

[23] M. Kuanr, P. Mohapatra, and S. S. Choudhury, ‘TSARS: A Tree-Similarity Algorithm-Based Agricultural Recommender System’, in *Recommender System with Machine Learning and Artificial Intelligence*, 1st ed., S. N. Mohanty, J. M. Chatterjee, S. Jain, A. A. Elngar, and P. Gupta, Eds. Wiley, 2020, pp. 387–400. doi: 10.1002/9781119711582.ch20.

[24] J. S. Breese, D. Heckerman, and C. Kadie, ‘Empirical Analysis of Predictive Algorithms for Collaborative Filtering’.

[25] S. Maddodi, ‘Netflix Bigdata Analytics- The Emergence of Data Driven Recommendation’.

[26] M. A. Ghazanfar, A. Prügel-Bennett, and S. Szedmak, ‘Kernel-Mapping Recommender system algorithms’, *Information Sciences*, vol. 208, pp. 81–104, Nov. 2012, doi: 10.1016/j.ins.2012.04.012.

[27] A. S. Das, M. Datar, A. Garg, and S. Rajaram, ‘Google news personalization: scalable online collaborative filtering’, in *Proceedings of the 16th international conference on World Wide Web*, Banff Alberta Canada, May 2007, pp. 271–280. doi: 10.1145/1242572.1242610.

[28] G. Adomavicius and A. Tuzhilin, ‘Toward the next generation of recommender systems: a survey of the state-of-the-art and possible extensions’, *IEEE Trans. Knowl. Data Eng.*, vol. 17, no. 6, pp. 734–749, Jun. 2005, doi: 10.1109/TKDE.2005.99.

[29] A. I. Schein, A. Popescul, L. H. Ungar, and D. M. Pennock, ‘Methods and Metrics for Cold-Start Recommendations’.

[30] Kai Yu, A. Schwaighofer, V. Tresp, Xiaowei Xu, and H. Kriegel, ‘Probabilistic memory-based collaborative filtering’, *IEEE Trans. Knowl. Data Eng.*, vol. 16, no. 1, pp. 56–69, Jan. 2004, doi: 10.1109/TKDE.2004.1264822.

[31] ‘OTTO - Mode, Möbel & Technik » Zum Online-Shop’. https://www.otto.de/ (accessed Feb. 04, 2023).

[32] ‘OTTO (GmbH & Co KG) – At a glance’, *OTTO (GmbH & Co KG) – At a glance*. https://www.otto.de/unternehmen/en/who-whe-are/at-a-glance (accessed Feb. 04, 2023).

[33] ‘Otto Group: Otto Group growing sustainably and successfully in e-commerce’. https://www.ottogroup.com/en/medien/newsroom/meldungen/Otto-Group-growing-sustainably-and-successfully-in-e-commerce.php (accessed Feb. 04, 2023).

[34] Philipp Normann, Sophie Baumeister, Timo Wilm, ‘OTTO Recommender Systems Dataset: A real-world e-commerce dataset for session-based recommender systems research’. OTTO (GmbH & Co. KG), Nov. 01, 2022. Accessed: Feb. 04, 2023. [Online]. Available: https://github.com/otto-de/recsys-dataset

[35] D. L. Olson and D. Delen, *Advanced data mining techniques*. Berlin Heidelberg: Springer, 2008.
[36] D. Powers, ‘Evaluation: From Precision, Recall and F-Factor to ROC, Informedness, Markedness & Correlation’.
[37] P. Perruchet and R. Peereman, ‘The exploitation of distributional information in syllable processing’, *Journal of Neurolinguistics*, vol. 17, no. 2–3, pp. 97–119, Mar. 2004, doi: 10.1016/S0911-6044(03)00059-9.
[38] ‘pandas documentation — pandas 1.5.3 documentation’. https://pandas.pydata.org/docs/ (accessed Feb. 08, 2023).
[39] ‘pickle — Python object serialization — Python 3.11.1 documentation’. https://docs.python.org/3/library/pickle.html#comparison-with-json (accessed Feb. 08, 2023).
[40] H. Chen and B. R. Bakshi, ‘Linear Approaches for Nonlinear Modeling’, in *Comprehensive Chemometrics*, Elsevier, 2009, pp. 497–504. doi: 10.1016/B978-0-444-64165-6.02009-7.
[41] C. A. Micchelli, ‘Interpolation of scattered data: Distance matrices and conditionally positive definite functions’.

Appendix – Code

The ***code*** of the tests, as well as all the visualizations presented are available at the link:

https://github.com/amartid/E-commerce-recommendation-system

The ***data*** is stored on the Kaggle platform:

https://www.kaggle.com/competitions/otto-recommender-system/data

and can be downloaded using their API:

kaggle competitions download -c otto-recommender-system

To run the final experiment code it is required to install:

- Python
- Jupyter Notebook
- Install the necessary Python packages
- Place the data files and ***ipynb*** files in a the folder with the ***data***