RECOMMENDATION SYSTEM FOR ICARUS PLATFORM

By

Kristian Litsis

A thesis submitted to
the University of Cyprus
for the degree of
COMPUTER SCIENCE

Icarus
Department of Computer Science
University of Cyprus

May 2020

Declaration

I hereby certify that the material, which I now submit for assessment on the programmes of study leading to the award of Bachelor of Science, is entirely my own work and has not been taken from the work of others except to the extent that such work has been cited and acknowledged within the text of my own work. No portion of the work contained in this thesis has been submitted in support of an application for another degree or qualification to this or any other institution.

Student Name

May 22, 2020

ACKNOWLEDGMENTS

I would like to express my heartfelt gratitude to my supervisor Dr. George Pallis, Assistant Professor in the Computer Science Department of University of Cyprus, for his endless willing and the opportunity that gave me to work my thesis in one of the biggest new-age trends that recommendation system represent and be part of a big European project like Icarus is. I would also like to thank him for trusting in me and for the opportunity he has given me to.
Moreover, I would like to thank the MS.C. candidate Chrysovalantis Christodoulou for his excellent guidance and the chance he gave me to learn an incredible amount of technologies and consultancies.
Last but not least, I would like to say a big thank you to my friends and family for the support and support they have shown me during this work.

DEDICATION

This dissertation/thesis is dedicated to my family and my friends who provided both emotional and financial support

ABSTRACT

The recent developments in machine learning algorithms in Recommendation systems and the integration of them into business have become a prime focus of research for gaining profit for both customers and the providers.

In this thesis, we focus on the Hybrid recommendation system that will be used for the ICARUS platform, a European Project in development, that will present to the customer's data about aviation industries. Therefore, a content-based system will be used for new users that entered the platform where it will solve the cold start problem and Collaborative filtering for finding similar users and items to recommend based on ratings. In the end, a weighted hybrid recommender will be implemented to combine the results of different techniques into a single recommendation list that will treat the outputs of each technique as inputs to generate a function. This function will make it possible to change the system dynamically by introducing a coefficient that will multiply each input. Then we have performed a data visualization process in order to represent the recommendation list of the User, and for this, we implemented a Restful API that can be called from every User. Moreover, we tried different approaches for the Hybridization with the Dataset provided by the Icarus Platform, and we compared those results to decide with which method to proceed.

In the end, Prediction accuracy, Decision support, and some non-traditional metrics were performed to evaluate the recommender where we achieved around 85% precision, 35% recall and around 50% f1-score. Finally, an outline of the conclusions drawn from the research is given and some suggestions for future work are proposed.

Table of Contents

Page

1 Introduction . . . 1
1.1 Motivation . . . 1
1.2 Challenges . . . 2
1.3 Contributions . . . 3
1.4 Outline Contents . . . 4

2 Literature review and Related work . . . 7
2.1 History and Overview about Recommendation Systems . . . 7
2.1.1 Types of Recommender System . . . 8
2.1.1.1 Content-based filtering . . . 8
2.1.1.2 Collaborative filtering . . . 9
2.1.1.3 Hybrid and Deep Learning . . . 10
2.2 Importance of Recommendation systems . . . 11
2.3 Recommendation systems in Aviation industry . . . 13
2.4 Challenges and problems of Recommenders . . . 13
2.4.1 Scalability . . . 13
2.4.2 Data Sparsity . . . 14
2.4.3 Cold Start . . . 15
2.4.4 Reduced Coverage . . . 15
2.4.5 Shilling Attacks . . . 16
2.4.6 Gray Sheep . . . 16
2.5 Similar Researches/Models on Recommender Systems . . . 17
2.6 Visualization for Recommender System . . . 20

3 Methodology . . . 22
3.1 Methodology Overview . . . 22
3.2 Problem Formulation . . . 23
3.3 Data Collection . . . 25
3.4 Data Visualization and REST API . . . 25

3.5 Building the Recommender 28
3.5.1 Collaborative filtering model 28
3.5.2 Content-Based model 29
3.5.3 Hybrid model 30
3.6 Prediction Generation 31

4 Evaluation 34
4.1 Recommender performance 34
4.2 Comparisons 37
4.2.1 Experiment Setup 37
4.2.2 Representation of data 38
4.2.3 Results and Comparison 39

5 Conclusion 49
5.1 Conclusion 49
5.2 Lesson Learnt 50
5.3 Future Work 51

References 54

Appendices

A Acronynms List 56

List of Figures

2.1 Types of recommenders 8
2.2 Content Based recommender 9
2.3 Collaborative filtering recommender 10
3.1 Overview of Icarus Recommender 24
3.2 Representation of Restful API 26
3.3 Method of Payment 27
3.4 Target Purpose. 27
3.5 k-NN basic prediction equation 29
3.6 Cosine similarity equation 29
3.7 Cosine similarity between two vectors 29
3.8 Representation of User/Item vectors 30
3.9 Hybrid model 31
3.10 Top 10 Predictions of the Recommender 32
4.1 Novelty of Recommender 36
4.2 Long tail plot of items 38
4.3 Distribution of all ratings 39
4.4 Coverage of the Recommenders 41

4.5 Mean Average recall 42

4.6 kNN results for different k neighbors 43

4.7 Estimation ratings comparison 44

4.8 kNN Baseline measures for top 5 recommendations 44

4.9 SVD measures for top 5 recommendations 45

4.10 Precision and Recall for 2 algorithms 46

4.11 kNN measures for top 10 recommendations 47

4.12 SVD measures for top 10 recommendations 47

List of Tables

2.1 Hybridization methods based on Robin Burke 11

2.2 Different recommenders and their advantages 18

4.1 Evaluation of different Algorithms 40

4.2 Evaluation of Novelty and Personalization 42

Chapter One

Introduction

Contents

1.1 Motivation 1

1.2 Challenges 2

1.3 Contributions 3

1.4 Outline Contents 4

1.1 Motivation

It is a fact that in the era we live in, due to the rapid development of technology and big data, people are more dependent on devices and different web-based platforms that help their everyday tasks. These tasks may include social interactions with their companions, websites where they can access the information that they need or for eCommerce shopping. This has lead to the advancement of different recommendations systems where such information like user preferences and interactions are gathered to help the system find which recommendations would be preferred the most and push the User to buy shortly. Recommendations systems are one of the most important research areas today's because it helps users to find their interest in the internet [15]. The users of such systems often have diverse, conflicting needs. Differences in personal preferences, social and educational backgrounds, and private or professional interests are pervasive. As a result, it seems desirable to have personalized intelligent systems that process, filter, and display available information in a manner that suits each individual using them. The need for personalization has led to the development of systems that adapt themselves by changing their behavior based on the inferred characteristics of the User interacting with them. Moreover, these systems will help not only the User be satisfied, but also there would be profits from companies and different organizations that will provide and sell the data.

1.2 Challenges

Today's Recommender system is a relatively new area of research in machine learning and artificial intelligence and has accumulated a lot of attention in every field to make a profit. A recommendation engine can be very powerful while, on the other hand, it can be torture that is easily manipulated. First and foremost, continuous and valid Dataset is required for the Recommender to work successfully, and this is very hard to find it. Because Icarus is still a running project, we had to create our mock data given the structure of the Dataset and the specifications that were presented in the deliverables.

Another obstacle that we faced was the cold start problem on how to deal with new users and products that don't have any history and interaction with the system. Besides that, in the user events dataset, the user-item rating matrix was very sparse(many NaN items) because we created a realistic dataset where many items are provided, and all those products will not be rated by many users, in our case the companies. In the real-world, a regular user does not give ratings to even 1 percent of the total items [29]. Therefore, around 99 percent of the cells of this matrix is empty. This sparsity makes training computationally inefficient, and the prediction very difficult.

Furthermore, we faced difficulties in dealing with the products that were practically the same by the description but different in their content. For example, different datasets were linked with airplanes, but one would present air tickets while others would present accidents. Since we don't use product descriptions for collaborative filtering, we can miss the information about similar items.

Because of the nature of the creation of the datasets that were based in the structure by using data fabrication, we introduced another challenge, the known Grey-Sheep Problem, which from the name it means that a group of users who have special tastes and may agree or disagree with the majority. These behaviors are unpredictable because related items are rated offbeat, and this will confuse the system to find the hidden patterns and will decrease the accuracy of the Recommender.

Last, a difficult found in the project was the evaluation of the Recommender. My thesis objective was to provide recommendations of the datasets to the users, which means that there is not just one correct result because it is not a classification problem. However, we are able to filter the unwanted recommendations based on the analysis that we performed, and we were able to provide the most similar items.

1.3 Contributions

Our research aims to provide results for the User that will use the platform and help both the provider and also the consumer to maximize the profit from both parties. Moreover, what we are trying to achieve is to create a hybrid recommender that will be able to eliminate all the possible problems and give recommendations based on different features. When deploying the collaborative filtering model, we often would run into problems that we have to predict for unseen items or users. The implementation of the hybrid model can solve this by analyzing the content and every specific characteristic of the data. On the other hand, a few of the data are missing cause we have not been able to collect them, and this will lead to difficulties in Content-based methods, but here comes again Hybridization where it will find the similarities between categories.

Another challenge that we wanted to contribute was on how to handle the data. These data are complex and can be derived from heterogeneous data sources. To handle this challenge, we decided to use ontologies that can be applied to facilitate the modeling of the data across multiple data sources. The Ontology is described as: "An explicit specification of a conceptualization"[Tom Gruber]. From these ontologies, we could gather all the metadata of the datasets and create an item vector that would describe users and items. By using the ICARUS ontology, the recommendation model would take into account both the contextual hierarchy and the semantic annotations of the concepts. Furthermore, it will utilize the semantic functionalities and will run through SPARQL queries, where scores/ranks that are that will be returned from the Recommender will be used to provide the most appropriate datasets that best match the preferences of a target user.

Furthermore, after lots of experiments that were performed in the Hybrid model, we decided to use a function that would combine all the algorithms, and we introduced a parameter that would change the model dynamically. This parameter is a coefficient in the weighted hybrid model that depends on the sparsity of the Dataset in the user interaction with the platform. By doing this, we achieved better results in the recommendations, and we could avoid many problems.

Last but not least, we decided to have for the Recommender a Restful API, which is a black box whose implementation details are unclear and can be called to recommend different data for the User. The core of the system is a flask app that receives a user ID and returns the relevant items for that User.

1.4 Outline Contents

Chapter 1: Introduction

In the introduction chapter, we briefly present how recommendation systems have expanded nowadays and how these systems are used to help users find their favorite items on the web. In addition, we mention the challenges that we encounter in our research and the output that we will represent as a recommendation to our User. Lastly, we talk about our contribution to this area and how it will be used for future projects.

Chapter 2: Literature review and related work

In the introduction chapter, we briefly present how recommendation systems have expanded nowadays and how these systems are used to help users find their favorite items on the web. In addition, we mention the challenges that we encounter in our research and the output that we will represent as a recommendation to our User. Lastly, we talk about our contribution to this area and how it will be used for future projects.

Chapter 3: Methodology

In the Methodology chapter, we explain in detail the process of our work, and we represent a detailed analysis of our research. We explain how our recommender system works using diagrams and graphs, how we create the data using Data Fabrication and different software that help us with the platform. In addition, we describe the process of creating a hybrid system by combining two of the known methods the collaborative filtering with content-based that are combined to give a final score in the form of a black box API that the User can access. Moreover, we represent all the technology used, and we provide the result of our experiment by drawing some conclusions.

Chapter 4: Evaluation

Chapter 4 focuses on the presentation of our findings and our results from the experiment. We compare our results with the features that literature work discovered by showing the similarities and dissimilarities between recommenders. Furthermore, in this chapter, we present all the different algorithms that we studied and compared their performance in various types of datasets.

Chapter 5: Conclusion

In the final chapter, we define our conclusions and summarize them to provide an outcome for the reader.We also provide all the lessons learned in this dissertations along with the challenges. Finally, we propose how the system can be improved in the future by analyzing in more detail user preferences and descriptions and by introducing more sophisticated techniques.

Chapter Two

Literature review and Related work

Contents

2.1 History and Overview about Recommendation Systems **7**

2.1.1 Types of Recommender System 8

2.2 Importance of Recommendation systems **11**

2.3 Recommendation systems in Aviation industry **13**

2.4 Challenges and problems of Recommenders **13**

2.4.1 Scalability 13

2.4.2 Data Sparsity 14

2.4.3 Cold Start 15

2.4.4 Reduced Coverage 15

2.4.5 Shilling Attacks 16

2.4.6 Gray Sheep 16

2.5 Similar Researches/Models on Recommender Systems **17**

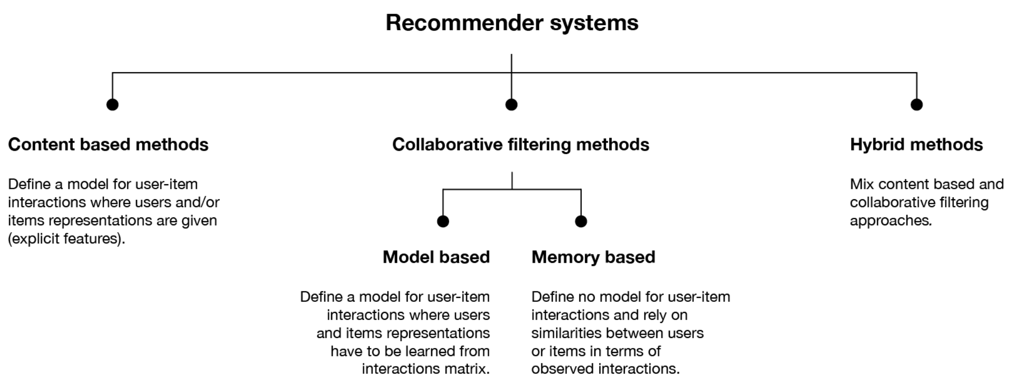
2.6 Visualization for Recommender System **20**

2.1 History and Overview about Recommendation Systems

Nowadays with these technological developments, people used to buy more products online on the web than from stores. In the past, the purchase of items was based on the reviews that their relatives or friends had given, but now as the internet has advanced, we need to assure clients that the product is good and they would like to buy. To give this confidence, recommender systems were built. Recommendations systems are machine learning applications in business. These engines filter out the products that a particular user would be interested in buying or would buy based on his/her previous purchase. The clear main purpose of the current recommender systems is to guide the user to useful/interesting objects.

2.1.1 Types of Recommender System

There are many recommendation filtering techniques that each system operates in several domains of application. In the following section, several types of Recommendation Systems are presented, along with specific characteristics and examples of applications in which they are used.



Recommender systems
Content based methods
Define a model for user-item interactions where users and/or items representations are given (explicit features).
Collaborative filtering methods
Model based
Define a model for user-item interactions where users and items representations have to be learned from interactions matrix.
Memory based
Define no model for user-item interactions and rely on similarities between users or items in terms of observed interactions.
Hybrid methods
Mix content based and collaborative filtering approaches.

Figure 2.1

2.1.1.1 Content-based filtering

The content-based filtering approach is based on a description of the item and profiling the user's preference. They ignore interactions between users and items. In a CB recommendation system, keywords are used to define the items. Besides that, a user profile is built to indicate the category of the item this User likes [21]. These methods try to recommend items that are related to those that a user liked in the past (or analyzing for the present). They try to find various candidate items by comparing them with previously rated items by the User, and when items with the same content are found, then the result is shown. Furthermore, since we make recommendations for only a particular user and we don't use any interaction of that User with the system, we make the Recommender more scalable in the term of the number of users. On the other hand, content-based algorithms depend on the size of the item-set. We need to examine all the items to find the similarities, and as new data are introduced, the accuracy of the Recommender will decrease, making this a drawback of this algorithm.

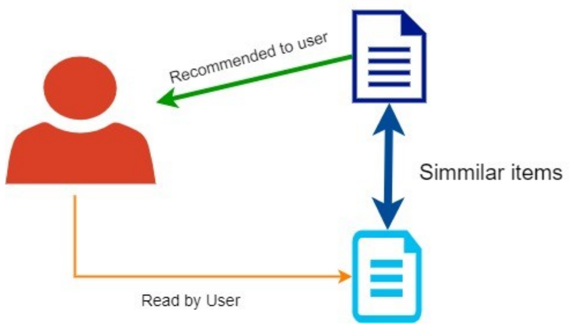


Figure 2.2 New item will be recommended to user based on similar items

2.1.1.2 Collaborative filtering

Collaborative filtering (CF) is the procedure of filtering or evaluating items over the judgments of other people(Schafer et al.). This model uses implicit or explicit interactions of users with items (like metadata or different rating and feedbacks). They try to match users with similar interests. This technology brings together the opinions of large interconnected communities on the web, supporting the filtering of substantial quantities of data [27]. A key advantage of CF is that this approach does not rely on analyzing the content of the item, and therefore it is capable of finding recommendations accurately without requiring information, but his obstacle is the new User that has not done any interaction in the system.

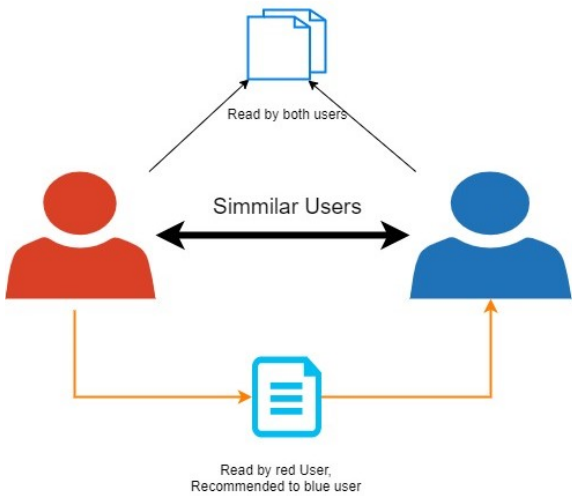


Figure 2.3 New item will be recommended to user based on similar user

2.1.1.3 Hybrid and Deep Learning

Hybrid Recommenders combine both approaches that were mentioned above and overcome a lot of the challenges of each method. These can be implemented in several ways, by adding some of the components of Collaborative filtering to content-based or vice versa, or by combining both of them to a new unified system. These methods are more accurate and work well in different challenges by eliminating each weakness of each algorithm mentioned. Although the positive facts that we presented for the hybrid models, estimating user ratings still remain a difficult task. This challenge lies in the complexity that it is not easy to describe user preferences just with some analysis in the metadata. Human interests are continuously changing due to various factors in the real-life making the recommendation very difficult.
On the other hand, there is also another category that can be used by combining multiple methods to create a new model, and we call them Deep Learning techniques. These methods use multiple layers of neurons that create a hidden layer(the reason why it's called deep), which we can not control, and by using functions such as gradient descent (SGD), they minimize the error of predictions. These methods can achieve greater accuracy when combined with content-based or collaborative filtering, and they can use both supervise or unsupervised learning.

Below, we represent all the methods that a hybrid system can be constructed, and for each method, we provide a description that was given by Burke in his book[4].

2.1. Table of Hybridization

Hybridization method	Description
Weighted	The scores of several recommendation techniques are combine together(as votes) to produce a single recommendation
Switching	The system switches between recommendation techniques depending on the current situation
Mixed	Recommendations from several different recommenders are presented at the same time
Feature combination	Features from different recommendation data sources are thrown together into a single recommendation algorithm
Cascade	One recommender refines the recommendations given by another
Feature augmentation	Output from one technique is used as an input feature to another
Meta-level	The model learned by one recommender is used as input to another

Table 2.1 Hybridization methods based on Robin Burke

In my research, we decided to go with the Weighted method, and we implemented this by representing each recommendation as a function using a coefficient that was dependent on the sparsity of the data. So the system would be dynamically changed as soon as more user interaction was imported to the system.

2.2 Importance of Recommendation systems

The evolution of the industry and, in particular, the computers in combination with the rapid development of network infrastructure has bought online shopping to a new level. This has become a challenge in e-commerce. We can't wait for customers to come to us. We have to figure out where they are, go there and drag them back to the store (Paul Gram). However, as the information grows and becomes larger every day, the internet becomes overloaded. This leads directly to the use of the recommendation technology to manage this information and provide what is best for the clients.

The recommenders are systems that use different algorithms to suggest items to a user based on different characteristics. We encounter them in different areas, shopping, news reading, movies, songs, and many others. Recommenders are used to make every User's decision easier. These decisions are mostly about low-cost environments such as book and movie suggestions, with their primary scope being to relieve the User from long searches (Jannach et al., 2010; Ricci et al., 2011) [10] [23].

There are a lot of benefits from the use of these recommendation systems. The most important benefits include making business at any time (availability) and from any possible place and financial gaining for both the business, which reduces the cost of maintenance and salaries and for the customer who buys cheaper without giving any extra effort. E-commerce systems use these systems widely to improve sales (Ricci, 2011) [23]. It helps the business by increasing its profit by attracting as many customers as possible and trying to win their loyalty to come back for acquiring other services/items according to their behavior, benefiting from the so-called Long Tail Theory[1]. Their sales would be up (Amazon), and the User will be more satisfied because they will decrease the searching cost and would be pleased with the diversity of the products that are represented. Moreover, recommendations generate a substantial amount of additional revenue for business(Malcolm et al.)[6]. The analysis in Malcolm et al. paper showed incomes would increase by introducing shoppers to a new category to continue their shopping. They noticed that in order to maintain a steady flow of direct extra revenue, the model files must be updated frequently, or the performance would fall off rapidly.

2.3 Recommendation systems in Aviation industry

The aviation industry has become one of the most important subjects in our life, and various airlines are spreading across a large number of countries. Daily a large number of flights operates in different locations for the transport of millions of peoples. One of the major problems that this industry encounter is the lack of proper recommendations systems for the User-based on the experience of the customer[11].

Many different algorithms are used in this field to solve these obstacles.[31] Tuteja used Flight Recommendation client (FRC) to recommend flights to customers on the basis of user preferences and feedbacks. This system was created to help customers to discover and select the most appropriate flights. Amadeus Company created an intelligent application that would recommend possible destinations for the customers based on business intelligence pieces of information as well as some unstructured information from the web. They used Euclidean distance and cosine similarity to find the top recommendations for the customers [2].

2.4 Challenges and problems of Recommenders

In this section, we discuss the key challenges for every Recommender. For each challenge, we introduce the problem and then present a possible solution for these problems.

2.4.1 Scalability

Scalability is yet another challenge facing current e-commerce recommendation systems. Many large websites may have millions of users nowadays, and these sites are visited repeatedly consuming resources. These same sites want to maintain the activity and responsiveness, and it should be scalable to billion of users with different preferences and habits. Many profit-making recommender systems have never handled a database that large, or they may crash from the overload. Many Artificial intelligence algorithms that work well for small-scale problems are too inefficient to be used for very large problems. So there is a need for new algorithms or different techniques that can handle very large-scale problems while maintaining accuracy.

For passing this challenge, clustering techniques can be used to scale up the neighborhood formation process[25]. hese algorithms work by identifying groups of users who appear to have similar preferences, and once the clusters are created, then we could make the prediction of a new user based on the opinion that is created by gathering all the other similar users in the neighborhood. Another method used by (Koren et al.)[12] in the Netflix price that solves Scalability was the use of matrix factorization and SVD (Singular Value Decomposition), which divides the problem into the matrix and reduce the number of features by going to lower dimensions. This method helps to find hidden patterns in lower dimensions and make the system much more scalable.

2.4.2 Data Sparsity

Another challenge that we encounter in the recommendation systems nowadays is the well-known data sparsity problem. We have to store every action that a user does with the system in a matrix where each User has an interaction with an item. However, this user-item rating matrix is very sparse(many null items) because stores/online shops have many products, and all those products will not be rated by many users. Actually, there exist very few people that frequently rate products.

In order to reduce the sparsity problem, some researchers have proposed to reward every User for providing ratings to items. Others have proposed to capture the ratings by implicitly looking at the User's behavior [25].In his research, he approached to solve the sparsity problem by using user filtering agents called filter bots or dynamic agents to automatically rate items and filled the empty values. (Papagelis et al.)[20] used a similar method called trust inferences that are associations between users in order to gather the additional source of information.

2.4.3 Cold Start

Cold start problem is a challenge that may be associated with the sparsity of the Recommender. For systems that have just established, they are facing the cold start problem where the recommender system is unable to accurately recommend items due to the fact that only a few rating has been performed on items by each User and we can not find any similarity between user/items. Being unable to store user history, we can not find user preferences[28], and therefore we cannot fill the missing values using typical matrix factorization techniques.

By using only collaborative filtering, we can not help in cold start problems cause we don't have any history about the new User. However, introducing content-based information, we can improve our Recommender to find similarities between items/users (Schein,2002)[28]. In his paper, he introduced two machine learning algorithms to evaluate the data. Another solution would be the use of a k-arms bandit in order to consider the exploration versus exploitation in new items.[18] [17]

2.4.4 Reduced Coverage

With the increasing catalog of items, it is always important to get high coverage between different items while maintaining low latency. If recommender systems rely only on items that have been rated or the popular ones, then it is missing a lot of good items for the recommendation that are hidden because no one has rated them or they just have been published in the market. This is called the Coverage metrics, which is the percentage of items for which a recommender agent can provide predictions. The long tail distribution of items describes it best that the gain of every business is more in the long tail than in popular purchase[Amazon].

A solution is by using the bandit arms[13], in which we have to try to switch between exploration and exploitation for every item. Exploitation consists of trying to represent items that we know and have information on them in the search space, while in for exploration, we are seeking to find new positions in the searching space, hoping to find a better solution. If we found a brand-new item that advertises very well, we would like to present it to users more (exploitation). But concurrently, you would also want to present others which have not been shown as much, because they can be even more popular than the items you have shown already (exploration) by using e-greedy or more sophisticated methods.

2.4.5 Shilling Attacks

Because of the availability that exists on the web, everyone can enter to the internet very easily and try to do different malicious things. For a recommender, it introduces a challenge of not understanding whether a user is real, and his rating is based on the experience he had with the items, or he is trying to game the recommendation system. These ratings are carried out in order to influence the system's behavior and have been termed "shilling" or "profile injection" attacks that can also be made for personal profit. For example, some clients can be giving a lot of negative reviews for different items in order to distance themselves from their competitors.

A method for detecting suspicious ratings based on suspicious time windows and target item analysis can be used to detect those attackers [32]. Wei et al. analyzed data streams of the rating items by using time windows and find groups abnormalities between user ratings. He identified the attackers in four groups, random, average, bandwagon, and segment attacking models. Seeing each model the distribution of the variance of ratings, he could predict the character of the User.

2.4.6 Gray Sheep

The last challenge that we will describe is the known problem of gray sheep. It is a challenge that affects similar products that are practically the same in content but different in presentation. This problem we frequently face in collaborative filtering methods since we don't use product descriptions for collaborative filtering, and we can miss the information about synonymy. Since online stores have different codes for these items, finding synonymy can be a problem and will lead to low accuracy of the prediction by the Recommender. So the pure solution with collaborative filtering would fail in this challenge because it does not analyze the content of the items. There are also gray sheep users that are unique Users with very specific tastes that affect the performance of the Recommender directly negatively. A solution would be switching to a hybrid recommender where it will use content-based features to find similar items. Gray sheep users can be identified using clustering algorithms in an offline process, where the similarity threshold can be used to isolate these users from the rest of clusters finding them empirically [8].

2.5 Similar Researches/Models on Recommender Systems

Table 2.2 will depict other's papers authors names and the techniques used to pass different challenges 2.2.

No	Author name	Techniques used	Advantages
1	Panigrahia et.al. [19]	Alternating Least Square and Clustering techniques	Sparsity,Scalability
2	Linden et.al. [14]	Item to Item Collaborative Filtering	Scalable,faster for large dataset
3	GroupLens [26]	Content based and collaborative with special techniques (feedback,filters) Open Architecture	Platform,privacy,Openness and scalable
4	Töscher et.al. [30]	Neighborhood-based algorithm with RMF(Regular matrix factorization)	Scalable, Improve speed and accuracy for estimation of unknown variable
5	Tharun et.al. [22]	Collaborative filtering approach	Solve sparsity,cold start problem and shriller attacks

6	Breese et.al. [3]	Collaborative filtering using Bayesian methods for similarity	System more accurate, probabilistic,smaller memory requirement and faster predictions. (SLOW training)
7	Koren et.al. [12]	Matrix Factorization Technique	Solve scalability problems and better accuracy (with feedbacks), flexibility with real-life situations.
8	Salakhutdinov et.al. [24]	Restricted Boltzman Machines	Outperform SVD by RMSE (Root mean square error)
9	Ghazanfar et.al. [7]	Cascading hybrid recommendation System	Eliminate problems of scalability,data sparsity, cold start and reduced coverage
10	A.Dev and Mohan [5]	Big data recommendation using Map-Reduce framework for distributed computing	High performance and parallelism

Table 2.2 Different recommenders and their advantages

Panigrahia et al.[19] created a new hybrid algorithm called User-oriented collaborative filtering where they used Dimensionality reduction techniques like Alternative Least Square and Cluster techniques in order to overcome limitations of collaborative filtering such as data sparsity and Scalability. They also tried to reduce cold start problems by correlating the User to products through features. Apache Spark was used for better computation and parallelism.

Greg Liden et al.[14] used a new Item to Item collaborative filtering that matches each of the User's purchases and the ratings of items, to similar items and then combine them to a recommendation list. This made the system to be more scalable and faster for large datasets.

GroupLens(Resnick et al.) [26] developed an open architecture that worked as a platform that would recommend news for different customers, and they use techniques like filtering and feedbacks from customers to predict ratings with some heuristic methods. Social and information filtering was done.

Toscher, Jahrer et al. [30] improved neighborhood-based algorithms for the Large-Scale system. The introduced the problem as regression, which enabled them to extract the similarities from the data, and the regular matrix factorization (RMF) was improved with neighborhood-aware techniques, which made the Recommender more scalable and more accurate.

Tharun and Nagaraju [22], in their paper, created a recommender using item-based collaborative filtering with where they identified the relationships among various items to avoid cold start and other problems.

Bresse, Heckerman, and Kadie, [3] in their paper described a collaborative filtering recommender using statistical Bayesian methods to improve accuracy for the predictions. Smaller memory use used but the training was more delayed.

A matrix factorization technique for the recommendation system was used in Koren et al.[12] and were given the winner for the Netflix price competition. These techniques allowed the incorporation of additional information such as implicit feedback, temporal effects, and confidence levels. Scalability was solved, and the Recommender was more flexible for real-life situations.

Salakhutdinov, Mnih, and Hinton [24] showed how a class of two-layer undirected graphical model, Restricted Boltzmann Machine(RBM), could be used to outperform collaborative filtering recommenders that could not handle very large data sets.

The efficient learning of each neuron leads to better performance also from the SVD algorithm.

Ghazanfar et al [7] on their paper described a Cascading hybrid recommender by applying machine learning techniques for filtering unseen information and could predict whether a user would like a given resource. They used Content-based, Collaborative filtering, and demographic Recommender to create a hybrid system. By combining those techniques, they eliminated problems like Scalability, data sparsity, cold start, and reduced coverage.

A.Dev and A.Mohan[5] introduced in their paper Map-Reduced techniques that work with big data. The system was distributed, and they used a set-similarity join to provide customized and personalized item recommendations to the User.

2.6 Visualization for Recommender System

Data Visualization is often one of the main milestones that we have to think while performing a variety of analytical tasks. Because of the large datasets and information out there, there is a need for tools that can support visual concepts and analyze the data. To acquire knowledge, the pursuit of information need to be undertaken. There is very little research in the data visualization, and we encounter them in movie catalogs, for example, Netflix, where User can see their recommendations by sliding into them, but there is a need for a more sophisticated tool being more user-friendly than before.

The goal of a visualization recommender system is to search into data for interesting trends and patterns to speed up data analysis(Illinsky, 2011)[9]. These patterns may be then presented to the User at different stages of analysis, for example, when they first entered the system, while performing some task, or viewing a particular visualization.

Tamara Munzner [16] made a 3-step model for data visualization design. According to the model that was described in the research, the first step that we need to solve is to decide what we want to show to the User. Secondly, we need to explain why we want to show it by providing implicit and explicit reasons, and finally, we need to decide how we are going to represent it. (Munzner and Maguire, 2015).

Chapter Three

Methodology

Contents

3.1 Methodology Overview **22**

3.2 Problem Formulation **23**

3.3 Data Collection **25**

3.4 Data Visualization and REST API **25**

3.5 Building the Recommender **28**

3.5.1 Collaborative filtering model 28

3.5.2 Content-Based model 29

3.5.3 Hybrid model 30

3.6 Prediction Generation **31**

3.1 Methodology Overview

In this section, we will describe the methodology that we used in the recommendation system. We are using the weighted model as a hybridization method where the coefficients that are used for every algorithm are weights that affect the impact of the Recommender. The system will use Collaborative filtering methods, both user-to-user CF and item-to-item CF, and the metric that will be used to measure the similarity will be cosine similarity. On the other hand, to avoid different problems related to recommendation systems like cold start and data sparsity, we will combine collaborative techniques with Content-based methods. Users will be represented as a vector providing their preferences, and another vector will be constructed for the item combined with a description that shows the category where it belongs. The Recommender will be dynamically changed based on the coefficients described above that will multiply each result that comes from every technique. These coefficients depend on the sparsity of the Dataset. Our experiment is implemented using Python and ran on a Windows PC with the Intel Core i7 processor having a speed of 2.9GHz and RAM of 4GB. In order to find the best performance of the hybrid technique, there are three experiments performing different combinations of the method used on content-based filtering (CB) and item-based collaborative filtering (CF). The algorithm that was used is provided by a surprise library for the implementation of the Collaborative filtering part, and for the API, we used the SWAGGER framework that is combined with a flask framework for back-end connection and to handle all the request.

3.2 Problem Formulation

The research approach provides a solution in the recommendation system in the Icarus platform. Our solution consists of gathering the data from the platform. We analyzed those data to extract user preference, and the objective was to provide various recommendations that would satisfy the User. We had different inputs to analyze user behaviors through the interactions with the system and also user and item categories. By collecting those features, we created a utility matrix for each user-item combination, and through different machine learning algorithms, we would calculate each score and provide the best match for the customer.

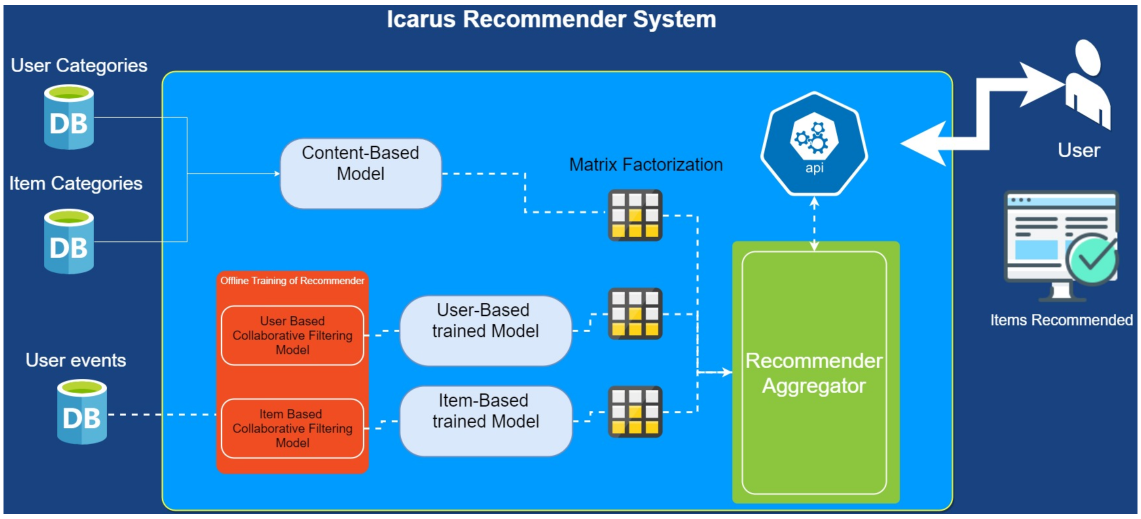


Figure 3.1 A diagram which describe Icarus recommender

As shown in the diagram above, Icarus Recommender has divided into three main parts the first part of the offline training, the second part of the aggregation of each technique and the last part the prediction and the recommendation of items in the form of Restful API that will be represented to the User.

In the first part of the Recommender, the system will gather data from the files where user interactions with the items are presented. The Dataset will have different details describing each rate that the User does on a scale from 0 to 5. First, data will be processed, and after that, the training will begin. The algorithm that we will be using for the Collaborative Filtering will be KNN(K- nearest neighbor), and we decided to use this algorithm to solve the problem of Scalability, for the system to be able to handle large datasets and to have a better understanding on how the algorithm works.

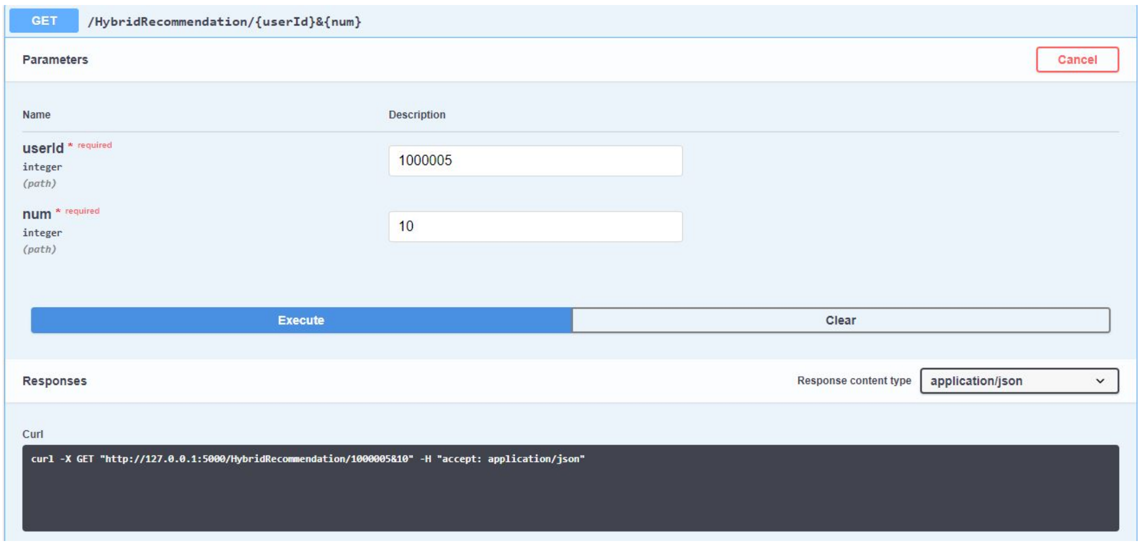
The phase two of the Recommender starts as soon as the training has been completed. Also, in this phase, the content-based algorithm will start, and we will perform cosine similarity in the two vectors to create the matrix that will contain user/items and their similarity. Each category will have a score that has a range from 0 to 1. All the results will arrive in the aggregator, where is the process where Hybridization takes part. Each matrix will be multiplied with a coefficient that will be related to the sparsity of the Dataset. After that, every User will use the API that is created, and the Recommender will return the best results for that User.

3.3 Data Collection

The data will be provided by the Icarus platform. They are represented in JSON format in order to be simple and make the preprocessing faster. Icarus will provide both user interactions with the system and also the category of the profiles. By taking these categories, we will create the vector needed for the content-based methods. From the metadata, we filtered out many unnecessary features, and we created a matrix where each User would provide the rating for a particular item. In our case, the User could be a company that had provided that Dataset (item), or a simple user that liked the item, added to the favorite list, or proceed to checkout. Python was used to run the program and JSON libraries along with NumPy and scipy to create the matrices and vectors representations of users and items.

3.4 Data Visualization and REST API

For the visualization of the data, we decided to use a Rest API that will be a black box whose implementation details are unclear and can be called to recommend different items from the User. It will help both the developers because there is no need to install additional software or libraries when creating the API, and also it will provide a great deal of flexibility. We know that a Rest API is stateless, so it doesn't need any resources or methods to run, and for this reason, it can handle multiple user types of calls and return even different data formats(GET, POST, etc.). Also, the system is layered, so different sections will be working together to build a hierarchy. It makes the system much more scalable. Our API was build using Flask and Swagger that are two open-source frameworks. The first one we used to create the Rest API by handling all the recommendations methods while the second framework was used to develop the documentation of it.



/HybridRecommendation/{userId}&{num}

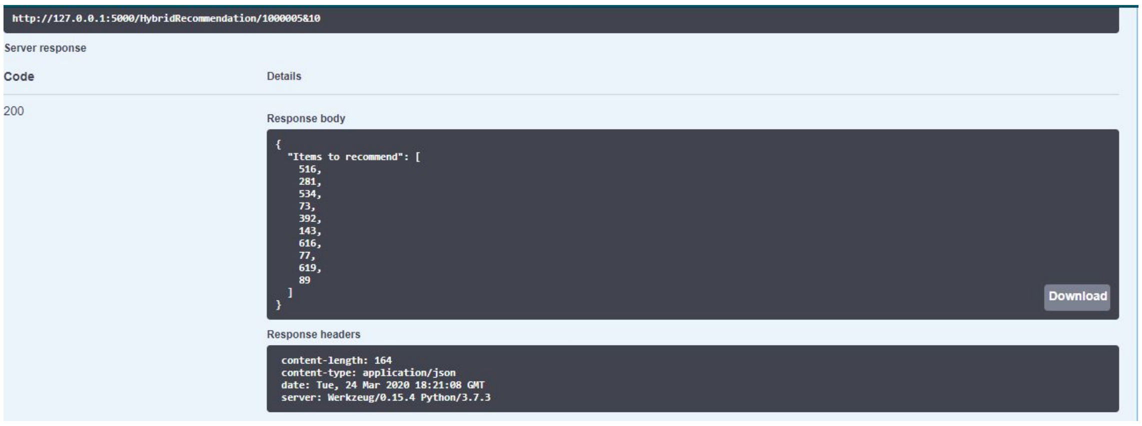
Parameters

Name	Description
userId * required integer (path)	1000005
num * required integer (path)	10

Curl

```
curl -X GET "http://127.0.0.1:5000/HybridRecommendation/1000005&10" -H "accept: application/json"
```

(a) Hybrid API design



```
http://127.0.0.1:5000/HybridRecommendation/1000005&10
```

Server response

Code	Details
200	Response body

```
{
  "Items to recommend": [
    516,
    281,
    534,
    73,
    392,
    143,
    616,
    77,
    619,
    89
  ]
}
```

Response headers

```
content-length: 164
content-type: application/json
date: Tue, 24 Mar 2020 18:21:08 GMT
server: Werkzeug/0.15.4 Python/3.7.3
```

(b) Hybrid API recommendations

Figure 3.2 Representation of Restful API

We also used Elasticsearch, an search engine library where we loaded our JSON dataset. For the visualization of the data, Kibana was used, which is an open-source data visualization dashboard for Elasticsearch. We visualized below Icarus platform dataset and summarized three different figures.

In the figure below, we can see that most of the purchases that happened on the platform were via bank transfer. The second most popular method was via credit or debit card.

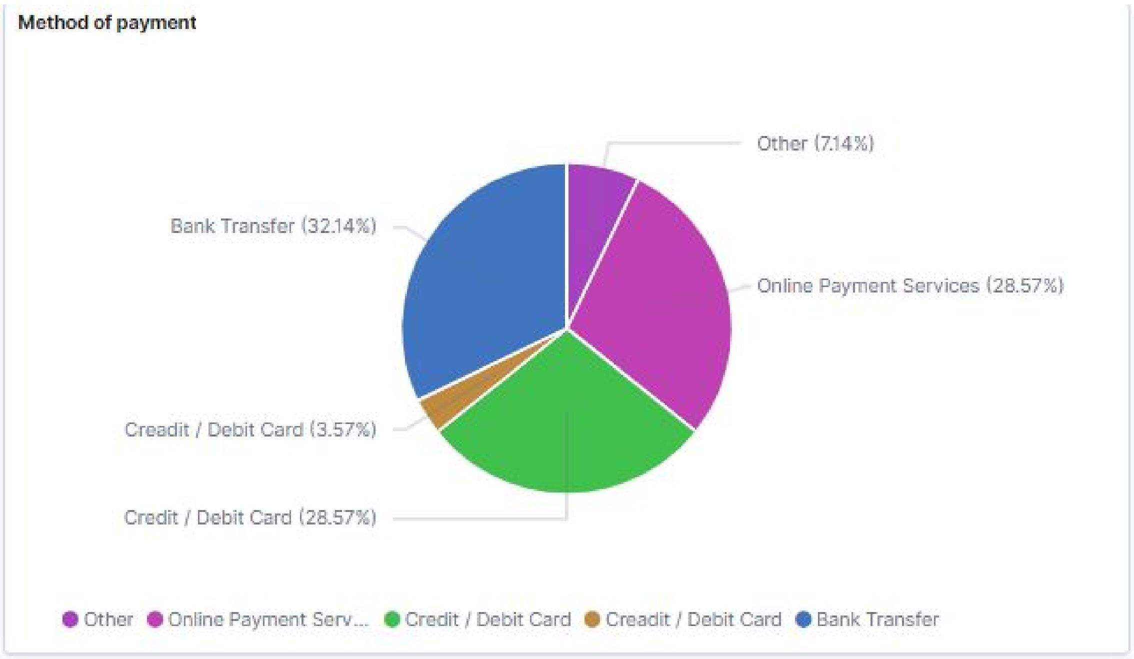


Figure 3.3 Method of payment

The figure below we visualize the target purpose of each item in the Dataset. We notice that business is the most popular purpose around 55% while the non-profit target is the least one around 6%

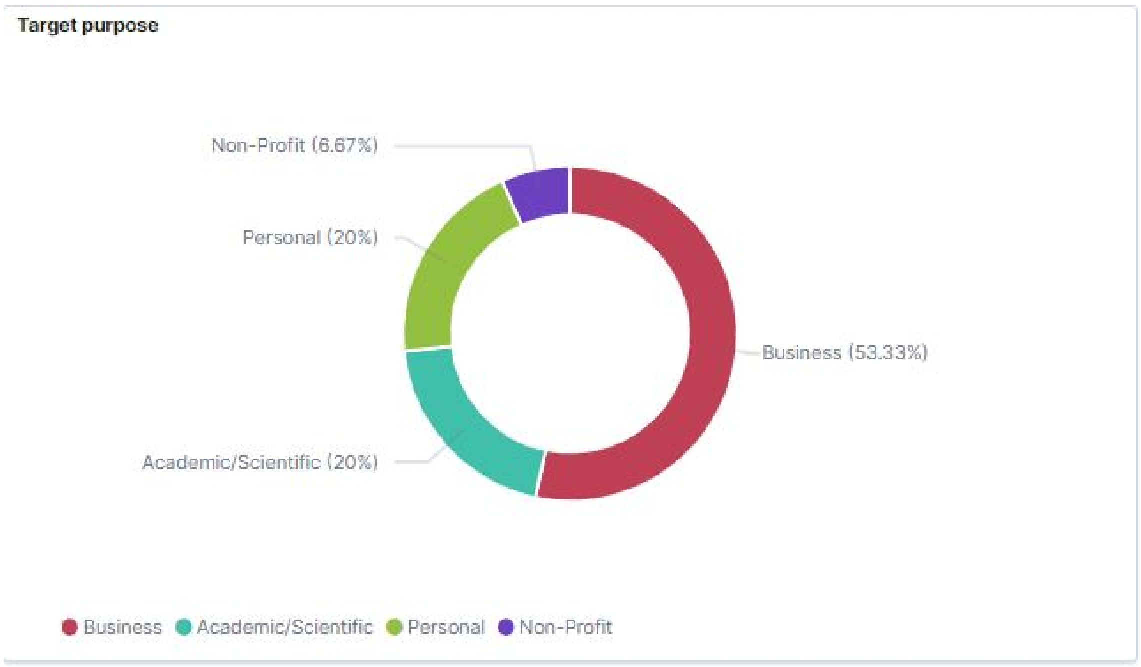


Figure 3.4 The target purpose of the dataset

3.5 Building the Recommender

3.5.1 Collaborative filtering model

The main technique that is used in our Recommender will be the Collaborativefiltering method using k-Nearest Neighbors(kNN). This is a machine learning algorithm that tries to find clusters of similar users based on common item ratings and makes predictions based on the users of the top-k nearest neighbors. The reason why we decided this algorithm is that it makes the system much more scalable, and it performs quicker then SVD or some other algorithms. We used a surprise library, which is a python scikit for recommender systems. To find k-Nearest neighbors, we tried the kNN basic algorithm.
The kNN Basic is the simplest algorithm that we decided to start the training in the beginning, and it takes into account the similarities between users or items, which can be switched by giving a parameter True/False at the beginning of the model. This algorithm essentially boils down to forming a majority vote between the K most similar instances to a given "unseen" observation. It uses cosine similarity (see formula below) between neighbors and return the average rating as a prediction for the given item. After we run some experiments on the kNN family algorithms, we noticed that kNN baseline gave us better performance, so we decided to go with this method for our collaborative filtering recommender. Generally, it is the same algorithm, but this technique takes into account a baseline rating. This approach uses heuristics, simple summary statistics, randomness, and machine learning to create predictions for a dataset.

The collaborative filtering algorithm has three steps. The first step is to profile every User or item if we are using item-based CF in order to find which of them are similar to the target, the second step is to gather all the items selected by all the neighbors and associate a coefficient that will perform like weights that represent the importance of the item, and the last step is to present the recommendations that have the highest score. The main step in this technique is the first step where the neighborhoods are created, and where the accuracy and Precision rely on.

$$\hat{r}_{ui} = \frac{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v) \cdot r_{vi}}{\sum\limits_{v \in N_i^k(u)} \text{sim}(u,v)} \qquad (3.1)$$

Figure 3.5 k-NN basic prediction equation

$$CosineSim(u,v) = \frac{\sum\limits_{i \in I_{uv}} r_{ui} \cdot r_{vi}}{\sqrt{\sum\limits_{i \in I_{uv}} r_{ui}^2} \cdot \sqrt{\sum\limits_{i \in I_{uv}} r_{vi}^2}} \qquad (3.2)$$

Figure 3.6 Cosine similarity equation

3.5.2 Content-Based model

The content-based algorithm is based on the similarity between items and users. In this approach, users and items are represented as vectors in the m dimensional user/item space where this is related to the number of users and the number of items that we have in the platform. Cosine similarity is used to measure the similarity of an inner product space of those vectors, and it computes the cosine of the angle between them. As shown in figure 3.4 below, the smaller that angle, the more similar they are, and we try to find the best matches for our recommendations.

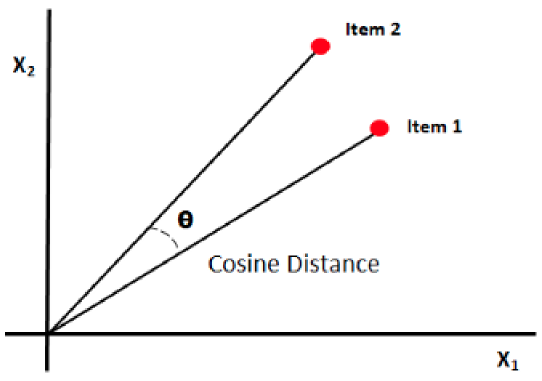
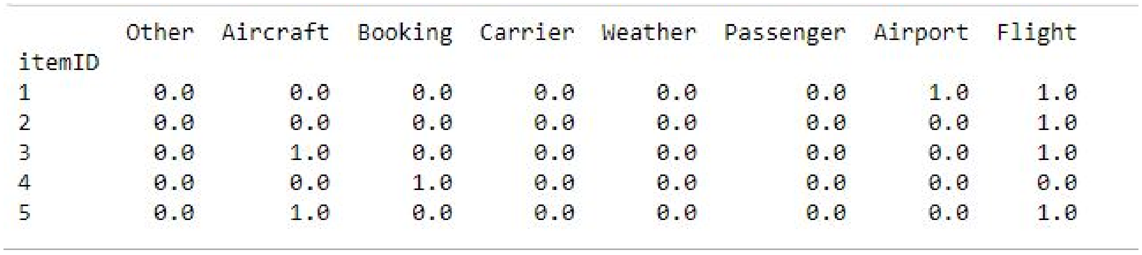


Figure 3.7 Vector representation of items.
The more θ is small,the more similar are the items

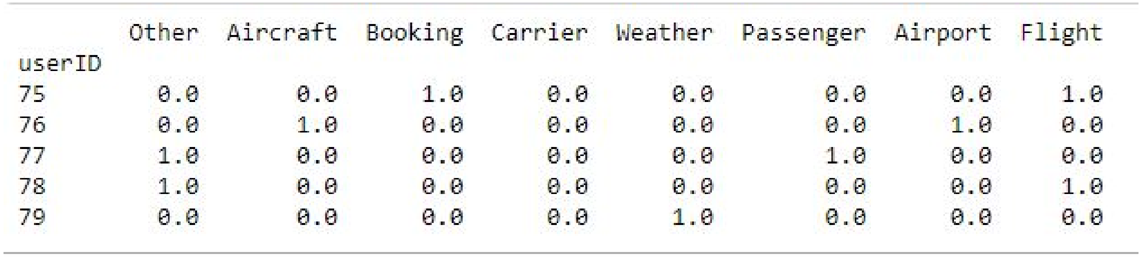
Here describe how the vectors work.

First, the data are gathered and preprocessed. From the preprocess, we filter only the userID and the category of the items that the user likes. This is also done for each item where we get the itemID and the category where it is placed. After the data has been gathered, we create a matrix representation where each User represented as a vector with his id and the category that he prefers, and the value is set to 1. After vector creation, we use cosine similarity method provided by sklearn and find all the best matches for every User. The values vary from -1 to 1, with the best score be 1 if the items or users will be identical.



```
        Other  Aircraft  Booking  Carrier  Weather  Passenger  Airport  Flight
itemID
1         0.0       0.0      0.0      0.0      0.0        0.0      1.0     1.0
2         0.0       0.0      0.0      0.0      0.0        0.0      0.0     1.0
3         0.0       1.0      0.0      0.0      0.0        0.0      0.0     1.0
4         0.0       0.0      1.0      0.0      0.0        0.0      0.0     0.0
5         0.0       1.0      0.0      0.0      0.0        0.0      0.0     1.0
```

(a) Item vector category



```
        Other  Aircraft  Booking  Carrier  Weather  Passenger  Airport  Flight
userID
75        0.0       0.0      1.0      0.0      0.0        0.0      0.0     1.0
76        0.0       1.0      0.0      0.0      0.0        0.0      1.0     0.0
77        1.0       0.0      0.0      0.0      0.0        1.0      0.0     0.0
78        1.0       0.0      0.0      0.0      0.0        0.0      0.0     1.0
79        0.0       0.0      0.0      0.0      1.0        0.0      0.0     0.0
```

(b) User vector category.

Figure 3.8 Representation of User/Item vectors

3.5.3 Hybrid model

The methods that were described above, the collaborative filtering and content-based, are great techniques that are used a lot nowadays, but both have their faults. Content-based algorithms have flaws when presenting a recommendation for the User and items that don't have any history on the system and their recommendations have a lack of diversity while Collaborative filtering has flaws when the matrix of events is very sparse that means we don't have user interactions into the system and also in handling new users (Cold Start). To solve these challenges, we decided to use a hybrid model for Icarus Recommender. More specifically, the method of Hybridization that we used is the weighted hybrid strategy. This method combines the recommendations of multiple systems by computing weighted sums of their scores. In more detail, we would take the results that our recommenders would provide separately, and we introduced a coefficient, α, that would generate a function that can be changed anytime. This coefficient act as a weight where it multiplies the result that comes from the content-based method and with (1-α) that will multiply the result that would come from the Collaborative-filtering methods. The result, in the end, will merge together, creating a matrix with users and items where we will measure the similarity, and we would present to the User the top recommendations. We decided this coefficient to be dependent on the sparsity of the matrix in user interactions, and we made this in order for the system to be changed dynamically and to give us the best performance. By doing this, we represent a solution to all the problems that we discussed above.

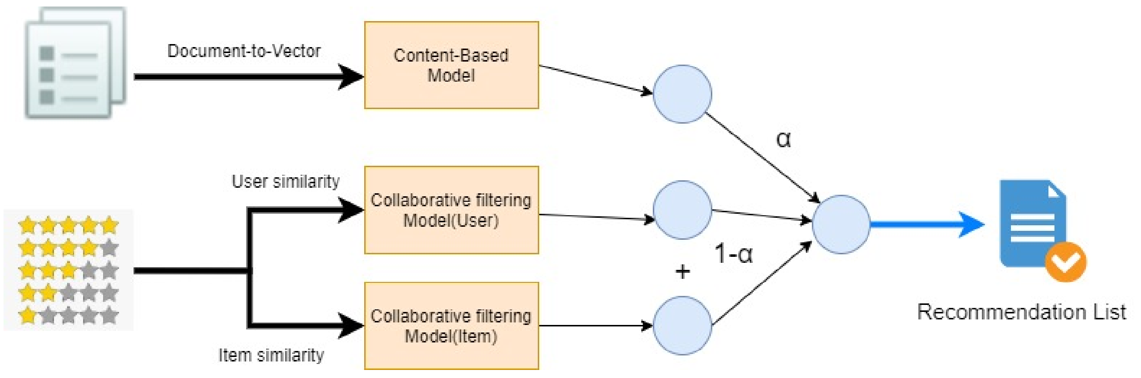
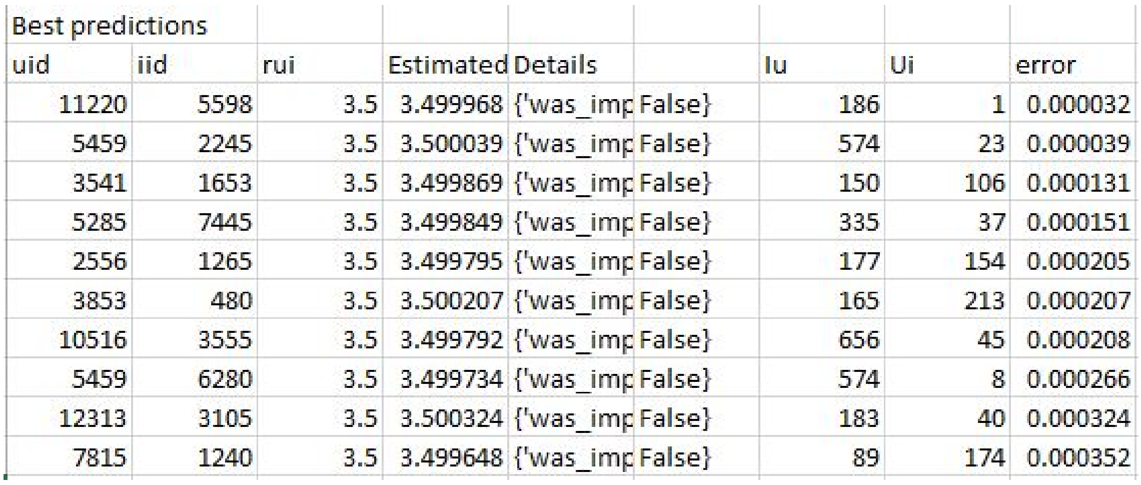


Figure 3.9 The diagram of Icarus Hybrid model

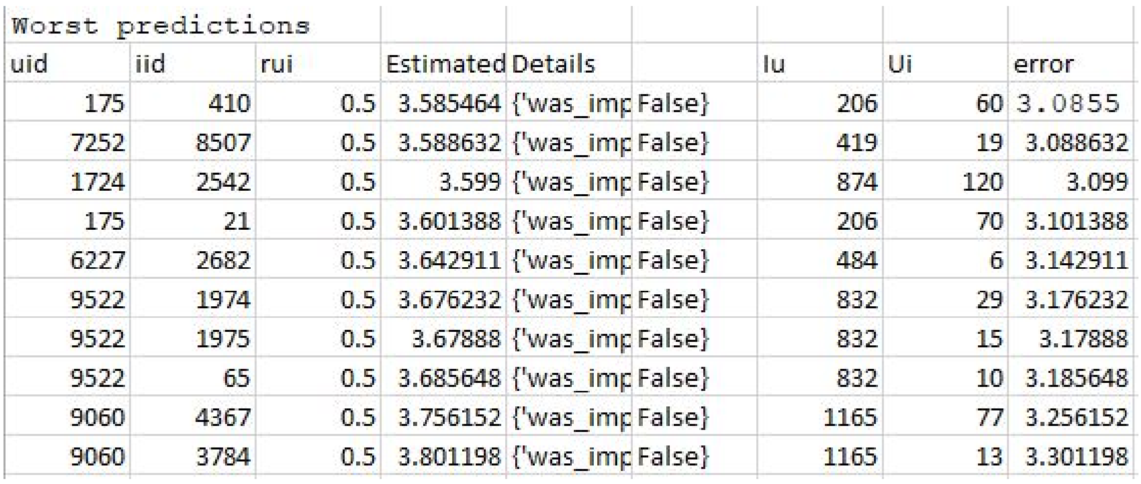
3.6 Prediction Generation

We split the dataset in training and testing, more specifically in 75% of the data in training and the remaining 25% in testing. We used the train_test_split method from the surprise library. The data are shuffled to make the prediction more realistic, and we use k-fold to split into sections (fold=5). By using models that have been built, the performance can be evaluated by predicting user ratings on the testing set. We will represent the user rating prediction, and along with that, we added an extra column that is the real rating of the User for specific items. The estimated value is scaled from 0 to 5 to remove any unnecessary noise. The evaluation metric used is the Mean Absolute error(MAE). RMSE is an error measurement comparing the real rating value and the prediction score. We expect the value of RMSE as small as possible. The value of zero means that no error exists in our prediction, and we found with accurate user rating. Below are the representation of the top 10 best and worst predictions.



Best predictions								
uid	iid	rui	Estimated	Details		Iu	Ui	error
11220	5598	3.5	3.499968	{'was_imp	False}	186	1	0.000032
5459	2245	3.5	3.500039	{'was_imp	False}	574	23	0.000039
3541	1653	3.5	3.499869	{'was_imp	False}	150	106	0.000131
5285	7445	3.5	3.499849	{'was_imp	False}	335	37	0.000151
2556	1265	3.5	3.499795	{'was_imp	False}	177	154	0.000205
3853	480	3.5	3.500207	{'was_imp	False}	165	213	0.000207
10516	3555	3.5	3.499792	{'was_imp	False}	656	45	0.000208
5459	6280	3.5	3.499734	{'was_imp	False}	574	8	0.000266
12313	3105	3.5	3.500324	{'was_imp	False}	183	40	0.000324
7815	1240	3.5	3.499648	{'was_imp	False}	89	174	0.000352

(a) Top 10 Best Predictions



Worst predictions								
uid	iid	rui	Estimated	Details		Iu	Ui	error
175	410	0.5	3.585464	{'was_imp	False}	206	60	3.0855
7252	8507	0.5	3.588632	{'was_imp	False}	419	19	3.088632
1724	2542	0.5	3.599	{'was_imp	False}	874	120	3.099
175	21	0.5	3.601388	{'was_imp	False}	206	70	3.101388
6227	2682	0.5	3.642911	{'was_imp	False}	484	6	3.142911
9522	1974	0.5	3.676232	{'was_imp	False}	832	29	3.176232
9522	1975	0.5	3.67888	{'was_imp	False}	832	15	3.17888
9522	65	0.5	3.685648	{'was_imp	False}	832	10	3.185648
9060	4367	0.5	3.756152	{'was_imp	False}	1165	77	3.256152
9060	3784	0.5	3.801198	{'was_imp	False}	1165	13	3.301198

(b) Top 10 Worst Predictions.

Figure 3.10 Top 10 Predictions of the Recommender

Uid(userId), iid(itemId), rui(rating of user for item) , Estimated(predicted rating) and the error.

Chapter Four

Evaluation

Contents

4.1 Recommender performance **34**

4.2 Comparisons **37**

4.2.1 Experiment Setup 37

4.2.2 Representation of data 38

4.2.3 Results and Comparison 39

4.1 Recommender performance

To measure the performance of our Recommender, we decided to evaluate it through several metrics. There are three categories in total that we evaluated, and those are Prediction accuracy metrics, Decision support metrics, and some non-traditional metrics.

Prediction accuracy metrics are techniques that we use to measures how close our prediction is to the real value. For those measures, we used MAE (Mean absolute error), MSE (Mean squared error) and RMSE (root mean squared error).

Mean absolute error measures the average difference of the error in the set of predictions, and because we have the use of absolute, we don't consider the direction of the error. It is the average over the sample of the absolute difference between predictions and actual ratings.

$$\text{MAE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} |r_{ui} - \hat{r}_{ui}|$$

On the other hand, we have the MSE and RMSE that are quite similar in their equations. The difference with MAE is that they use quadratic scoring rules where MSE squares the error in the predictions, and RMSE introduces the squared root of the average squared magnitude.RMSE method has the benefit of penalizing large errors, so sometimes, it gives us better information about the performance of the Recommender.

$$\text{MSE} = \frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2. \quad (4.1)$$

$$\text{RMSE} = \sqrt{\frac{1}{|\hat{R}|} \sum_{\hat{r}_{ui} \in \hat{R}} (r_{ui} - \hat{r}_{ui})^2}. \quad (4.2)$$

The second category of the measures that we used is the decision support metrics. Their focus is on the correct classification or differentiation between right and wrong predictions. In other words, we are not only interested in whether the Recommender properly predicts the ratings but whether the system will predict that the User will prefer this item in the future. We have 2 main concepts here the Precision and the recall, and then we calculate f1-score.

$$\text{Precision} = \frac{\sharp \text{ True-Positive(TP)}}{\sharp \text{ True-Positive(TP)} + \sharp \text{ False-Positive(FP)}} \quad (4.3)$$

$$\text{Recall} = \frac{\sharp \text{ True-Positive(TP)}}{\sharp \text{ True-Positive(TP)} + \sharp \text{ False-Negative(FN)}} \quad (4.4)$$

In our case for the recommendation system, these concepts would be presented as Precision, the number of our recommendations that are relevant divided by the number of all the items that were recommended and for recall, the number of recommendations that are relevant divided by all possible relevant items. To simplify, Precision will count how many items did users like from all the recommendations and recall, will count what proportion of items that the User liked were actually recommended.

$$\text{Recommender system precision:} \qquad P = \frac{\sharp \text{ of our recommendations that are relevant}}{\sharp \text{ of items we recommended}}$$

$$\text{Recommender system recall:} \qquad r = \frac{\sharp \text{ of our recommendations that are relevant}}{\sharp \text{ of all the possible relevant items}}$$

F1-score is simply related to the above equations and carries the balance between Precision and recall and is calculated as the equation below shows.

$$\text{F1-score} = \frac{2 * (Precision * Recall)}{Precision + Recall}$$

The last metrics that we tried are two non-traditional methods that include coverage and novelty. With coverage, we try to show the percentage of items or users that the model is able to recommend. The higher the value of coverage, the better the system.

$$\text{coverage} = \frac{I}{N} \times 100\%$$

Novelty, on the other hand, shows how many unknown items are to a user. If the value of novelty is high, then this means that less popular items are being recommended and the contrary.

$$novelty = \frac{1}{|U|} \sum_{\forall u \in U} \sum_{\forall i \in topN} \frac{\log_2(\frac{count(i)}{|U|})}{|N|}$$

Figure 4.1 Novelty of Recommender

Personalization is the dissimilarity between User's lists of recommendations. We used this metric to study our recommendations and provide a score. A high score indicates User's recommendations are different, whether a low personalization score indicates User's recommendations are very similar.

We used those metrics to evaluate our Recommender for different models. We trained different collaborative filtering algorithms, and we measured the performance of every technique to select the best algorithm that best suits the Icarus recommender.

4.2 Comparisons

4.2.1 Experiment Setup

One of the main contributions that we made is to compare different types of algorithms, mainly in the collaborative filtering method that will be used later to create the hybrid model. Therefore we trained every model separately with the same Dataset to see each model performance. We used the algorithms that the Surprise library enables us, and for each technique, we used the Cross-validation method to estimate the skills of each model for measuring the performance of Recommender for unseen data. After the result, we decided to take some extra evaluation for the kNN Baseline, SVD, and BaselineOnly algorithms for better understanding. Moreover, we have to mention that every experiment takes place at my personal computer, which is composed of 2-cores Hyper-threading and 4-threads CPU clocked in 2.5 GHz to 3.1 in turbo mode, 4 Gb RAM, and Windows 10 operating system. From the software perspective, we used Jupyter Notebook, which helps us to write Python programming language. Surprise library used to perform every recommendation technique and sklearn for measuring the similarity for the content-based method. The matplot library was used for the representation of the graph and the metrics.

4.2.2 Representation of data

Before we run our experiments for the evaluation of the system, we decided to study our Dataset and see if we can extract some features. The first thing that we did was to see the long-tail graph. It shows the distribution of ratings or popularity among every item in our set. In the x-ax, we have the items, and on y-ax, we display the popularity of the items. Overall, it can be seen that the number of popular items is related smaller than the unpopular ones, with respectively 548 and 6168 items. The long-tail graph shows us some valuable information about diversity and sparsity. It shows us that moving to the right will bring us more diversity among items, and for sparsity, it means that the right side would have a sparse rating matrix, and the more we go to the right, the worse the recommendation will be.

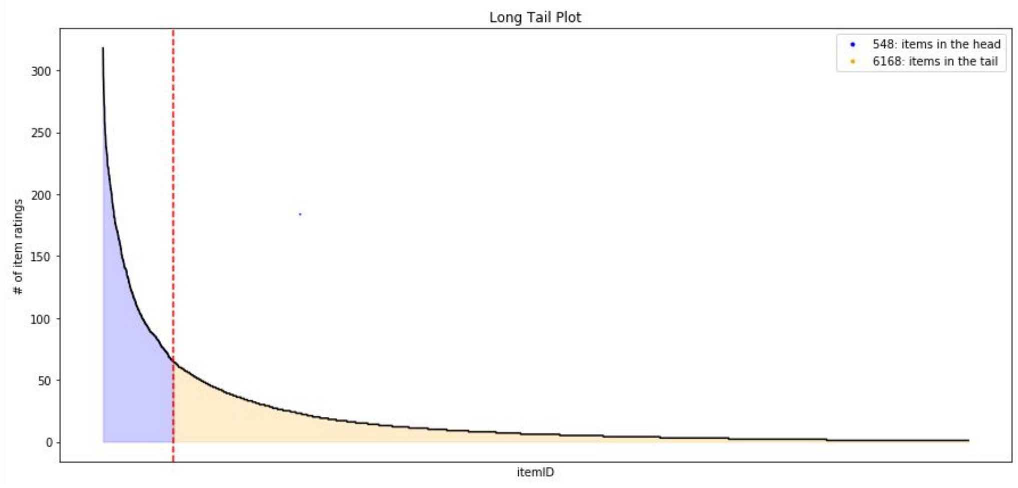


Figure 4.2 Long tail for popular and nonpopular items

Below we represent a histogram for every rating that we have in the user-item interaction dataset. We see an even distribution of ratings among items where the most popular rate would be 4.0, which means that the user/company have liked the item and have bought it.

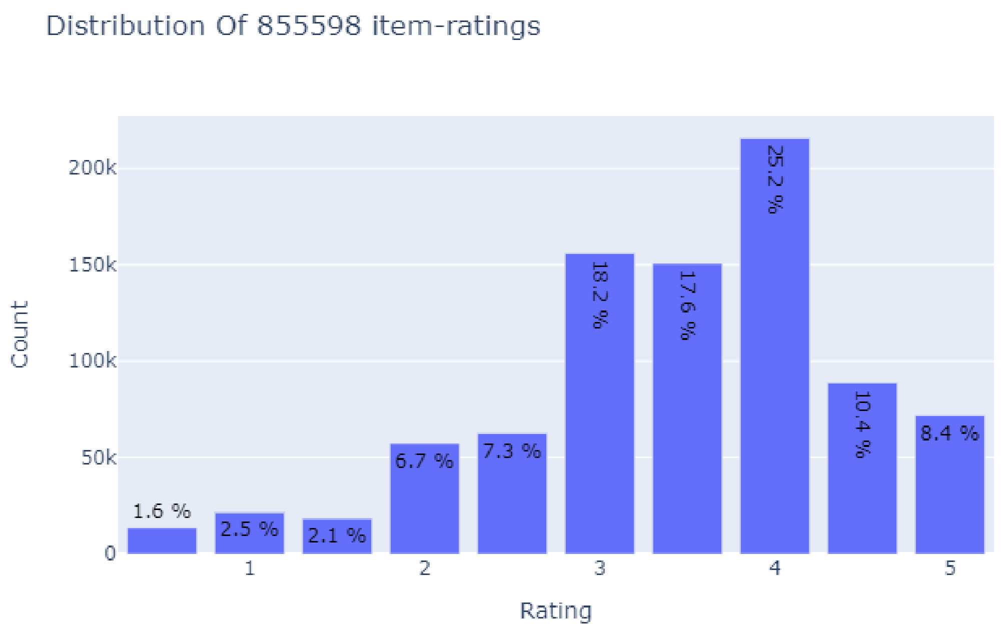


Figure 4.3 Distribution of all ratings from users

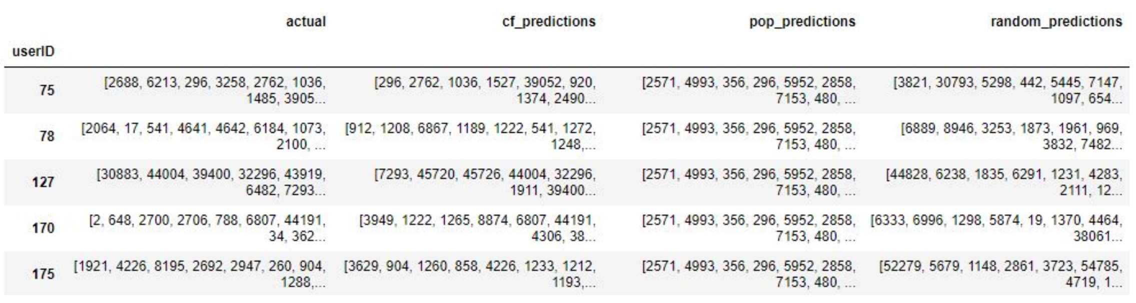
4.2.3 Results and Comparison

In this section, we present the results of our research by providing a table that summarizes the performance of each collaborative filtering, where we measured the RMSE, the time that is needed for the training of the model, and test time. Moreover, we will present different metrics that were performed on the kNN Baseline algorithm that we choose for our system, but also some of the best algorithms that gave us better performance in table 1.1. Overall, it can be seen that we decided to go with the kNN baseline method because it gave almost similar results with SVDpp in RMSE error, but it required much less time for the training and testing of the model, and by creating neighborhoods we made the system scalable. By using the kNN baseline, we could have had a better image of how the recommendations of the hybrid system are created and because this technique doesn't hide any information like the SVD method and we could change the system dynamically by switching from collaborative filtering to content-based method depending on the sparsity of the Dataset. Also, the use of kNN made it easy the combination of Collaborative filtering with Content-based because the matrix we're the same, and the method for comparing similarities was cosine similarity, so we did not have to change anything in the content-based method.

Algorithm	**RMSE**	**FIT_TIME**	**TEST_TIME**
SVDpp	**0.812378**	1465.699063	43.294074
KNNBaseline	0.823541	0.625860	12.269759
SVD	0.825586	12.993089	0.786548
BaselineOnly	0.828726	**0.117569**	**0.380656**
KNNWithZScore	0.82711	0.612947	10.401690
KNNWithMeans	0.836682	0.496899	9.412825
SlopeOne	0.841448	12.199456	36.792021
NMF	0.859715	13.379431	0.493514
KNNBasic	0.877670	0.466622	9.115010
CoClustering	0.889319	3.035781	0.479544
NormalPredictor	1.367177	0.171363	0.469020

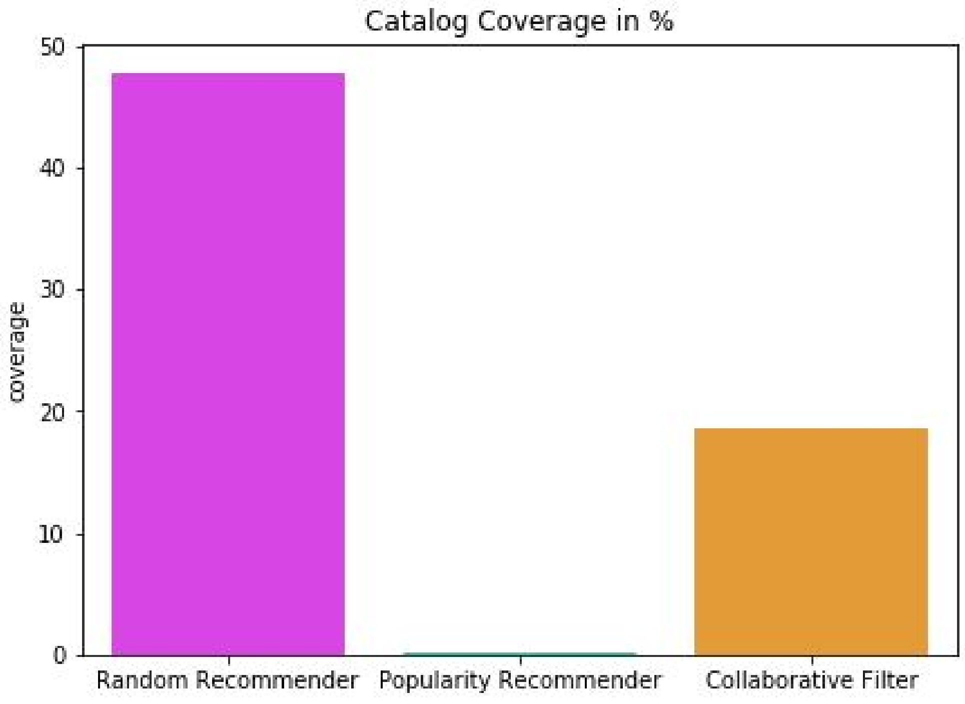
Table 4.1 Evaluation of different Algorithms

After the model was tested with the knn Baseline method, we tried to run some analyses in the predictions of the items. For this experiment, we decided to measure the coverage of the Recommender by comparing the system with 2 other methods of predictions that use the recommendation of the popular items and the other method that picks random items for recommendation. As we can see in the figure above, the catalog coverage for the recommendation of the popular item is the smallest while the random Recommender has 100% coverage. Our Collaborative filtering is something between in the middle that means that the system is able to cover 30% of our items in the catalog.



userID	actual	cf_predictions	pop_predictions	random_predictions
75	[2688, 6213, 296, 3258, 2762, 1036, 1485, 3905...	[296, 2762, 1036, 1527, 39052, 920, 1374, 2490...	[2571, 4993, 356, 296, 5952, 2858, 7153, 480, ...	[3821, 30793, 5298, 442, 5445, 7147, 1097, 654...
78	[2064, 17, 541, 4641, 4642, 6184, 1073, 2100, ...	[912, 1208, 6867, 1189, 1222, 541, 1272, 1248,...	[2571, 4993, 356, 296, 5952, 2858, 7153, 480, ...	[6889, 8946, 3253, 1873, 1961, 969, 3832, 7482...
127	[30883, 44004, 39400, 32296, 43919, 6482, 7293...	[7293, 45720, 45726, 44004, 32296, 1911, 39400...	[2571, 4993, 356, 296, 5952, 2858, 7153, 480, ...	[44828, 6238, 1835, 6291, 1231, 4283, 2111, 12...
170	[2, 648, 2700, 2706, 788, 6807, 44191, 34, 362...	[3949, 1222, 1265, 8874, 6807, 44191, 4306, 38...	[2571, 4993, 356, 296, 5952, 2858, 7153, 480, ...	[6333, 6996, 1298, 5874, 19, 1370, 4464, 38061...
175	[1921, 4226, 8195, 2692, 2947, 260, 904, 1288,...	[3629, 904, 1260, 858, 4226, 1233, 1212, 1193,...	[2571, 4993, 356, 296, 5952, 2858, 7153, 480, ...	[52279, 5679, 1148, 2861, 3723, 54785, 4719, 1...

(a) Different methods of predictions



(b) Coverage of different predictions

Figure 4.4 Coverage of the Recommenders

We measured the novelty and the personalization of our Recommender, and we noticed that our collaborative filtering method had values similar to the Random Recommender, which means that the system is able to propose novel and unexpected items that User didn't know. Also, we measured personalization that measures the dissimilarity between user recommendation lists that varies in scale from 0 to 1, and we noticed that we had around 0.95 score, which means that the recommendations were very different for each User.

Novelty	**Value**
Random Novelty	3.0858
Popular Novelty	0.4835
Collaborative Filtering Novelty	1.9108
Personalization	0.9475

Table 4.2 Evaluation of Novelty and Personalization

Here we perform another experiment to measure the average recall among the recommenders that were mentioned above. As we can see recall in random and popular Recommender is near to 0 while collaborative filtering shows us the opposite which means that around 30% of our recommendations were actually liked by the user.

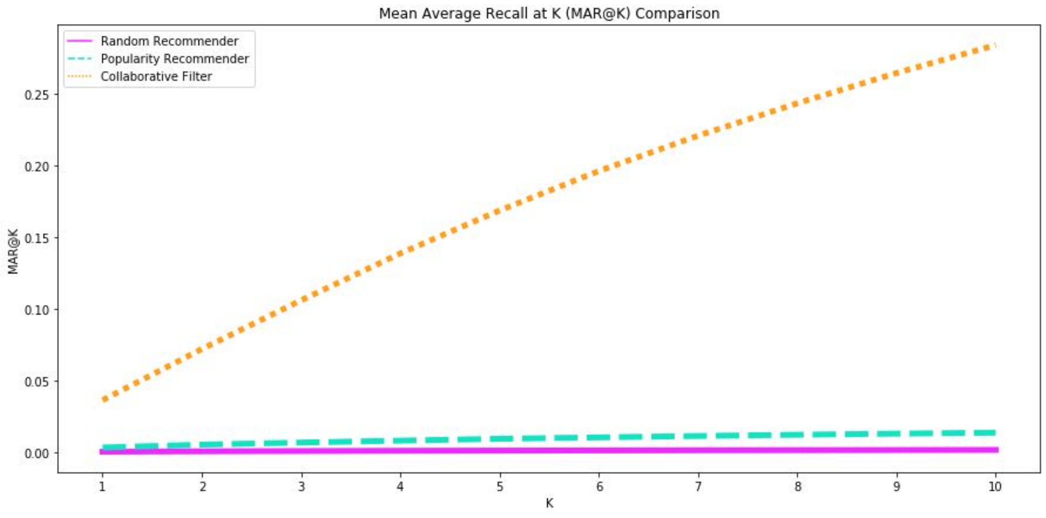


Figure 4.5 Mean Average recall for different predictions(k = max number of predicted items)

We decided to run some experiments with all the knn methods to see how each technique would perform with our Dataset. We run the experiment with different k neighbors from 15 to 60, and we use the cosine technique for the similarity. We notice that the number of neighbors increases, the RMSE decreases. This happens because the algorithm increases the generalizability at the cost of variance, and the system will create a smoother function for the classification. The model has more neighbors and is more complex by increasing this parameter.

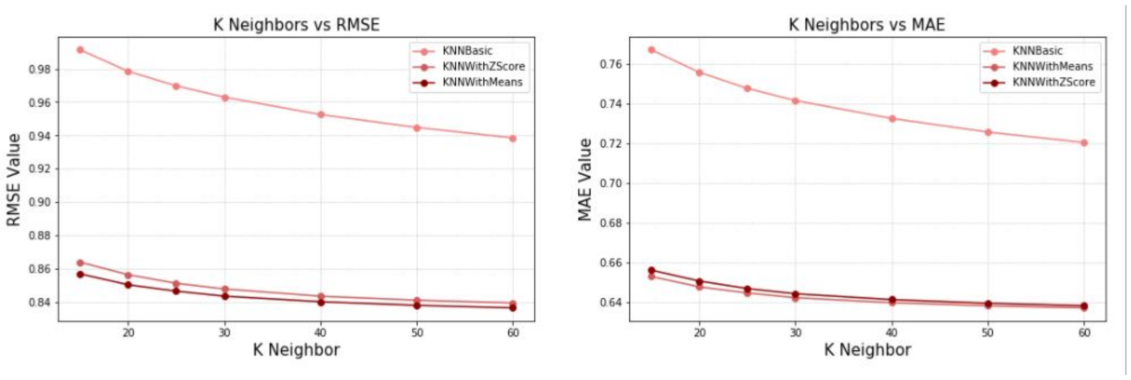


Figure 4.6 Comparisons of algorithm for different neighbors

Here we present distribution of rating in 3 different algorithms. We notice that knn has the prediction accumulated in the center of the ratings, and this because it creates clusters of neighbors, and we know that most of the ratings in our Dataset are accumulated in the range between 3 and 4.5 stars. The other two algorithms SVD and Baseline, have better distribution and can also handle other extreme ratings, but we are not very interested in extremities. In our research, we were interested in the recommendations that are liked by the users in our case companies, and we wanted them to find similar datasets that are in their preferences, so it seemed right to use kNN algorithm to create neighborhoods.

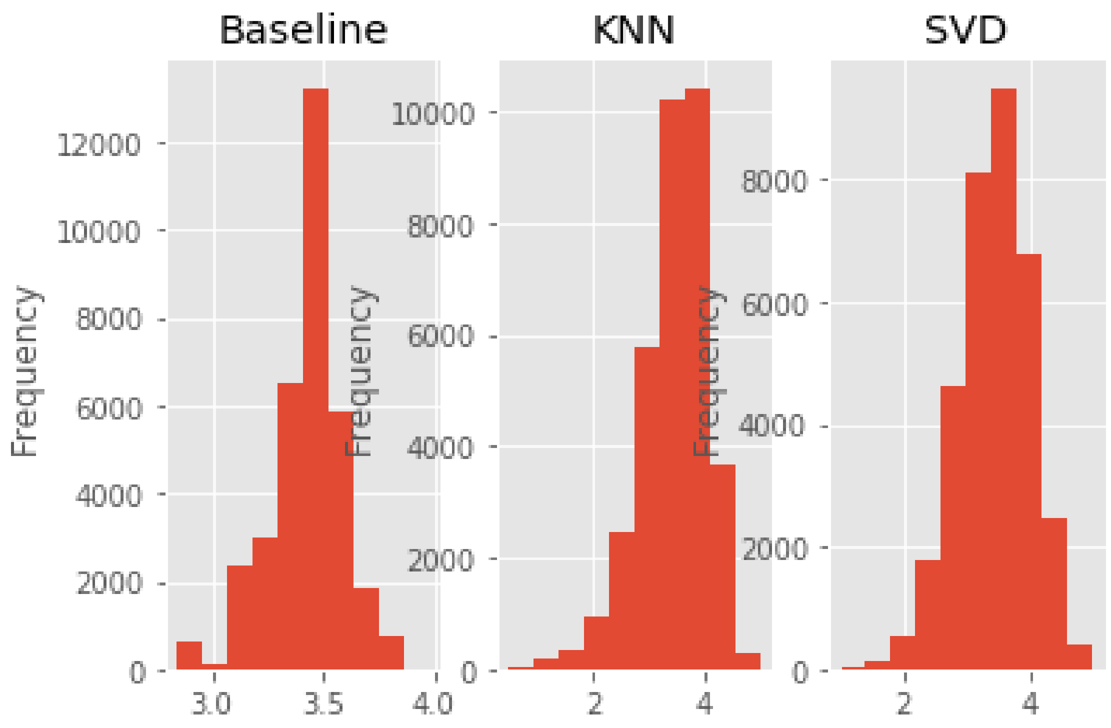
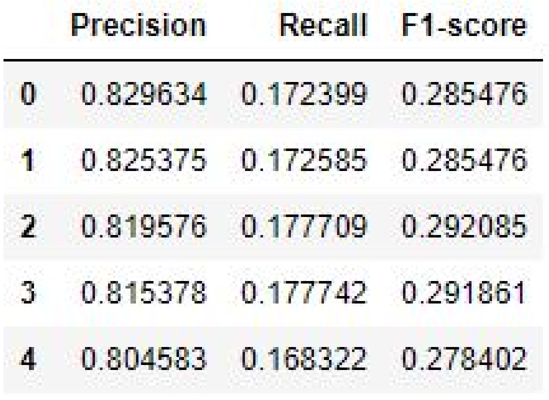


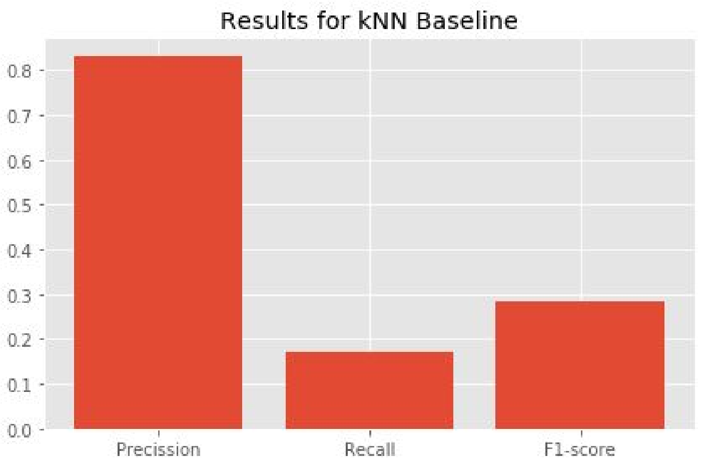
Figure 4.7 Comparisons of ratings for different algorithms

The last metrics that we measured were the decision support metrics. We decided to experiment in two main algorithms the kNN and SVD where we used the top 5 recommendations with a rating threshold of 4.5.



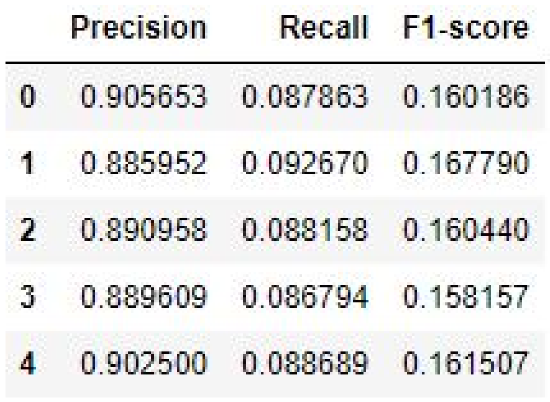
	Precision	Recall	F1-score
0	0.829634	0.172399	0.285476
1	0.825375	0.172585	0.285476
2	0.819576	0.177709	0.292085
3	0.815378	0.177742	0.291861
4	0.804583	0.168322	0.278402

(a) kNN Baseline measures for k-folds = 5



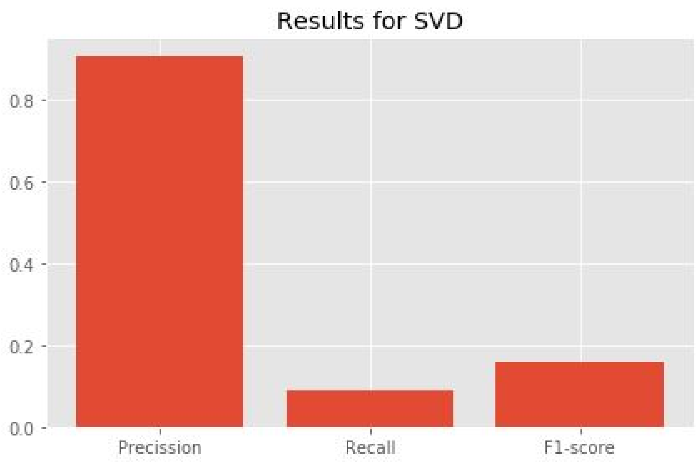
(b) Plot of kNN measures

Figure 4.8 kNN Baseline measures for top 5 recommendations



	Precision	Recall	F1-score
0	0.905653	0.087863	0.160186
1	0.885952	0.092670	0.167790
2	0.890958	0.088158	0.160440
3	0.889609	0.086794	0.158157
4	0.902500	0.088689	0.161507

(a) SVD measures for k-folds = 5

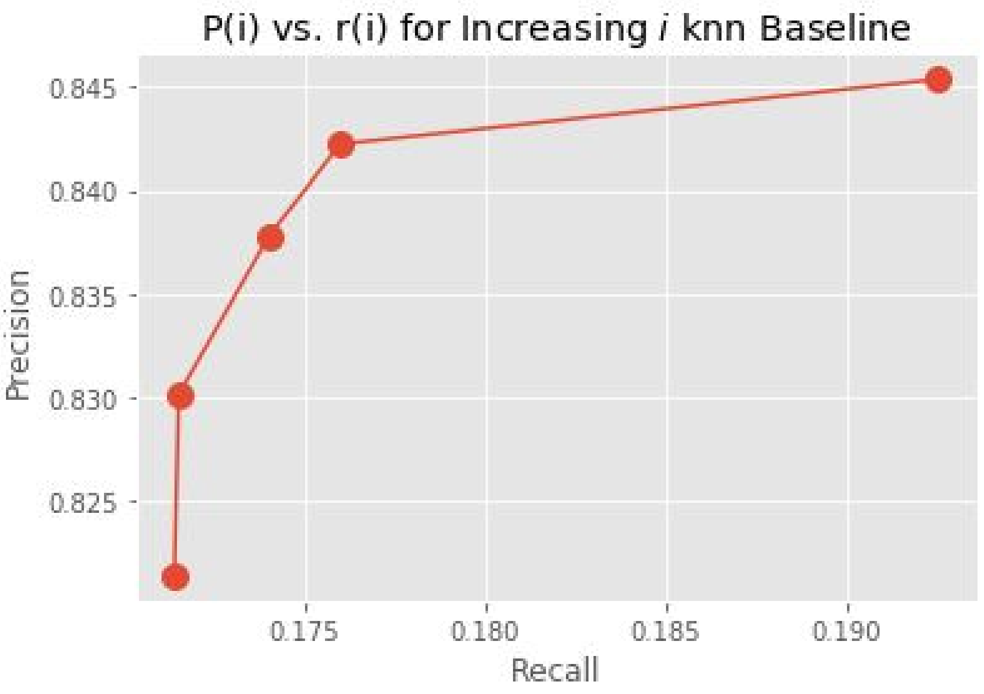


(b) Plot of SVD measures

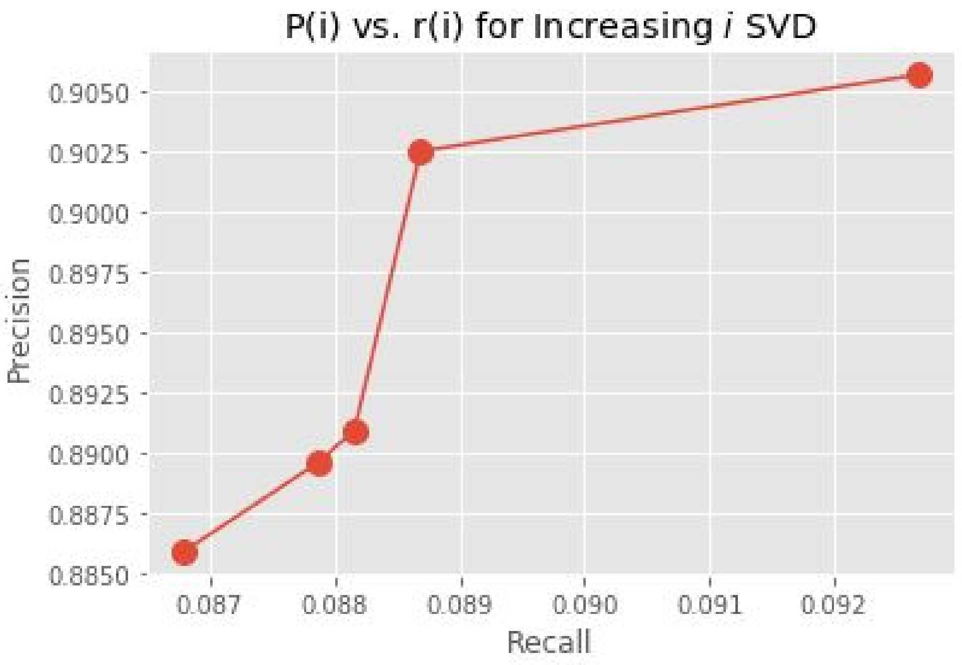
Figure 4.9 SVD measures for top 5 recommendations

As we can see from the figures above, the SVD method has better results regarding the Precision, but we noticed that kNN has significantly better performance regarding the Recall and F1-score.

Below there is the graphical representation of Precision and Recall, and we think them as a function of the index i, where i symbolize the next prediction of the item. The resulting plot depends heavily on the particular sequence of correct/incorrect recommendations. If the recommendation is correct, then both Precision and recall will increase. It can be seen that the kNN baseline reaches a better and smoother curve of the line, whereas recall values are greater than in the SVD method.



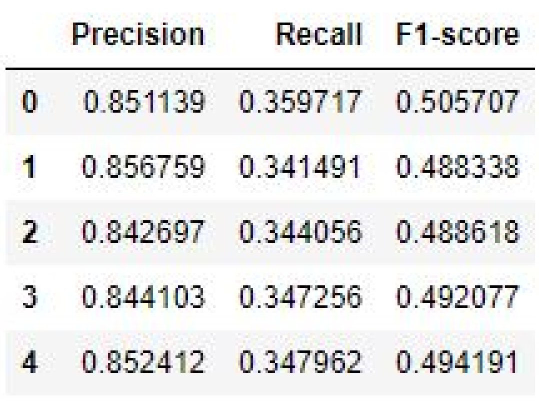
(a) Precision/Recall kNN



(b) Precision/Recall SVD

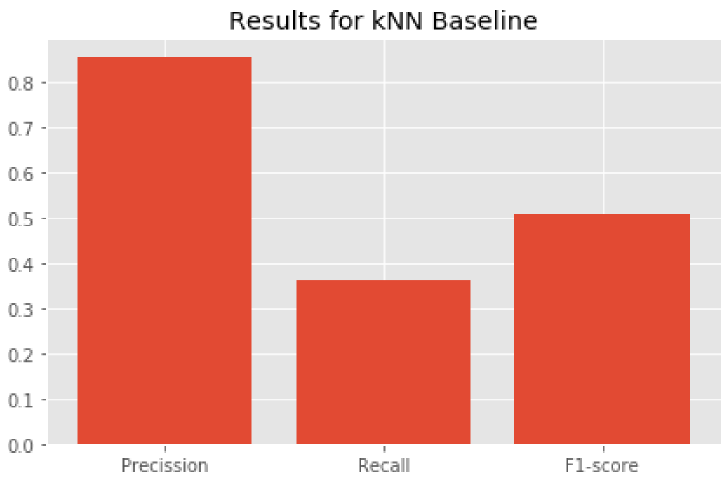
Figure 4.10 Precision and Recall for 2 algorithms

We decided to run another experiment and change the number of recommendations to the top 10 and give a threshold of 3.5 in the ratings. As the figures showed, the Precision of the system remained the same around 85%, but we noticed an increase in the recall value to around 35% that was the same for the kNN and SVD method. Regarding the F1-score, both methods gave the same score of around 50%.



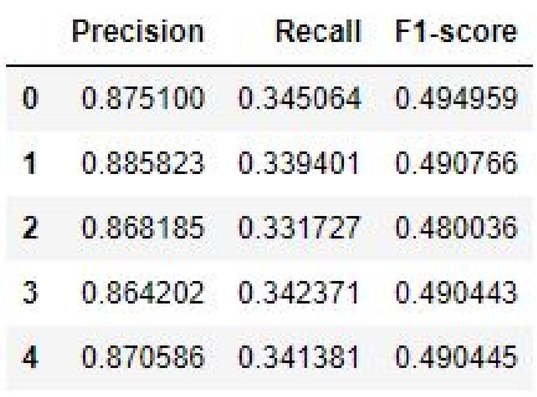
	Precision	Recall	F1-score
0	0.851139	0.359717	0.505707
1	0.856759	0.341491	0.488338
2	0.842697	0.344056	0.488618
3	0.844103	0.347256	0.492077
4	0.852412	0.347962	0.494191

(a) kNN measures for k-folds = 5



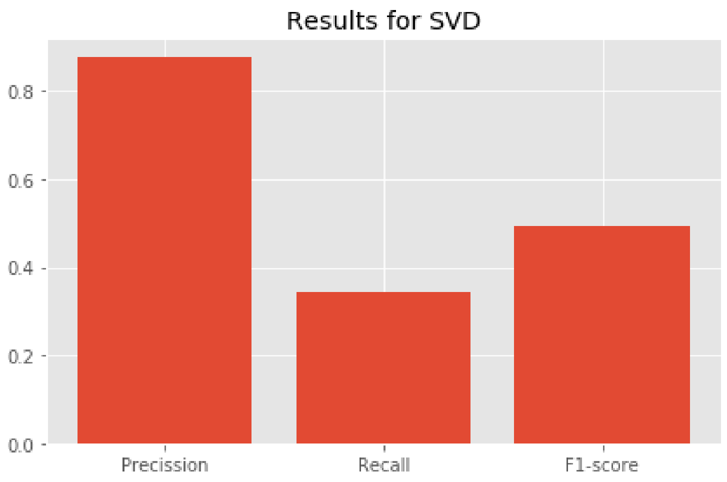
(b) Plot of kNN measures

Figure 4.11 kNN measures for top 10 recommendations



	Precision	Recall	F1-score
0	0.875100	0.345064	0.494959
1	0.885823	0.339401	0.490766
2	0.868185	0.331727	0.480036
3	0.864202	0.342371	0.490443
4	0.870586	0.341381	0.490445

(a) SVD measures for k-folds = 5



(b) Plot of SVD measures

Figure 4.12 SVD measures for top 10 recommendations

To conclude, SVD and kNN methods provide fast and accurate ways of estimating missing values in the user event interaction matrix. Both methods would surpass the methodology of recommending the most popular items or by estimating the row average of the ratings of that User. Seeing the result and comparing the graphs, we concluded that the SVD method performs well with large datasets. When given small datasets, the performance decrease, so we wanted to go with kNN, which works well with large and small datasets. We saw that performance of the kNN methods declines when a lower number of neighbors is used for estimation, primarily due to overemphasis of a few dominant expression patterns, so as many more neighbors were used, a more generalized model will be created, thus better performance of the system.

Chapter Five

Conclusion

Contents

5.1 Conclusion 49
5.2 Lesson Learnt 50
5.3 Future Work 51

5.1 Conclusion

This research aimed to identify effective algorithms and different techniques that would be used to contribute to the Icarus recommendations system. We showed that the problems of Collaborative filtering and Content-based recommenders could be solved by combining them with a weighted hybrid model, as we described in chapter 3.
Moreover, we tried to create a system that would change dynamically, and our Recommender depending on the coefficients that we introduced, making it possible to combine user interests with collaborative selections. In comparison with different recommenders that used algorithms like SVD (Single value decomposition), our recommendation has been shown to have similar results in performance, with the advantage that it was better in the cost of training time. Our method made it possible for the system to be scalable and handle multiple users and data. We plan to continue our experiments and implementations of the dynamic processes involved, in particular, to further doing more research in areas of the collaborative and content-

based components by improving their relative performance by decreasing the error in order to have better results in the future.

5.2 Lesson Learnt

This research allowed me to pursue my interests, learn something new, and to point my problem-solving skills in the sector of recommendation systems. I gained hands-on experience completing research in a creative project like ICARUS is. Assisting in the project gave me opportunities to discover new knowledge about different algorithms that are used nowadays for the recommendation, their advantages, and disadvantage, and I learned valuable skills and tools that are essential for a data scientist. I earned a lot of experience working in a research lab closely with faculty mentors and accomplished academic credentials that will help create a well-rounded resume.

The research showed that by combining different components and different approaches to implement a hybrid model, we use each algorithm's advantages and use them to solve many problems. Creating a dynamically changed recommender would give better predictions and would change over time, making it easy for the developer and also for the users of the system to be satisfied. We also learned that evaluating system performance is very difficult because this is not a classification problem, and the system should be trained and tested in real life. This was also shown in the metrics that we provided where the recall measures were around 50% that aren't very important in the recommender section, but what is important was that we achieved around 85% in Precision, where relevant items that we provided were liked by the User. The real challenge that we encounter in recommenders today is the extraction of features and providing unlimited and continuous data that affect directly in the performance. If this system isn't updated frequently, then the performance and the incomes would fall off rapidly.

5.3 Future Work

Our research results and conclusions can be effectively used in future work. As we discussed in the contribution section, our hybrid Recommender would prevent some of the challenges, but it still needs some improvements. Different filters may be included in the first phases of the Recommender to prevent different attacks and remove some of the outliers. Moreover, great future work would be the integration of the content-based system with the TF-IDF technique to find the similarity between items when provided the description and by adding demographic-based filtering techniques as soon as they are provided from the datasets. The system could then be evaluated again as soon as the interactions of the user increase. We can add different components in the hybrid model, and one of this can be a deep layered network that will take inputs of users and items, and will give back some results that can be merged with our Hybrid to create a more powerful model that would handle all the kinds of users with unique tastes. Last but not least, more advanced research should be focused on the ontology to integrate it with the datasets that the Recommender will use.

References

[1] Chris Anderson. *The long tail: Why the future of business is selling less of more.* Hachette Books, 2006.

[2] Raul Sergio Barth. “Design and implementation of a flight recommendation engine.” In: (2014).

[3] John S Breese, David Heckerman, and Carl Kadie. “Empirical analysis of predictive algorithms for collaborative filtering”. In: *arXiv preprint arXiv:1301.7363* (2013).

[4] Robin Burke. “Hybrid recommender systems: Survey and experiments”. In: *User modeling and user-adapted interaction* 12.4 (2002), pp. 331–370.

[5] Arpan V Dev and Anuraj Mohan. “Recommendation system for big data applications based on set similarity of user preferences”. In: *2016 International Conference on Next Generation Intelligent Systems (ICNGIS)*. IEEE. 2016, pp. 1–6.

[6] M Benjamin Dias et al. “The value of personalised recommender systems to e-business: a case study”. In: *Proceedings of the 2008 ACM conference on Recommender systems*. 2008, pp. 291–294.

[7] Mustansar Ali Ghazanfar and Adam Prugel-Bennett. “A scalable, accurate hybrid recommender system”. In: *2010 Third International Conference on Knowledge Discovery and Data Mining*. IEEE. 2010, pp. 94–98.

[8] Mustansar Ghazanfar and Adam Prugel-Bennett. “Fulfilling the needs of gray-sheep users in recommender systems, a clustering solution”. In: (2011).

[9] Noah Iliinsky and Julie Steele. *Designing data visualizations: Representing informational Relationships.* " O’Reilly Media, Inc.", 2011.

[10] D Jannach, M Zanker, and A Felfernig. *et G. Friedrich (2010). Recommender Systems: An Introduction.*

[11] D Khaturia et al. “A Comparative study on Airline Recommendation System Using Sentimental Analysis on Customer Tweets”. In: *International Journal of Advanced Science and Technology* 111 (2018), pp. 107–114.

[12] Yehuda Koren, Robert Bell, and Chris Volinsky. “Matrix factorization techniques for recommender systems”. In: *Computer* 42.8 (2009), pp. 30–37.

[13] Lihong Li et al. "A contextual-bandit approach to personalized news article recommendation". In: *Proceedings of the 19th international conference on World wide web*. 2010, pp. 661–670.

[14] Greg Linden, Brent Smith, and Jeremy York. "Amazon. com recommendations: Item-to-item collaborative filtering". In: *IEEE Internet computing* 7.1 (2003), pp. 76–80.

[15] Marwa Hussien Mohamed, Mohamed Helmy Khafagy, and Mohamed Hasan Ibrahim. "Recommender Systems Challenges and Solutions Survey". In: *2019 International Conference on Innovative Trends in Computer Engineering (ITCE)*. IEEE. 2019, pp. 149–155.

[16] Tamara Munzner. *Visualization analysis and design*. CRC press, 2014.

[17] Hai Thanh Nguyen and Anders Kofod-Petersen. "Using multi-armed bandit to solve cold-start problems in recommender systems at telco". In: *Mining Intelligence and Knowledge Exploration*. Springer, 2014, pp. 21–30.

[18] Hai Thanh Nguyen, Jérémie Mary, and Philippe Preux. "Cold-start problems in recommendation systems via contextual-bandit algorithms". In: *arXiv preprint arXiv:1405.7544* (2014).

[19] Sasmita Panigrahi, Rakesh Kumar Lenka, and Ananya Stitipragyan. "A Hybrid Distributed Collaborative Filtering Recommender Engine Using Apache Spark." In: *ANT/SEIT*. 2016, pp. 1000–1006.

[20] Manos Papagelis, Dimitris Plexousakis, and Themistoklis Kutsuras. "Alleviating the sparsity problem of collaborative filtering using trust inferences". In: *International conference on trust management*. Springer. 2005, pp. 224–239.

[21] Michael J Pazzani and Daniel Billsus. "Content-based recommendation systems". In: *The adaptive web*. Springer, 2007, pp. 325–341.

[22] Lakshmi Tharun Ponnam et al. "Movie recommender system using item based collaborative filtering technique". In: *2016 International Conference on Emerging Trends in Engineering, Technology and Science (ICETETS)*. IEEE. 2016, pp. 1–5.

[23] Francesco Ricci, Lior Rokach, and Bracha Shapira. "Introduction to recommender systems handbook". In: *Recommender systems handbook*. Springer, 2011, pp. 1–35.

[24] Ruslan Salakhutdinov, Andriy Mnih, and Geoffrey Hinton. "Restricted Boltzmann machines for collaborative filtering". In: *Proceedings of the 24th international conference on Machine learning*. 2007, pp. 791–798.

[25] Badrul M Sarwar et al. "Recommender systems for large-scale e-commerce: Scalable neighborhood formation using clustering". In: *Proceedings of the fifth international conference on computer and information technology*. Vol. 1. 2002, pp. 291–324.

[26] Badrul Sarwar et al. *Application of dimensionality reduction in recommender system-a case study*. Tech. rep. Minnesota Univ Minneapolis Dept of Computer Science, 2000.

[27] Ben Schafer et al. "Collaborative Filtering Recommender Systems". In: Jan. 2007.

[28] Andrew I Schein et al. "Methods and metrics for cold-start recommendations". In: *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*. 2002, pp. 253–260.

[29] G. Sharma. "How to Build a Recommender System(RS) - Data Driven Investor. Retrived from: https://medium.com/datadriveninvestor/how-to-built-a-recommender-system-rs-616c988d64b2". In: (2018).

[30] Andreas Töscher, Michael Jahrer, and Robert Legenstein. "Improved neighborhood-based algorithms for large-scale recommender systems". In: *Proceedings of the 2nd KDD Workshop on Large-Scale Recommender Systems and the Netflix Prize Competition*. 2008, pp. 1–6.

[31] Mohit Tuteja. "Flight Recommendation System based on user feedback, weighting technique and context aware recommendation system". In: *International Journal of Engineering and Computer Science* 5.9 (2016).

[32] Wei Zhou et al. "Shilling attack detection for recommender systems based on credibility of group users and rating time series". In: *PloS one* 13.5 (2018).

APPENDICES

Appendix A

Acronynms List

CB	Content Based
CF	Collaborative filtering
API	Application programming interface
REST	Representational state transfer
ML	Machine learning
AI	Artificial intelligence
NAN	Not a Number
DLN	Deep layered network
SGD	Stochastic gradient descent
SVD	Singular Value Decomposition
RMF	Regular matrix factorization
RBM	Restricted Boltzman Machines
MSE	Mean square error
MAE	Mean absolute error
RMSE	Root mean square error
ALS	Alternative Least Square

kNN	K-nearest neighbor)
JSON	JavaScript Object Notation
TP	True-Positive
FP	False-Positive
TN	True-Negative
FN	False-Negative
P	Precision
r	Recall
CV	Cross-validation
NMF	Non-negative matrix factorization
Tf-idf	Term frequency-inverse document frequency