INSTITUTO SUPERIOR DE ENGENHARIA DO PORTO
MESTRADO EM ENGENHARIA INFORMÁTICA




# Algoritmo Híbrido de Recomendação

**LICÍNIO CAMPOS NEVES CASTANHEIRA DE CARVALHO**
Julho de 2018

POLITÉCNICO
DO PORTO


# A Hybrid Recommendation Algorithm

**Licínio Campos Neves Castanheira de Carvalho**

**Dissertation for obtaining a Master's Degree in Informatics Engineering, Area of Specialisation in Knowledge and Information Systems**

**Supervisor: Fátima Rodrigues**

Porto, July 2018

# Dedication

I dedicate this to my family, especially to my mother and father for always being there for me, helping and supporting me.

# Resumo

Nesta era tecnológica em que nos encontramos há cada vez mais informação disponível na internet, mas grande parte dessa informação não é relevante. Isto leva à necessidade de criar maneiras de filtrar informação, de forma a reduzir o tempo de recolha de informação útil.

Esta necessidade torna o uso de sistemas de recomendação muito apelativo, visto estes personalizarem as pesquisas de forma a ajudar os seus utilizadores a fazer escolhas mais informadas. Os sistemas de recomendação procuram recomendar os itens mais relevantes aos seus utilizadores, no entanto necessitam de informação sobre os utilizadores e os itens, de forma a melhor os poder organizar e categorizar.

Há vários tipos de sistemas de recomendação, cada um com as suas forças e fraquezas. De modo a superar as limitações destes sistemas surgiram os sistemas de recomendação híbridos, que procuram combinar características dos diferentes tipos de sistemas de recomendação de modo a reduzir, ou eliminar, as suas fraquezas.

Uma das limitações dos sistemas de recomendação acontece quando o próprio sistema não tem informação suficiente para fazer recomendações.

Esta limitação tem o nome de Cold Start e pode focar-se numa de duas áreas: quando a falta de informação vem do utilizador, conhecida como User Cold Start; e quando a falta de informação vem de um item, conhecida como Item Cold Start.

O foco desta dissertação é no User Cold Start, nomeadamente na criação de um sistema de recomendação híbrido capaz de lidar com esta situação.

A abordagem apresentada nesta dissertação procura combinar a segmentação de clientes com regras de associação. O objetivo passa por descobrir os utilizadores mais similares aos utilizadores numa situação de Cold Start e, através dos itens avaliados pelos utilizadores mais similares, recomendar os itens considerados mais relevantes, obtidos através de regras de associação.

O algoritmo híbrido apresentado nesta dissertação procura e classifica todos os tipos de utilizadores. Quando um utilizador numa situação de Cold Start está à procura de recomendações, o sistema encontra itens para recomendar através da aplicação de regras de associação a itens avaliados por utilizadores no mesmo grupo que o utilizador na situação de

Cold Start, cruzando essas regras com os itens avaliados por este último e apresentando as recomendações com base no resultado.

**Palavras-chave**: sistema de recomendações, data mining, sistema híbrido, cold start, novo utilizador

# Abstract

Recommender systems, or recommenders, are a way to filter the useful information from the data, in this age where there is a lot of available data.

A recommender system's purpose is to recommend relevant items to users, and to do that, it requires information on both, data from users and from items, to better organise and categorise both of them.

There are several types of recommenders, each best suited for a specific purpose, and with specific weaknesses. Then there are hybrid recommenders, made by combining one or more types of recommenders in a way that each type supresses, or at least limits, the weaknesses of the other types.

A very important weakness of recommender systems occurs when the system doesn't have enough information about something and so, it cannot make a recommendation. This problem known as a Cold Start problem is addressed in this thesis.

There are two types of Cold Start problems: those where the lack of information comes from a user (User Cold Start) and those where it comes from an item (Item Cold Start).

This thesis' main focus is on User Cold Start problems.

A novel approach is introduced in this thesis which combines clients' segmentation with association rules. The goal is first, finding the most similar users to cold start users and then, with the items rated by these similar users, recommend those that are most suitable, which are gotten through association rules.

The hybrid algorithm presented in this thesis finds and classifies all users' types. When a user in a Cold Start situation is looking for recommendations, the system finds the items to recommend to him by applying association rules to the items evaluated by users in the same user group as the Cold Start user, crossing them with the few items evaluated by the Cold Start user and finally making its recommendations based on that.

**Keywords**: recommender system, data mining, hybrid system, cold start, new user

# Index

# Figure Index

# Table Index

# Acronyms and Symbols

## List of Acronyms

| | |
|---|---|
| **CF** | Collaborative Filtering |
| **CB** | Content-Based Filtering |
| **IBCF** | Item-Based Collaborative Filtering |
| **UBCF** | User-Based Collaborative Filtering |
| **CS** | Cold Start |
| **SVM** | Support Vector Machine |
| **DT** | Decision Tree |
| **AR** | Association Rules |
| **kNN** | k-Nearest Neighbours |
| **ANN** | Artificial Neural Networks |
| **SOM** | Self-Organising Maps |
| **LA** | Link Analysis |
| **TP** | True Positive |
| **FP** | False Positive |
| **TN** | True Negative |
| **FN** | False Negative |
| **AIC** | Akaike Information Criteria |
| **BIC** | Bayesian Information Criteria |
| **MAE** | Mean Absolute Error |
| **NMAE** | Normalised Mean Absolute Error |
| **RMSE** | Root Mean Square Error |
| **NDPM** | Normalized Distance-based Performance Measure |

# 1 Introduction

Recommender engines fall under the sub category of information filtering systems that aim to forecast preferences or ratings given to the item by the user. Recommender engines are nowadays an integral portion of e-commerce sites, which help in recommending items or products of interest to people all around the world. The major assistances of having a recommender system are customer retention, information retrieval, personalization and many more. Also these systems can be used for recommendations of products such as music, books, restaurants, TV shows and movies. Presently these are used successfully in commercial websites such as Movielens, Amazon, eBay, Linkedln, Jinni, Facebook and many others.

## 1.1 Context

The information made available by the internet is constantly increasing, making access to it easier but the process of gathering it from relevant sources harder.

This makes recommender systems a very appealing concept because they allow the personalization of the information through web applications, therefore helping its users to make more informed, and generally better, decisions.

There are several types of recommender systems, each with their own strengths and weaknesses. Because of that, nowadays the most often used implementation of recommender systems is the hybrid approach, which combines different types of recommender systems, using the strengths of some to reduce, or even eliminate, the weaknesses of others. This

approach allows the creation of a more robust type of recommendations, which is why it is the most often used approach for newer recommender systems.

## 1.2 Problem

Even though hybrid recommender systems are more robust, they still have to deal with the weaknesses inherent to the various recommender systems that are part of it. One of those types of problems is the Cold Start problem.

A Cold Start problem occurs due to the lack of information on a new item or user that enters the system, making the recommender system incapable of correctly recommending relevant items to its users.

## 1.3 Objective

The main objective of the project documented in this report is the creation of a hybrid recommender algorithm, capable of dealing with the Cold Start User problem, that is when a brand-new user enters the system and the system doesn't have much information available about the new user for recommending items to him.

## 1.4 Expected Results

The aim of this dissertation is to create a hybrid solution that can give more accurate recommendations in situations of Cold Start Users.

The proposed solution is expected to perform better than the other approaches presented, with better results for accuracy and lower error rates when recommending items to new users, where the lack of data makes a conventional system unable to do so.

## 1.5 Value Analysis

The creation of this solution aims to help movie recommendation services and streaming services to better recommend brand-new movies of interest to their users and recommend relevant movies to new users.

Its main purpose is improving the service quality for movie recommendation services or streaming services.

## 1.6 Proposed Methodology

This solution is an implementation of a Recommender System using Data Mining techniques. Data mining will be used to extract knowledge hidden in data. To ensure quality of the work, the standard CRISP-DM methodology **(Wirth and Hipp, 2000)** will be used.

# 2 State of the Art

Recommender systems, also known as recommendation systems, have an increasingly larger relevance ever since the publication of the first paper on collaborative filtering in the mid-1990s and, with its relevance, so did its importance as a research area grew.

Since its early use, the internet has been constantly expanding, overwhelming the user with information from an increasing amount of sources.

In the current age of information, with ever-increasing amounts of user-generated content, a problem arises linked to the need to filter all this content.

This problem occurs because, although users having access to more information can be beneficial, the process of choosing which information is relevant becomes increasingly more complex as the alternative sources grow, which creates a demand for approaches that allow users to distinguish relevant from irrelevant information in the most efficient possible way.

This demand has prompted the interest in personalized recommender system's because the use of recommender systems is not only limited to making suggestions or helping in the decision process but also to filtering content.

This chapter presents the various types of recommender systems, ways for the recommenders to calculate the similarity between the items or users, as well as some common data mining techniques used in recommender systems. It also shows approaches presented by some authors in several scientific papers and a brief analysis of them.

## 2.1 Recommender Systems

Recommender systems are software tools and techniques that give suggestions and their main purpose is giving them to users, so that they can make better decisions.

A more in-depth way to characterize them would be as systems that use past opinions of members of a community to assist users in the same community in finding content that may be of their interest, usually from a very large set of alternatives.

The list of its applications is constantly growing and includes suggesting new videos to watch (e.g. when you are watching videos on YouTube), showing what products you might want to get (e.g. when you are buying items on Amazon), recommending new friends on Facebook or presenting other news you might want to read/watch on news websites.

A good recommender system focuses on making recommendations more personalized, using several means, like a user's digital footprint or the information about each product (i.e. the products specifications, its feedback from users, its comparison with other products, etc.).

Recommender systems are generally classified as one of three different types: Collaborative Filtering (CF), Content-based or Knowledge-based. However there are also Hybrid recommender systems, which are combinations of the aforementioned types of recommender systems.

### 2.1.1 Collaborative Filtering Recommenders

A CF recommender system focuses on the similarity between users. It puts its emphasis on the assumption that two users have similar interests if they acquired the same item in the past, as well as the assumption that, given that, they will have similar tastes in the future.

In this type of recommendation, filtering items from a large set of alternatives is done collaboratively between the preferences of the different users.

This recommendation approach only considers user preferences and does not take into account things like demographics, user attributes or the features of the items being recommended **(Lika et al., 2014)**. It can also get more accurate results if it uses a large set of user preferences.

However, this approach has been known to reveal two major types of problems: sparsity and scalability **(Park et al., 2012)**.

Data sparsity is a problem that occurs when only a small part of the available items is rated by the users, which leads to the amount of ratings in the rating matrix being insufficient for the system to make accurate predictions **(Shambour & Lu, 2015) (Wu et al., 2015)**.

Data scalability comes from situations when a system has millions of users or items, making the use of traditional Collaborative Filtering algorithms impractical. According to Twitter, the solution for these situations requires the scaling of recommendations through the use of clusters of machines **(Gupta et al., 2013)**.

The CF recommender system filters the information with a technique based either on the user's previous evaluations of items or the history of their previous purchases. This type of recommender system can be further divided into Item-Based Collaborative Filtering (IBCF) and User-Based Collaborative Filtering (UBCF).

These types of Collaborative Filtering recommender systems also have potential problems when dealing with new users or items. These types of problems are derived from data sparsity and are commonly known as Cold Start (CS) problems. According to **(Gorakala & Usuelli, 2015)**, they can be identified as one of three possible types:

1. Recommendations for new Users (i.e. User CS problems);
2. Recommendations of new Items (i.e. Item CS problems);
3. Recommendations of new Items for new Users (i.e. a combination of the two previous problems);

For example, when a new user that hasn't purchased any item on an online shop enters the system, neither an IBCF nor a UBCF recommender will be able to recommend any item. This happens with IBCF recommenders because it needs to know the items purchased by this new user, and happens with UBCF recommenders because it needs to know which users have similar preferences to this user but it doesn't have its item ratings.

A similar example with items would be that an item that hasn't been purchased by anyone would never be recommended, because an IBCF recommender compares items that have been purchased by the same users, thus it wouldn't be able to match the new item with any other. The new item would never be recommended with a UBCF recommender because it only recommends items purchased by similar users and the item wasn't purchased yet.

#### 2.1.1.1 Item-based Collaborative Filtering

The starting point in implementing an IBCF recommender system is the creation of a rating matrix in which each row corresponds to a user and each column to an item.

Its algorithm starts by measuring the similarity of every two items by comparing the ratings received by users and trying to find similarities between them.

It then finds the *k* items with the most similarities for each item, with *k* being a positive integer value.

After that, it identifies, for each user, the items with the greatest similarities to those in said user's past purchases.

#### 2.1.1.2 User-based Collaborative Filtering

A UBCF recommender system works opposite of IBCF recommender systems.

Given a new user, the algorithm identifies similar users, measuring the similarity of the new user with every other user and finds the most similar users.

When choosing the users with the greatest similarity to the new user, the algorithm may take one of two approaches: it can find the *k* most similar users (e.g. with the k-Nearest Neighbours' method), with *k* being a positive integer value; or it can choose all the users with a similarity rating above a specified threshold.

After that, the algorithm will rate the items purchased by the previously defined users, either with an average of all the users' ratings or with a weighted average rating that uses the similarities as weights.

Finally, the items chosen to be recommended by the system to the new user become the top-rated items of the previous process.

### 2.1.2 Content-based Recommenders

A Content-Based (CB) recommender system focuses on the similarity between the items and user profiles when making a recommendation. It analyses a set of items rated by a user, as well as ratings given by that same user, to create a profile to be used in future recommendations of items for users that fit that same profile.

This type of system recommends items that are similar to those that users with the same profile have shown interest for in the past. This means that it needs a measure with

which to calculate the similarity between different items. This calculation is done by taking into account the features associated with the compared items and is matched with the user's historical preferences.

This recommendation approach does not take into account the user's neighbourhood preferences, which means it doesn't require the item preferences of a large user group to increase its recommendation accuracy as it only considers the user's past preferences and item features.

This type of recommender system's basic principles are as follows: analysis of the description of a particular user's preferred items to determine the main common attributes that can be used to distinguish these same items, storing that information in a user profile; and comparing each item's attributes with the user profile, with the purpose of only recommending items that have a high similarity with said user's profile **(Lu et al., 2015)**.

However, this type of system also has limitations. One of them lies in the fact that it causes overspecialized recommendations, which means that it only recommends items very similar to those that the user already knows that exist **(Park et al., 2012)**.

One way used to address this limitation has been the introduction of controlled randomness to the process **(Adomavicius & Tuzhilin, 2005)**.

Another problem of this type of recommender system lies with its limited content analysis. This is due to the fact that if two distinct items have the same features, they are considered exactly the same to the system. For example, a good article and a poor article would be indistinguishable to the system if they both used the same terms **(Adomavicius & Tuzhilin, 2005)**.

### 2.1.3 Knowledge-based Recommenders

Knowledge-based recommenders, or constraint-based recommendation systems, are used in specific situations, where the users' purchase history is smaller.

This type of recommender takes into account item features, user preferences and recommendation criteria when trying to make recommendations. The user preferences are defined by explicitly asking the user in question. And the model accuracy is determined based on how useful the recommended item is to the user.

### 2.1.4 Advantages and Limitations of Recommender Systems

Table 1 shows the pros and cons of each type of recommender described previously.

| Approach | Advantages | Limitations |
|---|---|---|
| Collaborative Filtering Approach | - No domain knowledge required<br>- Quality of recommendations increases over time<br>- It can identify cross genre niches<br>- Implicit feedback is sufficient | - New-user problem (cold start problem)<br>- Gray sheep problem<br>- Scalability problem<br>- New-item problem (First-rate problem) |
| Content-based Filtering Approach | - No domain knowledge required<br>- New item recommendation<br>- Quality of recommendations increases over time<br>- Implicit feedback is sufficient | - New-user problem (cold start problem)<br>- Limited content analysis problem<br>- Over-specialisation<br>- Scalability problem |
| Knowledge-based Filtering Approach | - No cold start problem<br>- No over-specialisation problem<br>- Prone to preference changes<br>- No scalability problem | - Need domain knowledge<br>- Does not learn over time |

Table 1 – Table showing the Trade-Off between Recommendation Approaches **(Gupta & Goel, 2016)**

### 2.1.5 Hybrid Systems

Hybrid Systems are combinations of various features of distinct recommender systems with the purpose of building a more robust system. The combinations of various recommender systems can be used to remove the disadvantages of one system while using the advantages of another system, thus making the system more robust.

The most common combination of recommenders is that of a Collaborative Filtering recommender system with other recommender systems, in an attempt to avoid problems as Cold-Start, sparseness or scalability problems **(Lu et al., 2015)**.

An example would be the combination of collaborative-filtering systems, where the model performs badly when new items don't have ratings (i.e. an Item Cold Start problem), with content-based systems, where information about an item's features is available, which

would allow for a more accurate and efficient recommendation of new items **(Gorakala & Usuelli, 2015)**.

There have also been successful attempts at tackling the problems of CF systems by adding external information to the rating information. Examples of this external information include demographic information, content-based information, explicit trust information, semantic information and user's knowledge. However this type of approach, besides the difficulty inherent to the acquisition of external information has limitations such as making the recommender system less flexible or the difficulty inherent to the acquisition of external information **(Shambour & Lu, 2015)**.

There are several different approaches when building a Hybrid recommender system:

- ***Parallelized Hybrid Systems:***

  As its name suggests, this type of system runs the recommenders separately, then combining their results.

  The phase where the results are combined can have many different approaches when obtaining the final result. One could be the computation of the average of all results provided by the recommenders, while another could be choosing only one of the results, following a predetermined rule, but other different ways exist.

- ***Pipelined Hybrid Systems:***

  This type of system runs the recommenders in sequence, with the output of one being the input of the next.

- ***Monolithic Hybrid Systems:***

  This type of system integrates the different approaches in the same algorithm. Some examples of this are feature combination and feature augmentation.

  Feature combination creates an algorithm that learns from different types of inputs (e.g. an algorithm that can take account of ratings, user profiles and item descriptions).

  Feature augmentation builds the input of a recommender by combining different data sources.

## 2.2 Calculating Similarity

When choosing the possible items to recommend, all recommenders need a way to compare the similarity between either two items or users. For that, they may use different measures, such as the Euclidean distance, the Cosine distance or the Pearson Correlation Coefficient.

The Euclidean distance is the simplest technique for calculating the similarity between two items. It's the distance between two points, or objects, and it requires the quantification and/or representation of their attributes on a bi-dimensional plane. For that, it uses the following equation **(Gorakala & Usuelli, 2015)**:

$$Euclidean\ Distance(x, y) = \sqrt{\sum_{i=1}^{n} |x_i - y_i|^2} \qquad (1)$$

The Cosine distance, also known as the Cosine similarity, is a technique that measures the cosine of the angle between two vectors, each representing an item, using the following equation **(Gorakala & Usuelli, 2015)**:

$$similarity = \cos(\theta) = \frac{A-B}{\|A\|\|B\|} \qquad (2)$$

The Pearson Correlation Coefficient (PCC) tries to get the similarity of two items from the correlation between their variables. It is calculated as the co-variance of the two variables divided by the product of their standard deviations, as presented in the following equation **(Gorakala & Usuelli, 2015)**:

$$\rho_{X,Y} = \frac{cov(X,Y)}{\sigma_X \sigma_Y} \qquad (3)$$

**(Gorakala & Usuelli, 2015)** note that there have been empirical studies that have showed that the PCC outperformed other similarity measures for UBCF recommender systems and that the Cosine distance technique consistently performs well in IBCF recommender systems.

The Gower's Similarity Coefficient **(Gower, 1971)**, or Gower distance, is one of the most popular measures of proximity for mixed data types. It compares two cases, $i$ and $j$, and is defined by the following equation:

$$S_{ij} = \frac{\sum_k^n w_{ijk} S_{ijk}}{\sum_k^n w_{ijk}} \qquad (4)$$

## 2.3 Data Mining Techniques

Some of the data mining algorithms more commonly used by recommender systems include Clustering algorithms, Support Vector Machines, Decision Trees, Association Rules, k-Nearest Neighbours, Artificial Neural Networks, Regression, Link Analysis and Ensemble methods, such as Bagging, Boosting and Random Forests.

### 2.3.1 Clustering

Clustering, or cluster analysis, is the process that involves the grouping of objects in a way that objects grouped in one cluster are the most similar between themselves and objects in different clusters are as different as possible from each other.

Clustering is an unsupervised learning method, which means that it does not have response variables to predict, and that it instead tries to find patterns within the dataset made available.

It identifies a set of clusters, each representing a different category, used to describe the data. Among several clustering algorithms, the most popular in the area of Recommender Systems are K-means and Self-Organizing Maps (SOM) algorithms **(Park et al., 2012)**.

#### 2.3.1.1 K-Means

K-means Clustering **(Hartigan, 1975)** is an iterative clustering algorithm that takes as input *k*, in which *k* is the number of clusters to be formed from the data. It then makes partitions of a set of *n* items into the *k* clusters.

To achieve clustering, the K-means algorithm requires two steps: first, it randomly chooses centre points for each of the *k* clusters, assigning each data point to whichever cluster centre it is closer to; second, it moves the centroid (i.e. the centre of the cluster in the previous iteration), choosing its new position by calculating the mean position of all data points in its cluster.

It then repeats these two steps until all data points are grouped and the mean of the data points of each cluster (i.e. the centroid) does not change.

#### 2.3.1.2 K-Medoids

The K-Medoids algorithm is a clustering algorithm related to the K-Means algorithm and the *medoidshift* algorithm **(Jain, 2010)**. Like K-Means, this algorithm is partitional, which means it breaks the input dataset up into groups, or clusters.

This algorithm minimizes the sum of the dissimilarities between the points labelled to be in a cluster and a point designated as the centre of that cluster, while also choosing data points as centres, also known as *medoids* or exemplars.

The K-Medoids algorithm is easy to understand, more robust to noise and outliers when compared to K-Means and has the added benefit of having an observation serve as the exemplar for each cluster, however both run time and memory are quadratic (i.e. $O(n^2)$).

### 2.3.2 Support Vector Machine

Support Vector Machine (SVM) algorithms use supervised learning to solve classification problems, earning the title of one of the best algorithms to deal with classification-type problems. They also perform well with non-linear datasets. The foundations for SVM have been developed by Vapnik **(Vapnik, 1998)** and are very popular due to many attractive features, and promising empirical performance.

SVMs require a training set where each data point can belong to one of two categories. It then builds a model that assigns new data points into one of those two categories.

The aforementioned model is a representation of the data points of the training set as points in space. Those points are mapped in a way that the examples of the different categories are divided by the widest possible margin. New data points are then mapped in said space and the prediction of which category they belong to is based on the side of the margin they are mapped in.

The application of SVM to a dataset with *p* dimensions implies the mapping of the data to a *p-1* dimensional hyperplane followed by the definition of a clear boundary between the categories with the largest margin possible, as shown in Figure 1.

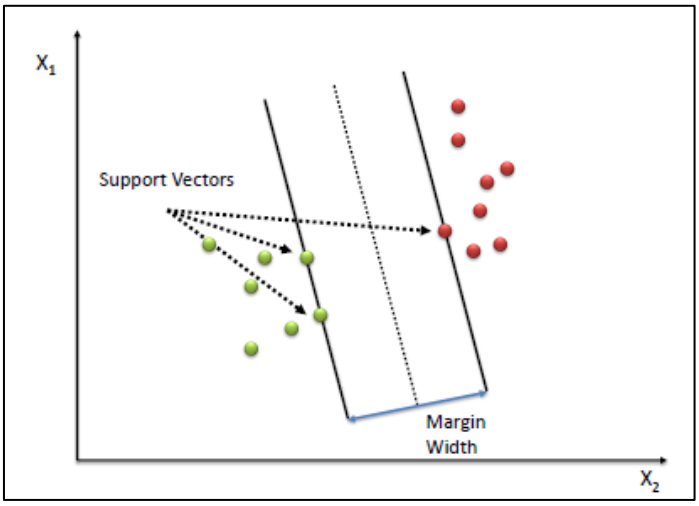


Figure 1 – Support Vectors Separating Two Different Categories **(1)**

The SVM classifier only depends on the data points that lie on the margins of the hyperplane. Those points are called support vectors.

The decision border is influenced only by the support vectors. It is not influenced by points located away from the borders. This means that alterations on data points that are not support vectors do not influence the decision border. However, alterations on the support vectors imply changes to the decision border.

SVMs were originally designed for binary classification. How to effectively extend it for multiclass classification is still an ongoing research issue. Several methods have been proposed where typically a multiclass classifier is constructed by combining several binary classifiers, although it requires significantly more computer power.

### 2.3.3 Decision Trees

Decision Trees (DT) are tree-based supervised learning algorithms used to solve classification problems.

They are simple and fast, although not very accurate when compared with logistic regression methods.

These types of algorithms build decision trees to classify cases into a set of known classes. In those trees, the top node is called the root node, with each internal non-terminal (i.e. non leaf) node representing a test of an attribute, and each terminal (i.e. leaf) node representing a class prediction **(Quinlan, 1986)**.

The tests of the attributes are used to split the data in a way that best separates each class from each other.

Then, the same process is repeated until the data is narrowed down to the point where all the bottom nodes represent the different response variables or when the data cannot be logically split any further.

After the creation of the tree model, predictions for new data points are taken through each tree node, answering said node's question and following the respective path until an end node is reached and a logical class can be chosen as the prediction response.

### 2.3.4 Association Rules

This data mining technique finds all association rules that have values above a user-specified minimum support and minimum confidence threshold levels.

Given a set of transactions, with each transaction containing a set of items, an association rule has the form "X=>Y" (i.e. X implies Y), where X and Y are two sets of items **(Agrawal et al., 1993)**.

The support and confidence thresholds are used to find the most interesting sets of rules. While the support measure refers to the frequency with which the set of items appears in the dataset, the confidence is a measure indicative of how many times the rule was found to be true (i.e. when a transaction has the set of items X, the set of items Y also occurring).

Another very important measure is the lift, which can be used to find out if the sets of items in a rule are dependent of one another. If the lift value equals one (1), then the two sets of items are independent of each other, but if it is lower than one, then they are dependent of each other and the respective rule can be useful.

### 2.3.5 K-Nearest Neighbours

The k-Nearest Neighbours (kNN) algorithm **(Dudani, 1976)** is a traditional algorithm in Collaborative Filtering recommender systems that uses a three-phase process to make its recommendations.

The first phase involves the construction of a user profile with the user's preference ratings. Following that, it applies a statistical or machine learning technique to find the *k* users

(i.e. neighbours) that have shown similar behaviours in the past to those of the target user. This allows the creation of a neighbourhood based on the level of similarity between the target user and other users. After the creation of the neighbourhood for a target user, the system makes a list of the top-*n* items that the target user is most likely to purchase by analysing the items that his neighbours have shown interest for **(Park et al., 2012)**.

### 2.3.6 Artificial Neural Network

Artificial Neural Networks are a parallel distributed information processing system that is able to learn and self-organize **(Mitchell, 1997)**.

An Artificial Neural Network (ANN), also known as a Neural Network, is a system loosely based on the way a human brain works, using large collection of interconnected neural units, known as perceptrons, as basic processing machines. It then uses those perceptrons to form a network that is able to perform very complex tasks.

The network is divided into layers with each perceptron of a layer being connected to the perceptrons of the previous and/or next layer, when possible. This network usually has three types of layers: an input layer, a hidden layer and an output layer.

Each perceptron receives an input, normally from perceptrons in the previous layer, applies a function to said input and propagates it to the next layer.

There are two types of neural network: the feed-forward networks, in which the values (i.e. signals) travel in one way, as described before; and the feedback networks, in which the values (i.e. signals) travel in both directions, allowing the creation of loops in the network. This last type of neural network can be very powerful but also more difficult to train, due to their dynamic nature, as their state is constantly changing.

### 2.3.7 Regression

Regression analysis is an effective tool for analysing associative relationships between dependent and one or more independent variables.

Its uses include prediction, curve fitting and testing systematic hypothesis about the relationships between variables **(James et al., 2013)**.

### 2.3.8 Link Analysis

Link Analysis (LA) is used to discover relations between domains in databases with large dimensions. Some of its algorithms include PageRank and HITS **(Ding et al., 2003)**.

It has several types, one of which being Social Network Analysis, that is a sociological approach that tries to analyse pattern relationships and interactions in order to find a fundamental social structure.

This technique has shown a lot of potential for improving the accuracy of web searches, with most of its algorithms handling each web page as a single node in a web graph.

### 2.3.9 Ensemble Methods

Ensemble methods use multiple learning algorithms, which allow better predictive results than the application of any single learning algorithm.

#### 2.3.9.1 Bagging

Bagging, or Bootstrap Aggregating **(Breiman, 1996)**, has the objective of improving the stability and accuracy of machine learning algorithms, by helping to avoid overfitting and reducing variance. It is mostly used with Decision Trees.

The Bagging ensemble randomly generates Bootstrap samples from the dataset, training the models individually. It then makes its predictions by aggregating or averaging the response variables of all samples.

#### 2.3.9.2 Random Forests

Even though they are built on a similar approach, Random Forests **(Breiman, 2001)** are more supervised learning algorithms than bagging methods.

Unlike bagging, that selects all the variables in all *n* samples generated, this algorithm selects only a few predictor variables, randomly, from the total variables for each of the *n* samples created. It then trains those samples with the models and the predictions are made with an average of the result of each model.

The dependency of strong predictors does not happen with this method because of the selection of only some of the variables rather than all of them in every iteration. It also de-correlates variables, leading to a reduced variability of the model and, thus, a more reliable one.

#### 2.3.9.3 Boosting

Boosting **(Freund et al., 1999)** fits a new model for each copy of the dataset, combining all the individual models to create a single predictive model. Each new model is built using information from models previously built.

Boosting is an iterative method that involves two steps: building of a new model on the residuals of previous models; calculating the residuals with the new model and updating the residuals used in the previous step. The aforementioned steps are repeated for multiple iterations, which allow the model to learn from its mistakes, thus improving its accuracy **(Gorakala & Usuelli, 2015)**.

## 2.4 Evaluation of Recommender Systems

This section introduces the evaluation techniques that might be used to evaluate recommender systems.

It is important to note that when evaluating a model it's critical to its performance to check if it is overfitting or underfitting and how well the model fits the test data and all future data.

Underfitting happens when the model generalises its predictions too much, performing poorly on the training set. It often happens when the model did not have enough data in the training set to learn from.

Overfitting of a model occurs when the model adapts too much to the training set, performing well in it, but performing poorly on the test set. This problem appears when the model memorizes data patterns in the training set, instead of learning from it.

To prevent any of the previously mentioned scenarios, models are evaluated with Cross Validation, Regularization, Model Comparison, ROC Curves, Confusion Matrix, among others.

### 2.4.1 Cross Validation

Cross Validation is a commonly-used model evaluation technique in which the data is divided into several separate datasets, that all minus one is used for training, and the one left, not used for training, is used for testing.

The model is then built and trained using the several training sets, and its performance evaluated using the test set. This process is repeated several times until all the partitions have been used once to test, and the test errors are calculated for every one of them. In the end, an average test error is calculated, to give an average of the model's accuracy.

### 2.4.2 Leave-One-Out

This evaluation technique is similar to Cross Validation but in this case, for each iteration, the test set only has one record.

For a sample with *n* records, this technique creates the model with *n-1* records and tests said model with the remaining record.

However, this requires a lot of processing power, making it most viable with small samples. It also has a high variance, due to it only being tested on one record.

### 2.4.3 Regularisation

With Regularisation a penalty is applied to the data variables not only to reduce the complexity of the model, but also to try and minimize the cost function.

The two most popular regularization techniques are Ridge Regression and Lasso Regression.

The process of Regularisation attempts to find which variables are expendable, based on the assumption that a smaller number of variables will improve the model's performance.

### 2.4.4 Confusion Matrix

The Confusion Matrix is used when evaluating a classification model and it involves calculating metrics such as the model's Precision, Recall/Sensitivity and Specificity.

The Confusion Matrix is a table that crosses the model's predictions with their actual values, finding the times in which the model made correct/incorrect classifications.

It divides the records into four categories: True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN), as shown in Table 2.

| | Actual POSITIVE | Actual NEGATIVE |
|---|---|---|
| **Predicted POSITIVE** | TRUE POSITIVE | FALSE NEGATIVE |
| **Predicted NEGATIVE** | FALSE POSITIVE | TRUE NEGATIVE |

Table 2 – Confusion Matrix

TP represents positive records that were correctly classified by the model, while FP are positive records that were incorrectly classified by the model.

FN represents negative records that were incorrectly classified by the model and TN are negative records that were correctly classified by the model. These values allow the calculation of metrics like Precision, Recall and Specificity.

Precision, or positive predictive value, is the probability of correctly classified records being relevant. Recall, also known as Sensitivity, or true positive rate, is the probability of relevant records being correctly classified. Specificity, or true negative rate, is the proportion of negative records that were correctly classified.

### 2.4.5 Model Comparison

This evaluation technique is based on the fact that a classification problem can be solved by using statistical models.

It is a technique that tries to find the best model, through the use of approaches such as *Akaike Information Criteria* (AIC), *Bayesian Information Criteria* (BIC) and *Adjusted* $R^2$. These approaches are calculated for each model, and the model with the lower value can be selected as the best model.

## 2.5 Evaluation Metrics

The metrics that will be used to compare the proposed solution to the alternatives presented in the state of the art will be the Mean Absolute Error (MAE), the Normalized MAE, the Root Mean Square Error (RMSE), the Normalized Distance-based Performance Measure (NDPM), Accuracy, Precision, Recall and the F1 measure.

Due to the fact that some of these metrics are not common to every paper, the comparison will be made one-on-one with the proposed solution.

The formula for MAE is shown in equation 5, with $p_i$ representing the prediction and $r_i$ representing the true rating value for user $i$:

$$MAE = \frac{1}{n}\sum_{i=1}^{n}|p_i - r_i| \quad (5)$$

The normalisation of the MAE is shown in equation 6, with $r_{max}$ and $r_{min}$ representing the upper and lower bounds of the ratings, respectively.

$$NMAE = \frac{MAE}{r_{max} - r_{min}} \quad (6)$$

RMSE has a lower tolerance for errors, amplifying the absolute errors between predicted and the actual values of ratings, showing a higher error value than the MAE. Equation 7 shows the RMSE's formula, with $X_{obs,i}$ corresponding to the value from the test set and $X_{model,,i}$ corresponding to the estimated value:

$$RMSE = \sqrt{\frac{\sum_{i=1}^{n}(X_{obs,i} - X_{model,,i})^2}{n}} \quad (7)$$

The NDPM formula is shown in equation 8:

$$NDPM = \frac{C_u^- + 0.5\, C_u^0}{C_u} \quad (8)$$

Assuming that the available data is that of real and predicted ratings of items for a user $u$: $C_u$ represents " (…) the number of pairs of items for which the real ranking asserts an ordering (i.e. not tied), that is the number of pairs with different values of the ratings."; $C^+$ represents "(…) the number of pairs for which the model ranking asserts the correct order (…)"; and $C^-$ corresponds to "(…) the number of pairs for which the model ranking asserts the (…) incorrect order (…)" **(Aleksandrova et al., 2016)**.

For the next metrics, the values used are those gathered from the creation of the Confusion Matrix, as presented previously: True Positive (TP), False Positive (FP), False Negative (FN) and True Negative (TN).

Accuracy represents the ratio of correctly predicted items in the amount of predictions made. Its formula is represented in equation 9:

$$Accuracy = \frac{TP+TN}{TP+FP+FN+TN} \quad (9)$$

As stated before, Precision is the probability of correctly classified records being relevant, and is represented by equation 10:

$$Precision = \frac{TP}{TP+FP} \quad (10)$$

As also previously stated, Recall is the probability of relevant items being correctly classified and is represented in equation 11:

$$Recall = \frac{TP}{TP+FN} \quad (11)$$

The F1 measure, also referred to as the F1-score, is the harmonic mean between Precision and Recall and is represented in equation 12:

$$F1\ measure = \frac{2(Precision \times Recall)}{Precision+Recall} \quad (12)$$

## 2.6 Hypotheses

The hypotheses to be tested will be that the proposed solution performs better than the presented alternatives, and, as stated in the previous section, the comparison of the proposed solution will be made one-on-one. The proposed approach's results will be compared separately with each of the methods presented in the papers analysed in the next section.

## 2.7 Methodology

The methodology for the proposed solution's model evaluation will use leave-one-out validation which was chosen because the models will be evaluated CS user by CS user, since there are only 32 CS users.

However, as each CS user has 19-20 ratings, five-fold cross validation will be used and each CS user's ratings will be separated into five (5) groups, with four groups used for model training and one for model testing.

After the model creation, these will be evaluated using the confusion matrix, which will allow the calculation of relevant metrics such as the model's Accuracy, Precision, Recall and F1 measure.

## 2.8 Statistical Tests

The statistical tests that will be conducted for the comparisons of all the approaches will be the “Student’s t test for paired samples”.

These tests were chosen for the following reasons: the fact that the comparison will be made using the same users, from the same dataset means that the samples will be paired; as stated previously, the comparison between the proposed solution and its alternatives will be made one-on-one, which means that it comprises two groups; and, finally, the choice to use the test of normal distribution was due to the fact that even though there is a small amount of data being used, there are more than the recommended thirty values. In this case there are thirty two (32) CS users.

Due to the fact that some of the papers don’t have enough information on the metrics used or don’t use metrics compatible with the evaluation metrics calculated for the proposed solution, not all papers previously presented will be compared with the proposed solution.

However, the papers that have compatible metrics to the proposed solution will be compared with it through the use of the “Welch two sample t test”.

This happens with these papers due to the fact that they either use anonymous data or from a different dataset from the one used in our solution (i.e. the “Netflix” dataset, which is no longer accessible, and the “BookCrossing” dataset). There are also a couple of papers that use “MovieLens 100K” or a more extensive version of the MovieLens dataset (i.e. the “MovieLens 1M” dataset) but don’t identify the users that were used for the training and testing of their method, making it nearly impossible to have paired samples.

## 2.9 Related Work

As referred previously, Cold Start (CS) problems come from the data sparsity limitations that CF recommender systems have with recommendations for new users or new items. This has led many researchers to try and solve these types of situations. The following sections show some work on those fields.

The papers were selected due to having interesting approaches, being recent and/or using the “MovieLens” dataset. The last reason allows for a better comparison with the solution to be developed.

Each of the following sections introduces different papers, starting with the method proposed by the authors and how it works, its results when applied to the dataset and its comparison with other methods chosen by the authors, followed by the their conclusions.

### 1) "Facing the cold start problem in recommender systems"

In this paper, **(Lika et al., 2014)** propose an algorithm that tries to deal with situations where a CF recommender system tries to make a recommendation for a new user but has no data available regarding said user's preferences. The proposed algorithm works in three phases:

1. The first phase has the purpose of classifying the user into a specific group, using classification techniques with algorithms like C4.5 or Naive Bayes.
2. In the second phase, the algorithm uses an intelligent technique with the purpose of finding the neighbours of the new user. This technique makes use of the new user's characteristics, utilizing the ones considered more important to find other users of the same group that are most compatible with that characteristic.
3. And, in the third phase, the outcome is calculated through the use of prediction techniques that estimate the new user's ratings.

The algorithm takes into account a user's demographic data, like its age, occupation and gender, with each attribute being given a specific weight for a scenario.

The authors applied their algorithms to the "MovieLens 1M" dataset, having created four distinct scenarios.

In the first scenario, their system focused primarily on the 'age' parameter , with the weights given to the parameters being 'age'=0.6, 'occupation'=0.3 and 'gender'=0.1; in the second scenario, its focus was on the 'occupation' parameter (i.e. 'age'=0.3, 'occupation'=0.6 and 'gender'=0.1); in the third scenario, it was focused on the 'gender' parameter (i.e. 'age'=0.3, 'occupation'=0.1 and 'gender'=0.6); and, in the fourth and last scenario, it considered all parameters equally, with the weighs being 'age'=0.33, 'occupation'=0.34 and 'gender'=0.33.

The algorithms they used were variations of the C4.5 algorithm, referred to as $C^2$ and $C^M$, as well as the Naive Bayes algorithm.

The evaluation metrics used by the authors to evaluate the algorithm's performance were the Mean Absolute Error (MAE) and the Root Mean Square Error (RMSE).

In all scenarios, the $C^2$ algorithm showed the best performance. The authors also concluded that, as the number of users grows, so do the MAE and RMSE's values decrease.

The aforementioned results led the authors to the conclusion that weights for each of the demographic attributes don't play an important role when the amount of users is below 1,000. They then increased the amount of users to a range from 1,000 to 5,000.

With this increase in the number of users, using the first scenario, they found that the difference in performance becomes smaller as the amount of users increase, and that the $C^2$ algorithm is heavily affected by the increase in the amount of users, no longer having the overall best performance and lowest values for MAE and RMSE, being replaced by the Naive Bayes and $C^M$ algorithms.

The authors then conclude that the system they propose works better in cases where there are more registered users.

### 2) "Simultaneous co-clustering and learning to address the cold start problem in recommender systems"

**(Luiz et al., 2015)** propose a hybrid recommender system, based on SCOAL, which tries to solve a pure Cold Start problem (i.e. no collaborative information available for new users).

The SCOAL algorithm uses an iterative approach of divide-and-conquer that combines clustering and learning tasks **(Luiz et al., 2015)**.

The authors' system is divided into two modules: one with simultaneous co-clustering and learning models, and the other, with the Cold Start recommendations.

The first module's purpose is finding user groups with similar interests, and building prediction models for each one of those groups.

The second module is used to identify the user group to which the new user belongs to, so that the most appropriate prediction model can be applied to said user's recommendations.

The author's proposed hybrid approach is used in the second module, and it's divided in two components:

1. **Cluster with Minimum Error (CME):** this approach is based on the assumption that a minimum number of ratings must be provided for a new user, so that the system can build said user's profile.
2. **Pure Cold Start Problems:** In this component, the authors use the row clusters identified by the SCOAL algorithm to build a classification model. Then, there is the generation of a dataset where its instances are the registered users, described by their attributes, also including their respective cluster labels, which are used as class labels in the classification process. Afterwards, a classifier that indicates the best prediction model for the new user is built.

The authors used three popular classifiers in their approach: Naive Bayes, J48 Decision Tree and Logistic Regression. Then, they optimised the classifier outputs through Weighted Prediction or Dynamic Classification.

For Weighted Prediction, the classifiers used were Naive Bayes, J48 Decision Tree, and Logistic Regression. For Dynamic Classification only the Naive Bayes classifier was used.

The authors use three datasets to evaluate their approach: "MovieLens 100K", "Jester" and "Netflix".

They created two scenarios: a partial Cold Start, with a small number of ratings per user available; and a pure Cold Start, with no ratings available for a new user.

Their experimental setup involved a 10-fold Cross Validation and the calculation of the Normalized Mean Absolute Error (NMAE). The SCOAL algorithm was given the arbitrary values of U=4, for the number of row clusters, and V=4, for the number of column clusters.

In the partial Cold Start scenario, the amount of available ratings for new users varied from 1 to 20 and three similarity measures were used for comparison: the Pearson Correlation Coefficient, the Cosine similarity measure and a constrained variant of the Pearson Correlation Coefficient, that "(…) allows only pairs of ratings on the same side (e.g. positive or negative ratings) to contribute to the correlation (…)" **(Luiz, et al., 2015)**. In this scenario, the authors concluded that the CME component presented overall better results than its counterparts.

In the pure Cold Start scenario, the authors defended that the Weighted Prediction and Dynamic Classification approaches could be used to address the problem of a lack of ratings for new users. In this scenario, the Dynamic Classification component outperformed the Weighted Prediction component in most situations.

The authors then concluded that the CME component is best suited for incremental Cold Start scenarios, while the Dynamic Classifier component works best in pure Cold Start scenarios.

### 3) "Identifying representative users in matrix factorization-based recommender systems: application to solving the content-less new item cold-start problem"

**(Aleksandrova et al., 2016)** propose an approach to automatically interpret the latent features that resulted from the application of Matrix Factorization (MF) to the system's users, without modifying the Matrix Factorization model used.

Matrix Factorization is based on an assumption that a rating matrix's meaning can be found in a small group of latent features, and it uses the rating matrix to obtain two low-rank matrices that represent the relations of the latent features with both users and items.

The authors state that the "(...) latent features in MF represent the relations between users and items (Koren et al.2009), in this work we consider that feature-related users will represent the preferences of other users of the system. Thus, these feature-related users will be referred to as representative users (...)" **(Aleksandrova et al., 2016)**.

The authors then explain how the representative users are found and their part in the process of making recommendations with Matrix Factorization, presenting the three phases needed for their identification: the normalization of the matrix that resulted from the factorization of the rating matrix; the formation of groups of pre-image candidates; and the identification of the representative users. Then, the representative users found are used in the rating of new items.

The authors used the "MovieLens 100K" and "Jester" datasets to evaluate their approach.

Their experimental setup involved a 5-fold Cross Validation, with 80% of the rating matrix used for the training sets and 20% for the test sets. The final result is the mean value of 5 values obtained for each fold.

In the non-Cold Start scenario, 80% of the rating matrix's randomly chosen ratings are used for model training and the remaining 20% are used for model testing.

The metrics used for model evaluation were the Normalized Root Mean Square Error (NRMSE), Normalized Distance-based Performance Measure (NDPM), Relative Deterioration (DET) and Test Coverage (COV).

When trying to identify the optimal number of features, the authors used various values for the regularization parameter, with values from 0 to 300, in increments of 5, for the "MovieLens 100K" dataset and from 0 to 30, with increments of 1, for the "Jester" dataset.

The difference between the error values for the different numbers of features for the Normalized Root Mean Square Error was of 0.0109 for "MovieLens 100K" and of 0.0007 for "Jester". For the Normalized Distance-based Performance Measure, it was of 0.0135 for "MovieLens 100K" and of 0.0064 for "Jester".

The authors then draw the conclusion that, when the optimal value of the regularization parameter is used for each number of features, the difference between the error values for the different numbers of features is insignificant.

Cold Start scenarios for both "Jester" and "MovieLens 100K" are then created, and compared with other Matrix Factorization-based models and with the RBMF approach.

In the experiments with the "Jester" dataset, based on the fact that not all users from the set of seed users might be able to provide ratings on new items for different reasons, the authors use the Test Coverage measure to better analyse the results. According to the authors, the proposed approach outperforms alternative methods of seed user identification, as well as the benchmark RBMF approach, in the Normalized Distance-based Performance Measure.

The authors defend that "(...) when using representative users as seeds, the unknown ratings from seed users can be replaced with the ratings of other closest candidates for being representative users (User-filling procedure), that allows make predictions for more new items" **(Aleksandrova et al., 2016)**.

The experiments with the "MovieLens 100K" tried not only to check if the proposed approach is still the best among all the alternatives, but also to find the filling procedure that guarantees the best performance for the proposed approach. The results of these experiments supported the results from the experiments with the "Jester" dataset, in regards to the proposed approach's superior performance with the Normalized Distance-based Performance Measure and that the use of the ratings of the closest candidates, instead of using mean-filling procedures, improves the quality of the items' ranking.

The authors conclude that the proposed use of representative users as seed leads to better predictions for ratings. They also conclude that the proposed approach has a better ranking than the RBMF approach.

### 4) "Collaborative filtering and deep learning based recommendation system for cold start items"

In this paper **(Wei et al., 2017)** the authors propose a method to be implemented in two types of Cold Start situations: dealing with a case of Cold Start where there are no ratings available, referred to as Complete Cold Start (CCS); and dealing with a case of Cold Start where there is a small number of ratings available, referred to as Incomplete Cold Start (ICS).

Due to CF models not being able to estimate the ratings of items in a Cold Start situation, the proposed method gathers additional content descriptions for said item, which are processed and used in the deep learning process of stacked denoising autoencoders (SDAE), with the purpose of predicting item ratings.

SDAE is a deep neural network, comprised of many denoising autoencoders (i.e. DAEs), with each layer working as a DAE and trained to minimize the errors when reconstructing its input. The first layers of the SDAE try to find the features in the noise-filled input, while the last layers try to reconstruct the resulting input in the output.

The proposed method uses the latent factor-based variant of 'timeSVD++' as its CF model.

The model used for the rating prediction in a situation of Complete Cold Start encompasses a content similarities-based approach and the 'timeSVD++' model.

The process starts with the use of a similarity measure to find the items in a non-Cold Start situation that are most similar to the ones with the Cold Start situation. Given that, the predictions for the items with the Cold Start situation are made from the non-Cold Start items that are most similar to them.

The authors state that, in the proposed method's model, "(...) the content features learned from the SDAE are utilized instead of the item features hidden in the rating matrix in the IRCD-CCS model for the CCS items." **(Wei et al., 2017)**, with IRCD-CCS being the nomenclature used when referring to the proposed method's approach for a Complete Cold Start situation.

The model used for the rating prediction in a situation of Incomplete Cold Start differs slightly from the previous one, namely through modifications made to 'timeSVD++', with the application of the features learnt by the SDAE into the process for latent factor training.

The authors used the "Netflix" dataset and also used data collected from IMDB regarding the items' (i.e. movies) plots to extract item content information.

Their experimental setup involved the division of the dataset into training and test sets, for both items with a Complete Cold Start situation and items with an Incomplete Cold Start situation.

The items were then ordered by the timestamp of their first received rating and divided into one of 100 intervals throughout the entire timespan of the dataset.

The metric used by the authors for model evaluation was the Root Mean Square Error (RMSE).

In the Complete Cold Start scenario, the proposed approach for said situation was compared with three recommender models: Top-of-All (ToA), Top-of-User (ToU) and Simple Average (SA).

The Top-of-All approach predicts the ratings for the items in a Complete Cold Start situation based on the top-*n* items not in a Cold Start situation that they are most similar to.

The Top-of-User approach predicts "(...) the ratings by a user for CCS items from their *M* most similar non-CS items within the set of non-CS items rated by the user (...)"**(Wei et al., 2017)**.

In the Simple Average approach, every user's rating of an item in a Complete Cold Start situation is set to the average of said user's ratings.

In the experiment, the proposed method presented the best overall results against all the other approaches.

The authors defended that those results showed an improvement on RMSE of about 0.05 when compared to Top-of-User, which was the second model with the best results.

They also explained the improvement found in the RMSE values for both the proposed method and the Top-of-User approach as the increase in the *M* variable, and the poor performance of the Top-of-All approach, due to it working best with smaller values for *M*.

In the Incomplete Cold Start scenario, the proposed approach for said situation was compared with four recommender models: ALS, SGD, 'timesSVD++' and CDL.

The comparison shows the proposed method for the Incomplete Cold Start situation as having the best overall results for the RMSE.

The authors state that, from the results, SGD is able to make better predictions for items in an Incomplete Cold Start situation than ALS. They also defend that it's because of the inclusion of deep learning and content information that the proposed approach is able to improve on 'timeSVD++'.

After comparing the proposed approach to 'timeSVD++', the authors note that their approach can reduce the values for RMSE in fewer iterations than 'timeSVD++'.

They also compared both of their approaches to find which worked best in a situation with very few ratings, reaching the conclusion that, in an Incomplete Cold Start situation, if there are very few ratings available (e.g. 3 or less per item), the approach that should be used is the one that deals with items in a Complete Cold Start situation.

The authors conclude that both of their approaches outperformed the existing baseline approaches for the recommendation of items in a Cold Start situation, also stating the large impact of including the time and item content information in the system.

### 5) "Context-Aware Personalization Using Neighborhood-Based Context Similarity"

**(Otebolaku & Andrade, 2017)** propose an algorithm that predicts a user's preference in a certain context, using the past experiences of his neighbours in that same context. The algorithm creates contextual profiles for the users, based on users' preferences in different contexts and it's based on the k-Nearest Neighbours approach, using the Pearson Correlation Coefficient to predict the user's preferences.

When creating a contextual user profile, the algorithm takes into account the user's actions in a certain context. The authors give the example of the information that characterizes a mobile user's consumptions as: activity, location, weather information, location illumination or noise level, time, etc.

The proposed system adopts the relevance feedback method, with the purpose of getting a user's opinion on recommendations, which, according to **(Otebolaku & Andrade,**

**2017)**, is a method that can speed up the process of learning a user's preferences while also increasing the quality of the recommendations.

When trying to recommend something to a new user, the system first determines said user's current contexts, to then, search all contextual user profiles, to try to find users with similar contexts. It defines those users as the neighbours of the new user and finds the *n* most similar users. Finally, the profiles of those users are the ones used to predict the preferences of the new user.

The authors used two sets of data for the evaluation of their approach: 4,500 movie metadata from "The Movie Database" (i.e. theMovieDB) crossed with more movie metadata from the "Internet Movie Database" (i.e. IMDB); and contextual user data retrieved from 200 anonymous users from the Faculty of Engineering, in the University of Porto.

The distribution for the anonymous users is as follows: 80% students, 9% researchers, 2% professors, 7% professionals and 2% others.

The user data collected was the consumed (i.e. watched/rated) movie's genre, language, country, (release) date and duration, as well as the user's context, which varied according to the place where the movie was consumed (i.e. home, cinema or office), its day (i.e. weekday or weekend), its time (i.e. morning, afternoon or evening) and how it was consumed (i.e. sitting, standing or walking).

The solution was implemented as part of the authors' Context-Aware Personalized Multimedia Recommendations application, which consisted in a mobile client application for handheld devices and a server application. The server contains the contextual user preference model, receiving contextual recommendation requests from the mobile client and then processing said requests in order to determine the new user's preferences.

Afterwards, it retrieves information relevant for those preferences from external sources, such as "theMovieDB" and "IMDB", and ranks them according to the user's preferences. Then, the top *k* items in that list are returned to the mobile client, and displayed as the recommendations for the user.

The evaluation metrics used by the authors to evaluate the algorithm's performance were the Average Precision (AP), used for every recommendation, and Mean Average Precision (MAP), used for all recommendations.

The recommendations were generated 10 times, with *k=5*, meaning the number of recommendations made were five for each time. The evaluation was made in four different context types (i.e. contexts), and once without a context, with 20 users representing new users, meaning they all had little or no information in their user profile model.

The results for every context had an Average Precision well above the run without a context, and had a variation from 30-70%.

The authors defend that the two best performing contexts had higher values for Average Precision than the remaining two because neighbours consumed more movies in them and imply that if the neighbours had consumed more movies in those lower Average Precision contexts, its results would have been better.

The Mean Average Precision was then used, in conjunction with the Confidence Level (CL) to better explore the results. For the different contexts, the Mean Average Precision varied between 0.412 and 0.640, while the scenario without a context had a Mean Average Precision of 0.224.

With the support of these results, the authors then reiterate that the quality of contextual recommendations tends to improve with the increasing of the number of recommendations.

By analysing the variances in performance with different neighbourhood sizes, the authors also defend that the quality of recommendations improves with the increasing of the size of the neighbourhood.

The authors defend that the main contribution of their document lies in the use of contextual information as a means of finding similar users/neighbours whose preferences would be of interest to a new user. They also defend that their paper shows that the Cold Start problem's impact can be minimized, with or without rating data.

The authors then conclude that the similarity between contexts can be "used to effectively mitigate the effects of the user-based cold-start problem" **(Otebolaku & Andrade, 2017)**.

### 6) "User-based Clustering with Top-N Recommendation on Cold-Start Problem"

**(Yanxiang et al., 2013)** propose a method to get better recommendations for users without their history information (i.e. new users). The proposed method works in three phases:

1. The first phase consists of the computation of the similarity matrix of the users. The authors use the Vector Cosine-based similarity measure to calculate the user similarities, due to the fact that it can measure the similarity between two users without using rating information.
2. In the second phase, the algorithm uses clustering to divide the users into different groups. The clustering method used is the broad first searching method of graph theory.
3. And, in the third phase, the average rating of each item of all users in the same group is calculated, and for each group, the top *n* items with the highest average rating are recommended.

The data used for the evaluation of this method came from the "MovieLens" dataset, from where they extracted the data of 100 users. This data was then randomly partitioned into training and testing sets, with two different ratios: 60% training and 40% testing for the first; and 80% training and 20% testing for the second. Each group's recommendations consist of its top 5 items.

In this paper, the evaluation metrics used by the authors are Mean Absolute Error (MAE), Root Mean Square Error (RMSE) and Precision.

Two different scenarios were created, one for each distribution of the data. They then calculated the Mean Absolute Error, Root Mean Square Error and Precision for each scenario.

Based on the results, the authors defend that the scenario with the larger training set wields better recommendation accuracy. They also state that less information regarding users' ratings can lead to a lower quality for the recommender system.

The authors use the distribution with the better overall results for the comparison (i.e. the scenario with the larger training set).

When calculating the Precision, they compare their method to some baseline methods, such as the Triadic Aspect model (TAM) and traditional Collaborative Filtering (CF). The

Average Precision of their method is of 0.651, which is 4.8% better than TAM. Lastly, TAM has a 45% improvement compared with traditional CF.

For the calculation of the Mean Absolute Error, the baseline methods chosen by the authors to compare the proposed method with were SVD-CF and Co-clustering CF. The MAE of the proposed method is 0.78, being the lowest value of all methods by a substantial margin. The proposed method shows an improvement of 5.4% on SVD-CF and of 6.7% on Co-clustering CF.

For the Root Mean Square Error comparison, the authors choose Functional Matrix Factorisations (FMF) and Decision Trees (DT) as the baseline methods to compare with their method. The proposed method had the lowest value, with it being 0.9321, showing an improvement of 1.2% and 3.2% for the FMF and DT methods, respectively.

They then evaluate the proposed method's average predicting accuracy for ratings. For that the authors randomly select 25 recommended items, comparing their real ratings with the predicted ones. They find that more than 60% of the items have a prediction accuracy of 0.9 and that the similarity between real and predicted ratings is of 0.9627.

The authors conclude that the method they propose shows an improvement in the accuracy of recommendations and has advantages for the Cold Start problem.

They also state that, while the accuracy of the predicted ratings is good, its variance is not that good. For this purpose, they defend that additional work on the clustering algorithm is needed.

### 7) "Using Association Rules to Solve the Cold-Start Problem in Recommender Systems"

**(Shaw et al., 2010)** propose the expansion of the user profile as a way to solve the Cold Start problem. The method derives association rules from the dataset, in order to give more information to the recommender, like patterns and associations of items. The authors also aim to check the performance of the method with redundant or non-redundant rules.

The recommender used to test the authors theory is the Taxonomy-driven Product Recommender (TPR). In a first phase, the TPR creates taxonomy-driven profiles. The taxonomies used have a structure similar to that of a Decision Tree. This means that a taxonomy has several topics/categories in a multileveled structure, with each topic having a parent, or super-topic, while also being able to have many children, or sub-topics.

Through the use of this user profile, the method is able to focus on a user's topics of interest, rather than singular items.

The authors then create a transactional dataset from the previously created user profiles, which is searched for patterns, and association rules are created based on those patterns.

Then, they use the user profiles in conjunction with the association rules to create a list with all possible combinations from the group of topics for each user. Each combination of topics is then cross-referenced with the association rules, to find the association rules that have the *left hand side* (LHS) that matches the combination. The *right hand side* (RHS) from those rules is then added to the respective user profile, with an assigned weight, based on the association rule's LHS.

The authors also defend the need to place limitations on the expansion of user profiles, using 1 to 5 rules in their experiments. They randomly divided the data into training and test sets.

The data used for the evaluation came from the "BookCrossing" dataset, which contains data pertaining to users, books and the ratings given to the books by the users, and the taxonomy tree and descriptors information came from "Amazon.com". The data used contains the information of 92,005 users (i.e. transactions), 12,147 topics and 270,868 books.

The evaluation metrics used by the authors in this paper are Precision, Recall and the F1 measure.

Using a Confidence minimum threshold of 50%, the authors were able to derive 37,827 association rules with MinMax. Then, the authors find and try to expand all user profiles with 5 or less ratings, doing so for 15,912 profiles.

Up to 10 recommendations are then made to those expanded user profiles, which are compared with the same recommendations made for the same profiles, before being expanded.

The set of user profiles without expansion has a precision of 0.00619, a recall of 0.0571 and the F1 value of 0.0112. The expanded user profiles can have values of precision, recall and F1 measure up to 0.00815, 0.0754 and 0.0147, respectively. According to the authors, this represents an overall improvement of around 31.5% over the profiles without expansion.

The authors go on to defend that, while the expanded profiles take longer to make recommendations for, that time is no different from a longer user profile without expansion.

When trying to test their theory that non-redundant rule sets perform as well as redundant rule sets, the authors use four different rule mining algorithms (i.e. MinMax, ReliableBasis, MinMax with HRR and ReliableBasis with HRR) with the same Confidence and Support minimum thresholds. The results presented by the authors regarding the differences in performance for each used algorithm showed no major difference due to redundancy or lack thereof.

The authors conclude that the proposed approach can improve the performance of a recommender system in a situation of Cold Start. They also defend that the results they got when comparing the performance of redundant to non-redundant rule sets showed their performance to be equivalent.

### 8) "A CF approach to mitigate the new user cold start problem"

In this paper **(Bobadilla et al., 2011)** propose the use of a similarity measure that reduces the impact of a User Cold Start problem on recommenders. They defend that their measure gathers more information than the "traditional" similarity measures when comparing two users, namely information based on the distribution and number of votes/ratings made by each of the users being compared.

In their approach, the authors defend that a positive user vote has a numerical value of 4 or 5, whilst they consider votes with values below 4 to be non-positive. They then justify their choice with the results of the experiments conducted that shows that their separation of the data into positive and non-positive "(…) not only does not worsen the precision/recall measurements, but rather it improves them both (…)" **(Bobadilla et al., 2011)**.

The proposed measure involves the creation of a linear combination of a group of simple similarity measures, with its respective weights being obtained through a neural learning optimisation process.

One of the simple measures used in the proposed approach is the Jaccard measure, with the purpose of processing the non-numerical information of the votes. The remaining measures use the difference between two votes when comparing them.

After defining the weights, it (i.e. the proposed measure) can be used to find the k-Nearest Neighbours of each Cold Start user to whom the system is trying to recommend items.

The authors decided on using a leave-one-out cross validation when evaluating the recommender's results, due to the low amount of items voted on by Cold Start users, in order to have the highest possible amount of training items. They also decided to define a Cold Start user as a user that has voted between 2 and 20 items.

The data used in the evaluation process came from the "Netflix" and the "MovieLens 1M" datasets.

The evaluation metrics used by the authors in this paper were the Mean Absolute Error (MAE), Coverage, Precision and Recall.

The authors created two experiments, with the first one being divided into two scenarios: one for checking the results of the Mean Absolute Error and Coverage for the range of neighbourhoods between 100 and 200; and another for checking the values of Precision and Recall for a range of number of recommendations between 2 and 20.

In the second experiment, the authors defined the values for the neighbourhood as 700 and the number of recommendations as 10, with the range of votes cast by Cold Start users being between 2 and 20.

The conclusions of the authors regarding the first experiment are that the proposed measure improves the quality of the predictions, when compared to "traditional" measures, when the measures are applied to Cold Start users. They also find that one of the comparison metrics (i.e. PIP) has better results, but only for neighbourhoods of fewer than 500.

The authors also show that the proposed measure has a negative aspect when the neighbours that are selected have a similar number of votes to the user to whom the recommendations are being made. In this case, the Coverage metric is worse.

The Precision metric has overall better results for the proposed measure, which led the authors to define the neighbourhood size for the second experiment as 700. The authors also find that the Recall metric also has overall better results for the proposed metric.

In the second experiment, with the "MovieLens" dataset, the proposed measure performs better overall, having lower MAE values and higher Precision and Recall. However, it has average values for the Coverage metric, not being the best, while also not the worst.

When using the “Netflix” dataset, the proposed measure has similar results to the ones from the “MovieLens” dataset, performing very well with MAE, Precision and Recall and having average values for Coverage.

The authors conclude by stating that the proposed metric provides results that are able to obtain better values for MAE, Precision and Recall. They also state that while the Coverage metric has worse results, the measures that have better results are seven times slower than the proposed measure.

### 2.9.1 Paper Summary

In this subsection, three tables are used to show a brief description of each of the papers presented in the previous sections’ related work.

<table>
<tr><th>Reference</th><th>Dataset</th><th>Technologies</th></tr>
<tr><td>(Lika et al., 2014)</td><td>MovieLens 1M: 1,000,209 ratings of 3,900 movies made by 6,040 users</td><td>Three Data Mining techniques were applied: two variations of C4.5 (i.e. $C^2$ and $C^M$) and Naïve Bayes.<br>Approximate results for 700 users:
<table>
<tr><td></td><td>$C^2$</td><td>$C^M$</td><td>NB</td></tr>
<tr><td>MAE</td><td>0.80</td><td>0.85</td><td>0.82</td></tr>
<tr><td>RMSE</td><td>1.01</td><td>1.05</td><td>1.02</td></tr>
</table>
Approximate results for 5,000 users:
<table>
<tr><td></td><td>$C^2$</td><td>$C^M$</td><td>NB</td></tr>
<tr><td>MAE</td><td>0.74</td><td>0.735</td><td>0.736</td></tr>
<tr><td>RMSE</td><td>0.94</td><td>0.935</td><td>0.935</td></tr>
</table></td></tr>
<tr><td>(Luiz et al., 2015)</td><td>MovieLens 100K: 100,000 ratings of 1,682 movies made by 943 users</td><td>Three Data Mining techniques were applied: Naive Bayes, J48 Decision Tree, and Logistic Regression.<br>Two types of optimisation were applied to the Data Mining techniques: Weighted Prediction (WP) and Dynamic Classification (DC).<br>Approximate results for the MovieLens 100K dataset:
<table>
<tr><td></td><td>NB-WP</td><td>J48-WP</td><td>LR-WP</td><td>DC</td></tr>
<tr><td>NMAE</td><td>0.191</td><td>0.193</td><td>0.191</td><td>0.186</td></tr>
<tr><td>CPU time (s)</td><td>0.031</td><td>0.027</td><td>0.028</td><td>4.789</td></tr>
</table></td></tr>
<tr><td>(Aleksandrova et al., 2016)</td><td>MovieLens 100K: 100,000 ratings of 1,682 movies made by 943 users</td><td>The Data Mining technique applied was: Matrix Factorization.
<table>
<tr><td>Number of Features K</td><td>5</td><td>50</td><td>500</td></tr>
<tr><td>Optimal NRMSE</td><td>0.2364</td><td>0.2473</td><td>0.2435</td></tr>
<tr><td>Optimal NDPM</td><td>0.3050</td><td>0.3056</td><td>0.2982</td></tr>
</table></td></tr>
</table>

Table 3 – Synopsis of the Presented Papers (1/3)

<table>
<tr><th>Reference</th><th>Dataset</th><th>Technologies</th></tr>
<tr><td>(Wei et al., 2017)</td><td>Netflix: more than 100 million ratings for 17,770 movies made by 480,189 users</td><td>The Data Mining technique applied was: Deep Neural Network.<br>Approximate RMSE for 20 most related items:
<table>
<tr><th>Size of Test Dataset</th><th>ToA</th><th>ToU</th><th>SA</th><th>Complete CS method</th></tr>
<tr><td>100</td><td>1.155</td><td>1.133</td><td>1.146</td><td>1.075</td></tr>
<tr><td>300</td><td>1.134</td><td>1.140</td><td>1.157</td><td>1.096</td></tr>
</table>
Approximate RMSE for 100 most related items:
<table>
<tr><th>Size of Dataset</th><th>ToA</th><th>ToU</th><th>SA</th><th>Complete CS method</th></tr>
<tr><td>100</td><td>1.224</td><td>1.133</td><td>1.146</td><td>1.053</td></tr>
<tr><td>300</td><td>1.217</td><td>1.127</td><td>1.157</td><td>1.082</td></tr>
</table></td></tr>
<tr><td>(Otebolaku & Andrade, 2017)</td><td>theMovieDB and IMDB: 4,500 movie metadata<br>Anonymous User Data: 200 users</td><td>Use of contextual data combined with k-Nearest Neighbours
<table>
<tr><th></th><th>No Context</th><th>Best Performing Context</th><th>Worst Performing Context</th></tr>
<tr><td>Highest AP value</td><td>0.31</td><td>0.68</td><td>0.45</td></tr>
<tr><td>MAP</td><td>0.224</td><td>0.640</td><td>0.412</td></tr>
</table></td></tr>
<tr><td>(Yanxiang et al., 2013)</td><td>MovieLens: data of 100 random users</td><td>The Data Mining technique applied was: Clustering, with broad first searching method of graph theory.
<table>
<tr><th></th><th>Proposed Method</th><th>TAM</th><th>CF</th></tr>
<tr><td>Precision</td><td>0.651</td><td>0.63</td><td>0.45</td></tr>
</table>
<table>
<tr><th></th><th>Proposed Method</th><th>SVD-CF</th><th>Co-clustering CF</th></tr>
<tr><td>MAE</td><td>0.7877</td><td>0.83</td><td>0.84</td></tr>
</table>
<table>
<tr><th></th><th>Proposed Method</th><th>FMF</th><th>DT</th></tr>
<tr><td>RMSE</td><td>0.9321</td><td>0.941</td><td>0.96</td></tr>
</table></td></tr>
</table>

Table 4 – Synopsis of the Presented Papers (2/3)

<table>
<tr><th>Reference</th><th>Dataset</th><th>Technologies</th></tr>
<tr><td>(Shaw et al., 2010)</td><td>BookCrossing: 92,005 users, 12,147 topics and 270,868 books</td><td>The Data Mining technique applied was: Association Rules.<br>The following table shows the reduction in the pruned association rules, by algorithm
<table>
<tr><td></td><td>No. of Rules</td><td>Reduction</td></tr>
<tr><td>MinMax (MM)</td><td>37,827</td><td>0%</td></tr>
<tr><td>ReliableBasis (RB)</td><td>36,852</td><td>2.58%</td></tr>
<tr><td>MM with HRR</td><td>37,555</td><td>0.72%</td></tr>
<tr><td>RB with HRR</td><td>36,604</td><td>3.23%</td></tr>
</table>
<table>
<tr><td>Profiles</td><td>Precision</td><td>Recall</td><td>F1 value</td></tr>
<tr><td>Without Expansion</td><td>0.00619</td><td>0.0571</td><td>0.0112</td></tr>
<tr><td>With Expansion</td><td>0.00815</td><td>0.0754</td><td>0.0147</td></tr>
</table></td></tr>
<tr><td>(Bobadilla et al., 2011)</td><td>Netflix: 480,189 users, 17,770 items and 100,480,507 votes/ratings<br>MovieLens: 6,040 users, 3,706 items and 1,480,507 ratings</td><td>The Data Mining technique applied was: k-Nearest Neighbours.
<table>
<tr><td>Average Value</td><td>MAE</td><td>Coverage</td><td>Precision</td><td>Recall</td></tr>
<tr><td>Proposed Method</td><td>0.78</td><td>0.86</td><td>0.46</td><td>0.84</td></tr>
<tr><td>PIP</td><td>0.79</td><td>0.97</td><td>0.44</td><td>0.83</td></tr>
</table></td></tr>
</table>

Table 5 – Synopsis of the Presented Papers (3/3)

# 3 Value Analysis

This chapter has the purpose of presenting the value of the product being developed: a hybrid algorithm that is able to deal with Cold Start problems.

## 3.1 Opportunity

Recommender Systems have an important part in today's society, helping user's find what they are looking for faster.

However these types of systems have limitations. One of those limitations is Cold Start, a type of problem that still hasn't been completely solved.

A Cold Start problem occurs when a recommender does not have enough information to properly recommend items to users.

This makes the creation of a recommender that can properly deal with Cold Start problems very appealing.

There are already several different approaches to Cold Start problems, but there is no optimal approach.

This means that the solution to be developed could try to improve upon some of the existing approaches' weaknesses and use some of their strengths, possibly increasing the competitors' interest in using the solution.

In spite of that, there are already some implementations of recommender systems, which would act as competitors, like YouTube, Facebook, Amazon and Netflix.

## 3.2 Idea

Given the aforementioned opportunity, the need of a recommender system that is able to deal with Cold Start problems was found.

Being too broad of a concept, the solution needed to have a target use, which was defined as recommending movies for users to watch, reducing the possible competitors to systems like those adopted by Netflix. This also means that, if the solution has a good performance, the competitors could turn into clients.

The solution's purpose would be to make the system that uses it both user-friendly to new users, and able to recommend brand new movies that may be of interest to a user.

However, there are costs associated with the development of a hybrid algorithm, which involves factors such as: if the software used needs a paid license; the space needed for the development of the algorithm; and the time needed to develop it.

## 3.3 Product

Value has been defined as a need, an interest, an attitude, a preference, among others. Its creation is what the customers are looking for. The value that this product (i.e. solution) creates is the fulfilment of a necessity: that of movie recommendation or streaming services being unable to properly recommend relevant movies to new users, mainly because they lack information about said users' interests. Its value can also be defined as the interest of said movie streaming companies providing a better service to its clients.

A product's perceived value, or customer value, is the customer's opinion on the product, its benefits and sacrifices. The customer value of the proposed product would be the fact that it allows its clients to offer a better service to the consumers, improving the experience for brand-new consumers, which would be able to get movie recommendations with the same quality as if they had already been a user for some time.

Not to be confused with customer value, the value of a product for the customer is, according to Woodall, the presence of benefits that are either recognized as attributes or

outcomes. Woodall also refers to it as the result of the comparison of the product's sacrifices and benefits.

The pre-purchase (i.e. "ex ante") value for the customer would be focused on the creation and maintaining of a trust relationship between the organization and the customer. The fact that the product is easily adapted to a customer's specifications, needing to adapt the input data and the product's output according to the customer's needs, can be considered part of the pre-purchase value for the customer but it can also be considered part of the value for the customer at the point of trade (i.e. transaction), as it is considered a benefit in both situations.

The use of a technical support team to assist the customers in the integration of the product into their system would allow increasing the product's reliability and would also help the organization grow a relationship of trusts with its clients. The value for the customer for this support can be considered a benefit at the point of trade and in a post-purchase situation (i.e. "ex post").

The product's benefits would be focused on its quality, customization capabilities, reliability and flexibility. They would also be focused on technical competence, maintaining a good image and a trust-based relationship with its customers.

Its sacrifices would be the time, effort and monetary costs needed to develop and provide technical support for the product, as well as the costs associated with maintaining the relationship with the customers.

## 3.4 Value Proposition

A hybrid recommender system that can deal with Cold Start situations can be used to make recommendations to users when other recommender systems could not.

Movie recommendation services or movie streaming companies are always looking for a way to improve their services and increase their user's interests in continuing to use them.

Being able to immediately recommend movies to brand-new users has always proven to be a difficult task. Another problem arose when new movies were introduced in the system but were not able to be recommended due to not being rated by users.

The proposed product's objective is to deal with both situations, allowing its target customers (i.e. movie recommendation services or movie streaming companies) to keep, and even increase, their user-base.

## 3.5 Canvas

The Canvas model is often used to present business ideas. That is why it is used in this section.

This section shows a hypothetical situation in which this project represents a business idea for a start-up organisation.

Figure 2 shows the graphical representation of the Canvas model.

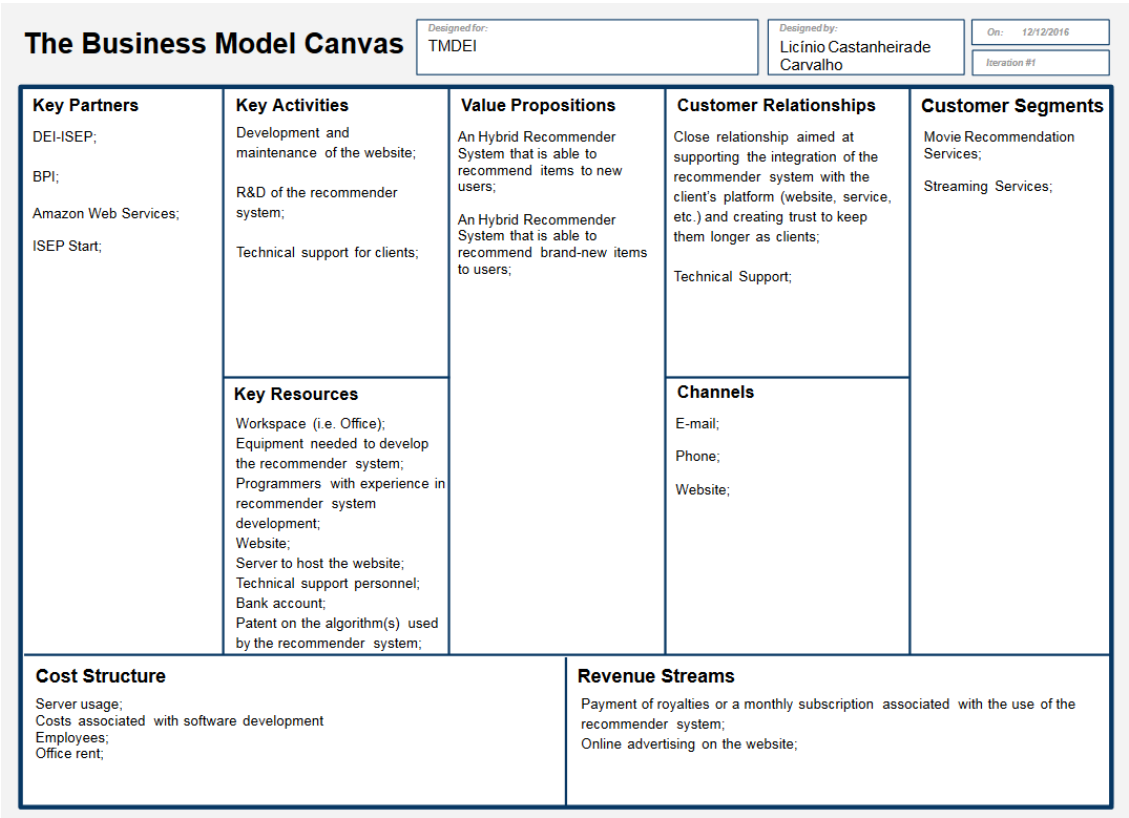


Figure 2 – Canvas Model

Customer segments are those to whom we are creating value. In this case, they are the companies that want to personalize or customize their service to their users and belong to one of two segments: Movie Recommendation Services and Streaming Services.

The segment of Movie Recommendation Services represents websites that want to recommend relevant movies to their users, while the Streaming Services segment wants to show their customers films or series they might be interested in, in order to sell more products.

The organisation's potential clients can include Netflix and MovieLens.

A simple way to define the organisation's value proposition is as the offer of a hybrid recommender system that is able to recommend movies to new users and also recommend brand-new movies to users.

With our product, we are delivering to our customers a way to recommend relevant movies to their new clients or even brand-new movies to clients who may have interest in them.

We are offering to each customer segment a way to be able to provide a more personalized service, showing only movies that their customers may be interested in.

To do that, we use several channels: through email, phone and a website. The first two channels can be used to contact our potential customers, while the website is used to advertise the product.

The email is a way to contact and be contacted by the clients and there is no financial cost if we use a free email service. However, the acquisition and maintenance of an email server would have great short-term costs. In this situation, the payment of a monthly fee to use an email service would have lower costs.

Like with the email, the phone is a way to contact and be contacted by the clients. However, it requires the creation of a contract with a telecommunications provider and should have a monthly payment structure for its costs.

The website is used to advertise the product. It also provides our contact information, should potential clients want to contact us. As the acquisition and maintenance of a server to host the website would have great short-term costs, the payment of a monthly fee to use an server to host the website could prove better, as it would have lower costs.

The relationships with our customers should be close, aimed at supporting the integration of the recommender system with the customer's platform, be it a website, a service, etc. Another relationship that should be established is a trust relationship, to keep

them as clients for longer. A relationship should also be created for the technical support, to make both the integration process and the product maintenance easier.

The cost of these relationships would involve costs associated with communication through telephones. Costs associated with email providers would only be applicable if an email server were acquired or a subscription to an email service were established.

The revenue streams for this business model would largely come from the payment of royalties or a monthly subscription associated with the use of the recommender system and a smaller part would come from the revenue from external advertisements on the website (e.g. Google Ads).

Even though the customers have been developing their own algorithms and recommender systems, if a better approach presented itself, most would try to use it. This is why one of the possible revenue streams is the payment of royalties.

The payment of both types of revenue streams could be through a money transfer, PayPal or credit card.

The organisation's key resources could be considered its workspace, the equipment needed to develop the recommender system, programmers with experience in recommender system development, the website and the server to host it, the technical support personnel, a bank account to receive the money transfers in and a patent on the algorithms developed, and used, for the recommender system.

The organization's key activities include the maintenance of the website, R&D of the recommender system, to continuously improve the recommender system, and the technical support for the customers.

Our key partners would be: Amazon Web Services, which would provide storage space in the cloud for the website and servers to process the algorithm's information; DEI-ISEP, which is the organization that proposed the development of this solution and would not only be able to assist in its development but also in future R&D for the recommender system; ISEP Start, where the workspace could be located at; and BPI, which could provide credit options as well as a bank account.

The cost structure of these activities would involve the server maintenance, the costs associated with the development of the website and the product, the employee salaries and the rent of the workspace.

## 3.6 Value Network & Value Chain

Using the business proposed in the Canvas model, in the previous section, a value network could be established between our organization, the customers and the key partners.

This happens because, with the customers, the relationships established provided value for the customer, both tangible, with the support in the integration of the product into their system, and intangible, with the creation of trust.

With the key partners, the monetary costs associated with the services provided are reduced, because of the creation of intangible value by our organization for them.

The use of Porter's value chain model would be very useful when analysing the business value of this product, as it is able to identify the organization's activities responsible for the satisfaction of the customer's needs. It would also help by identifying the importance of each of the organizations' processes and by structuring performance indicators.

Most importantly, it would allow analysing the organization from a top-down perspective, allowing the analysis from a more abstract point of view of the organization to a more detailed perspective of the organization's activities.

## 3.7 Analytic Hierarchy Process

The Analytic Hierarchy Process (AHP) can be used in the development of this product every time a decision has to be made.

An example of its application would be when which algorithm to use in a situation.

For example, there are four different algorithms (e.g. A, B, C, and D), and the criteria relevant for their comparison (e.g. their runtime, accuracy and mean absolute error).

First, we need to build a hierarchical decision tree, which can be found in Figure 3.

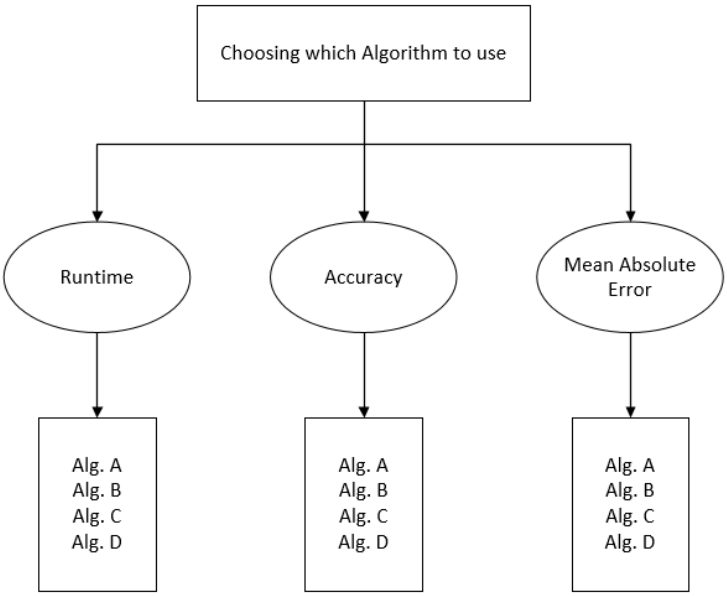


Figure 3 – Hierarchical Decision Tree

Then, a comparison table, in which to compare the various attributes, needs to be created.

Assuming that the algorithm's Accuracy is slightly more important than its Runtime and that its Mean Absolute Error is more important than its Accuracy, we obtain Table 6.

| | Runtime | Accuracy | Mean Absolute Error |
|---|---|---|---|
| Runtime | 1 | $^1/_2$ | $^1/_3$ |
| Accuracy | 2 | 1 | $^1/_2$ |
| Mean Absolute Error | 3 | 2 | 1 |
| Column Sum | 6 | $^7/_2$ | $^{11}/_6$ |

Table 6 – Comparison Table

Then, the table needs to be normalized by dividing each value by its columns sum, as shown in Table 7. This allows for the calculation of the relative priority, used in the priority vector.

| | Runtime | Accuracy | Mean Absolute Error | Relative Priority |
|---|---|---|---|---|
| Runtime | $^1/_6$ | $^1/_7$ | $^2/_{11}$ | 0.1638 |
| Accuracy | $^1/_3$ | $^2/_7$ | $^3/_{11}$ | 0.2973 |
| Mean Absolute Error | $^1/_2$ | $^4/_7$ | $^6/_{11}$ | 0.5390 |

Table 7 – Normalised Comparison Table

The relative priority column represents the priority vector, which has the criteria weights, showing the Mean Absolute Error as the most important, followed by Accuracy and then Runtime.

Next, we have to check the consistency of the relative priorities, calculating RC. For that we will need to convert Table 6 into a matrix.

$$\begin{bmatrix} 1 & ^1/_2 & ^1/_3 \\ 2 & 1 & ^1/_2 \\ 3 & 2 & 1 \end{bmatrix} \begin{bmatrix} 0.1638 \\ 0.2973 \\ 0.5390 \end{bmatrix} \cong \gamma_{max} \begin{bmatrix} 0.1638 \\ 0.2973 \\ 0.5390 \end{bmatrix}$$

$$\begin{bmatrix} 0.4921 \\ 0.8944 \\ 1.625 \end{bmatrix} \cong \gamma_{max} \begin{bmatrix} 0.1638 \\ 0.2973 \\ 0.5390 \end{bmatrix}$$

Afterwards, we calculate $\gamma_{max}$, which is as follows:

$$\gamma_{max} = average\left(^{0.4921}/_{0.1638}, ^{0.8944}/_{0.2973}, ^{1.625}/_{0.5390}\right) = 3.009$$

$$Then\ we\ can\ calculate\ the\ Consistency\ Index\ IC = {}^{(\gamma_{max} - n)}/_{n-1}$$

$$IC == {}^{(3.009 - 3)}\Big/_{(3-1)} = 0.0045$$

To calculate RC, we now do as follows:

$$RC = {}^{IC}/_{0.58} = {}^{0.0045}/_{0.58} = 0.0078,$$

$$as\ 0.0078 < 0.1, we\ can\ conclude\ that\ the\ relative\ priorities' values\ are\ consistent.$$

It is now that we can build the comparison matrices for the algorithms, by attribute.

| Runtime | Alg. A | Alg. B | Alg. C | Alg. D |
|---|---|---|---|---|
| Alg. A | 1 | 2 | 4 | 3 |
| Alg. B | $1/2$ | 1 | 3 | 2 |
| Alg. C | $1/4$ | $1/3$ | 1 | $1/2$ |
| Alg. D | $1/3$ | $1/2$ | 2 | 1 |
| Column Sum | $25/12$ | $23/6$ | 10 | $13/2$ |

Table 8 – Comparison Table for Runtime

| Runtime | Alg. A | Alg. B | Alg. C | Alg. D | Priority Vector |
|---|---|---|---|---|---|
| Alg. A | $12/25$ | $12/23$ | $2/5$ | $6/13$ | 0.4658 |
| Alg. B | $6/25$ | $6/23$ | $3/10$ | $4/13$ | 0.2771 |
| Alg. C | $3/25$ | $2/23$ | $1/10$ | $1/13$ | 0.0960 |
| Alg. D | $4/25$ | $3/23$ | $1/5$ | $2/13$ | 0.1611 |

Table 9 – Normalized Comparison Table for Runtime

| Accuracy | Alg. A | Alg. B | Alg. C | Alg. D |
|---|---|---|---|---|
| Alg. A | 1 | $1/2$ | $1/4$ | $1/2$ |
| Alg. B | 2 | 1 | $1/3$ | 1 |
| Alg. C | 4 | 3 | 1 | 3 |
| Alg. D | 2 | 1 | $1/3$ | 1 |
| Column Sum | 9 | $11/2$ | $23/12$ | $11/2$ |

Table 10 –Comparison Table for Accuracy

| Accuracy | Alg. A | Alg. B | Alg. C | Alg. D | Priority Vector |
|---|---|---|---|---|---|
| Alg. A | $1/9$ | $1/11$ | $3/23$ | $1/11$ | 0.1058 |
| Alg. B | $2/9$ | $2/11$ | $4/23$ | $2/11$ | 0.1899 |
| Alg. C | $4/9$ | $6/11$ | $12/23$ | $6/11$ | 0.5143 |
| Alg. D | $2/9$ | $2/11$ | $4/23$ | $2/11$ | 0.1899 |

Table 11 –Normalized Comparison Table for Accuracy

| Mean Absolute Error | Alg. A | Alg. B | Alg. C | Alg. D |
|---|---|---|---|---|
| Alg. A | 1 | $^1/_3$ | 2 | $^1/_2$ |
| Alg. B | 3 | 1 | 1 | 2 |
| Alg. C | $^1/_2$ | 1 | 1 | $^1/_2$ |
| Alg. D | 2 | $^1/_2$ | 2 | 1 |
| Column Sum | $^{13}/_2$ | $^{17}/_6$ | 6 | 4 |

Table 12 –Comparison Table for Mean Absolute Error

| Mean Absolute Error | Alg. A | Alg. B | Alg. C | Alg. D | Priority Vector |
|---|---|---|---|---|---|
| Alg. A | $^2/_{13}$ | $^2/_{17}$ | $^1/_3$ | $^1/_8$ | 0.1825 |
| Alg. B | $^6/_{13}$ | $^6/_{17}$ | $^1/_6$ | $^1/_2$ | 0.3703 |
| Alg. C | $^1/_{13}$ | $^6/_{17}$ | $^1/_6$ | $^1/_8$ | 0.1804 |
| Alg. D | $^4/_{13}$ | $^3/_{17}$ | $^1/_3$ | $^1/_4$ | 0.2669 |

Table 13 – Normalized Comparison Table for Mean Absolute Error

With the priority vectors found, we can now advance to the calculation of the best alternative. Using matrices, we obtain the following results:

$$Alg.A = (0.1638 \times 0.4658) + (0.2973 \times 0.1058) + (0.5390 \times 0.1825) = 0.2061,$$

$$Alg.B = (0.1638 \times 0.2771) + (0.2973 \times 0.1899) + (0.5390 \times 0.3703) = 0.3014,$$

$$Alg.C = (0.1638 \times 0.0960) + (0.2973 \times 0.5143) + (0.5390 \times 0.1804) = 0.2659,$$

$$Alg.D = (0.1638 \times 0.1611) + (0.2973 \times 0.1899) + (0.5390 \times 0.2669) = 0.2267.$$

According to these results, the algorithm that should be chosen was algorithm B.

As stated previously, this process can be applied to any type of decision in which the alternatives have more than one criterion.

These calculations were made as an example of what would be necessary if AHP had to be applied.

# 4 Design of the Solution

The design of the solution has to take into account that it is a hybrid recommender, which means its purpose is to be a more robust recommender system. It also has to be able to deal with Cold Start problems, as stated in this project's objectives.

## 4.1 Architecture

The previously mentioned challenges lead to the creation of separate components: one to find and classify the different User types (i.e. a User Categorisation); and one to make the recommendations for Users in Cold Start situations (i.e. a Recommendations component).

Figure 4 shows a simplified version of the architecture for the proposed solution, and its different interactions with users.

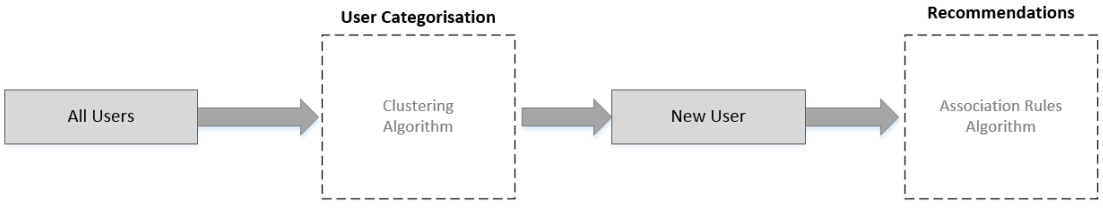


Figure 4 – Architecture of the Solution

The User Categorisation component takes all the available data relating to the users and applies a Clustering algorithm to it, so that it can find the different groups of users there are.

The Clustering algorithm used for this component is the *K-medoids* algorithm **(Kaufman & Rousseeuw, 1990)** due to its efficacy when dealing with categorical data, unlike K-means.

When the Recommendations component is trying to recommend items to a user in a Cold Start situation, it finds the users most similar to that user and all of their ratings. It then uses an Association Rules algorithm to find the associations between all the rated items and recommends the top *n* movies returned by the algorithm.

The Association Rules algorithm used to find new items to be able to be recommended is the *Apriori* algorithm **(Agrawal et al., 1993)**.

Figure 5 shows a representation of how the solution would work when applying clustering to find the users' groups (i.e. clusters).

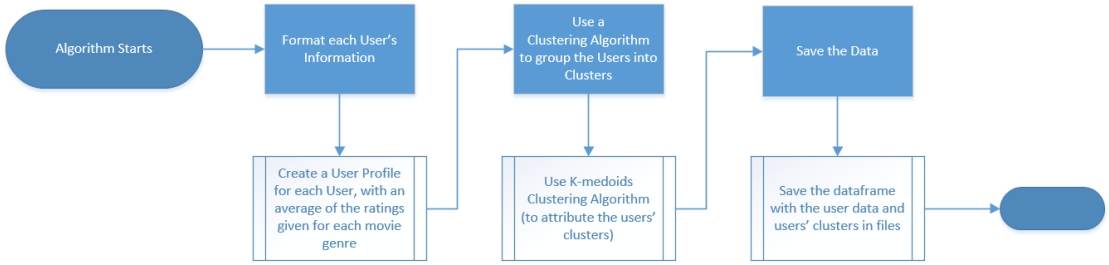


Figure 5 – The Solution's Flow for Finding the Users' Clusters

After the process described in Figure 5, the solution uses the association rules to find the best movie recommendations to a given-to user in a Cold Start situation, as shown by Figure 6.

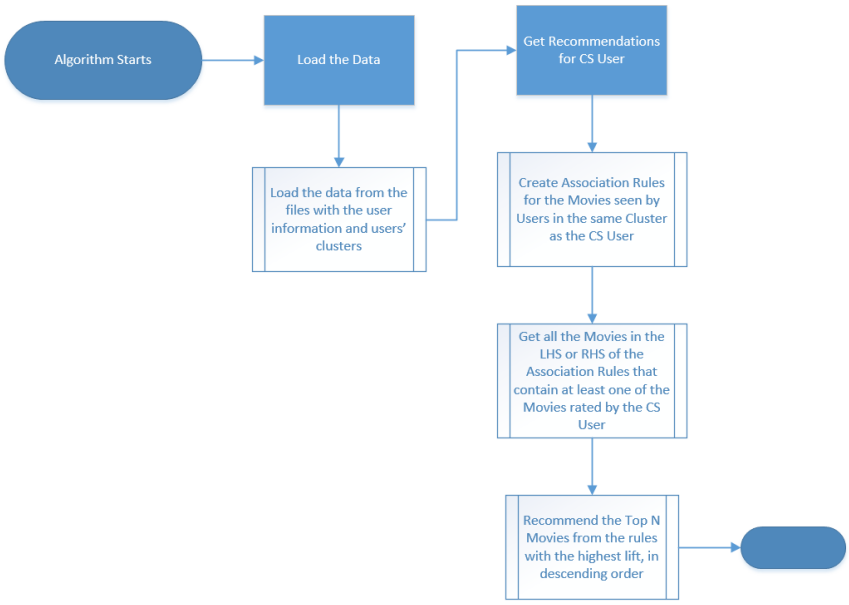


Figure 6 – The Solution's Flow for Making Recommendations to a Cold Start User

## 4.2 Programming Language

The programming language chosen to develop this solution was R, rather than Python, because of its ease of use and simplicity for creating graphics and the author's personal experience with R.

Another reason for choosing R over Python was due to the fact that neither is commonly considered better than the other, as there is still much debate regarding which of the two languages (i.e. R or Python) is better for data mining.

## 4.3 Data

The data that is to be used to train and test the solution is from the "MovieLens" website, a website for movie rating and recommendation, and is provided by GroupLens, that own a repository with various-sized datasets of "MovieLens"[1].

"MovieLens" has several datasets, each with a different number of ratings, movies and users. For the training and testing of the solution, only the "MovieLens 100K" will be used. A bigger dataset was not used due to the high processing power needed.

[1] https://grouplens.org/datasets/movielens/

As its name refers, “MovieLens 100K” is a dataset with 100,000 ratings. It also has 1,682 movies and 943 users.

The “MovieLens” dataset was chosen because of its common use in several papers about recommendation systems.

As previously stated, the data used to train and test the solution presented in this report is the “MovieLens 100K” dataset and in the following sections some graphics will be presented to show “MovieLens 100K” data distribution.

### 4.3.1 User Data

The data shown in this section pertains to the users in the system and its distribution is made according to the users’ age, gender and occupation.

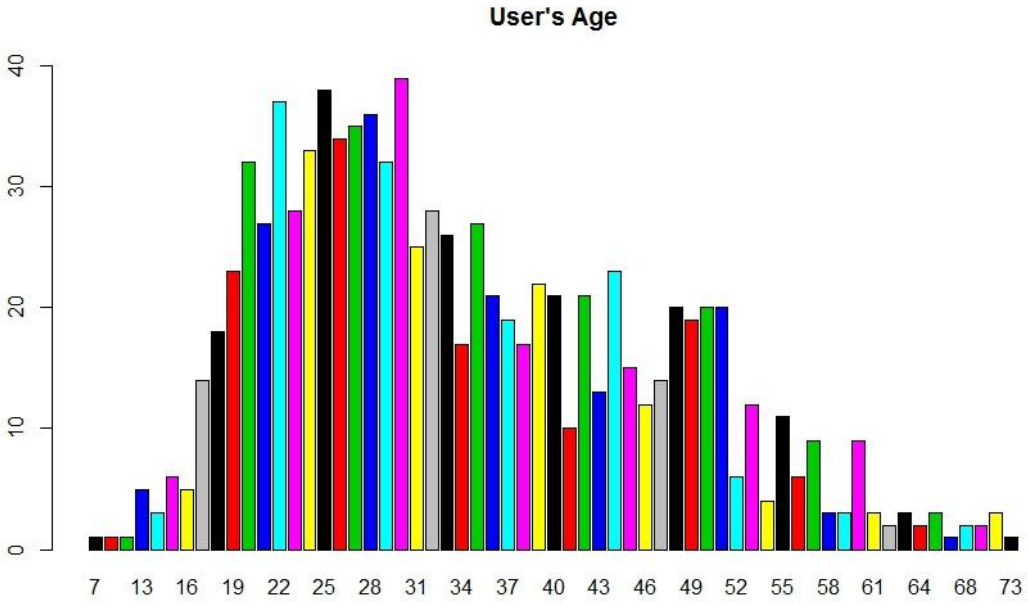


Figure 7 – Users’ Distribution by Age

Figure 7 shows the distribution of all users by their Age. The ages with most users are between twenty (20) and thirty (30). There are thirty-nine users that are 30 years old, followed closely by 38 twenty-five year old users and 37 twenty-two year old users. In a way to help in detecting the patterns in the user data, the users’ age was converted into age groups: Child, Teenager, Young Adult, Adult and Elderly.

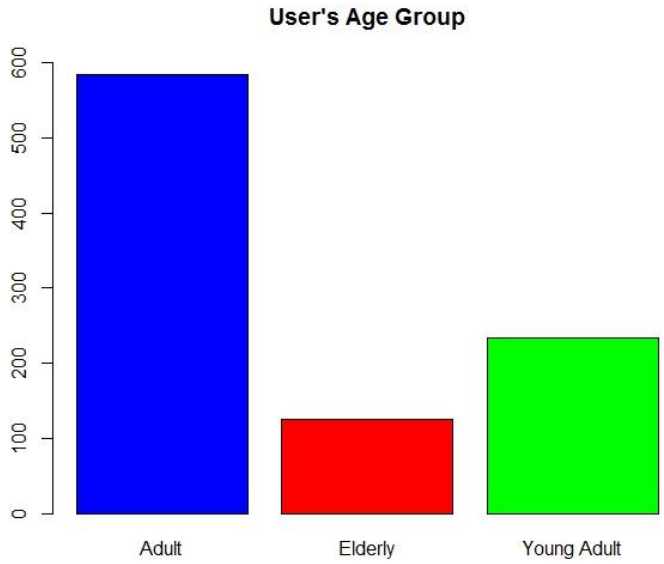


Figure 8 – Users' Distribution by Age Group

In Figure 8 is the distribution of the Users' data regarding Age Group. Around 61.90% of the users are adults, while 24.81% are young adults, followed by 13.26% that are elderly.

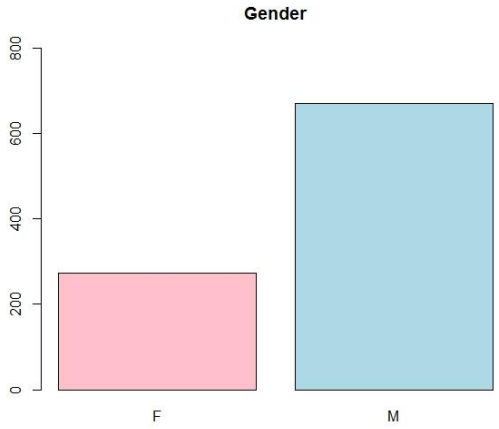


Figure 9 – Users' Distribution by Gender

Figure 9 represents the distribution of the users by gender, which shows that more than half the users are male. According to the data distribution, 71% of the users are male, while 29% are female.

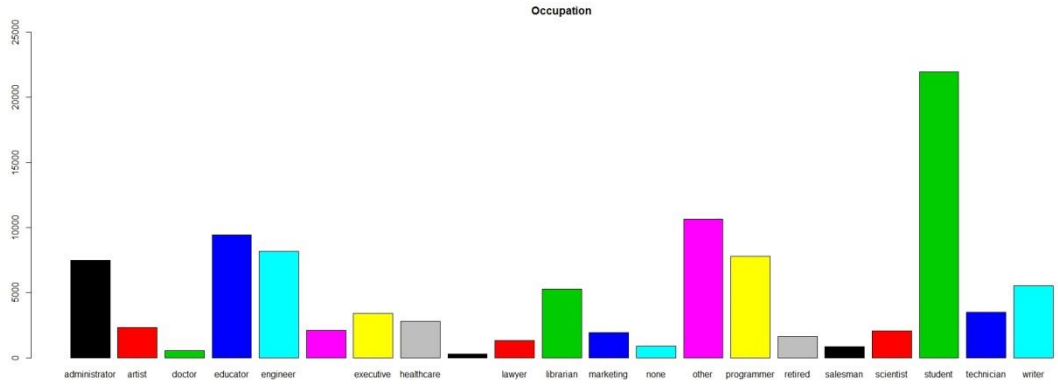


Figure 10 – Users' Distribution by Occupation

Figure 10 shows the Users' data distribution by user's occupation. In this dataset around 20% of the users are students, followed by 11% that have an occupation not listed in these options, 10% are educators, 8% are administrators, 7% are engineers and 7% are programmers, while the rest each are 5% or lower.

### 4.3.2 Movie Data

In this section, the data shown comes from the movies' information, such as its genres and release year.

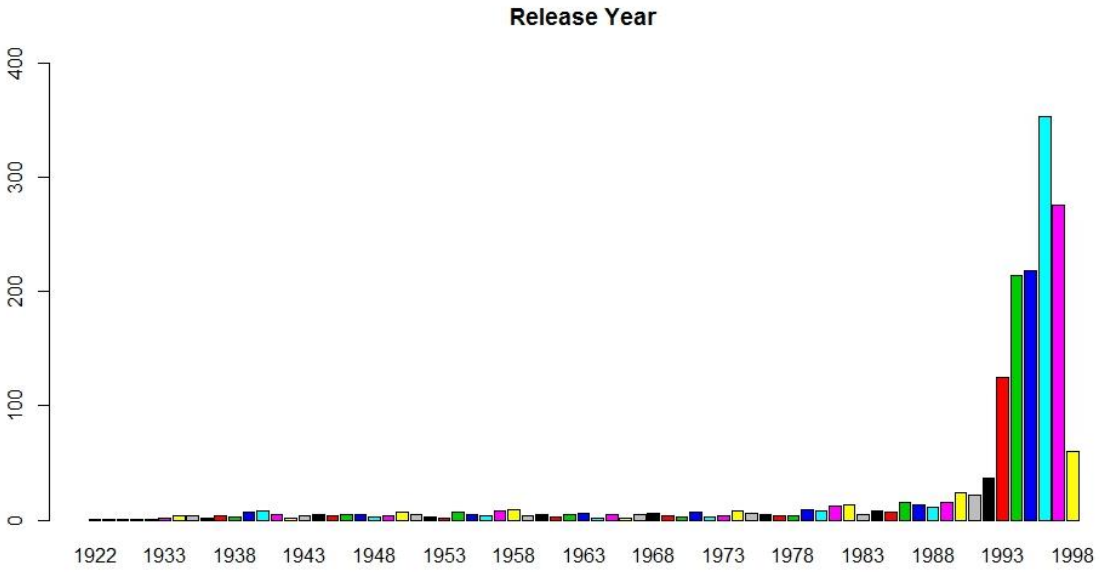


Figure 11 – Movies' Distribution by Release Year

Figure 11 represents the movie distribution according to release year, starting in 1922 and ending in 1996. It shows that there are a lot more movies from 1993 onwards. From 1922 to 1989, each year has less than 1% of all movies, while, from 1990 to 1992, there are between 1-2% of all movies. From 1993 onwards the percentage of movies increases to 7% in 1993, going as high as 21% in 1996 and then dropping to 3.6% in 1998.

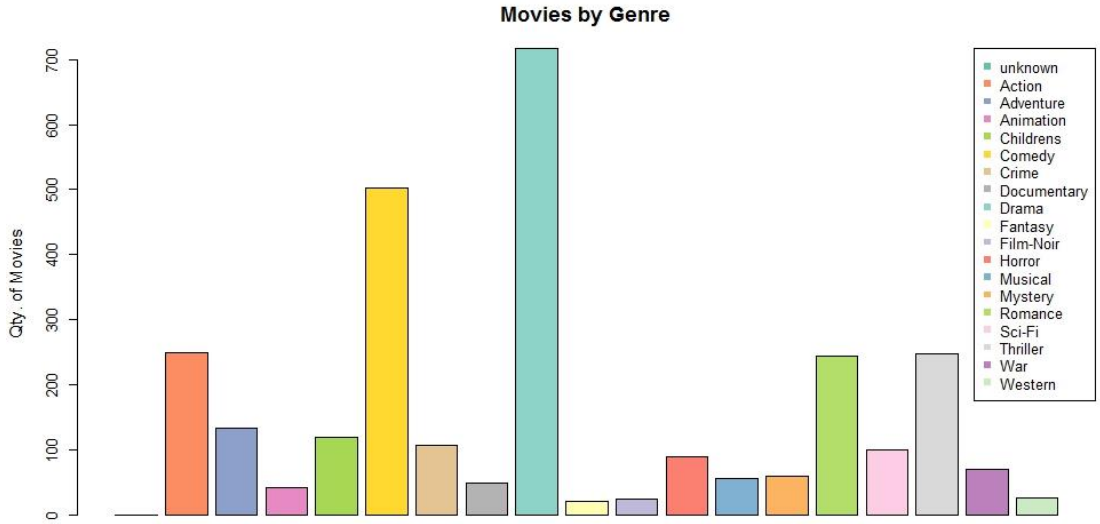


Figure 12– Movies' Distribution by Genre

Figure 12 shows the movies' distribution by genre. There are nineteen (19) distinct genres. The genre most common in movies is Drama, being in above 700 movies, followed by Comedy, in 500 movies. Next come Action, Thriller and Romance, in around 250 movies each. Then come Adventure and Children's in 133 and 120, after which all remaining genres are in around 100 or less movies.

### 4.3.3 Rating Data

In this section the data regarding users' ratings is shown and compared with the user and movie information presented in the previous sections.

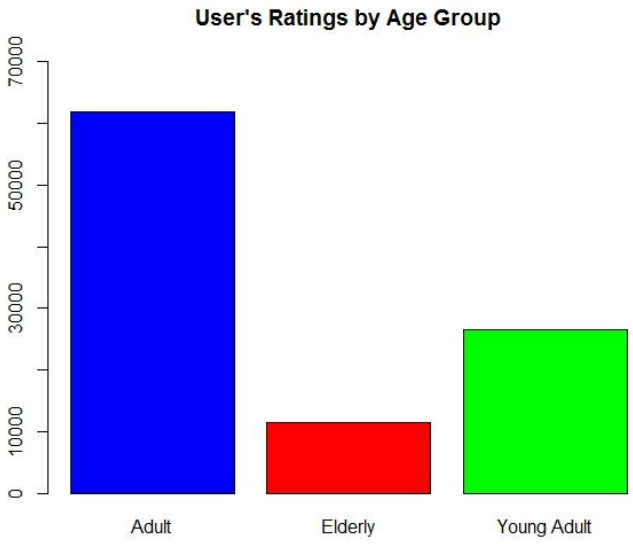


Figure 13 – User Ratings' Distribution by Age Group

When compared with Figure 8, Figure 13 is very similar, with 61.94% of the ratings having been made by adult users, 26.54% by young adults and 11.52% by elderly users.

The biggest difference between the users and the user ratings is that young adult users seem to be more active than elderly users. They represent 24.81% of all system users but are responsible for 26.54% of all ratings, while elderly users represent 13.26% but only 11.52% of ratings were made by them. The adult users' values don't differ much, with a value of 61.94%.

Similarly to the users' age group, the users' gender doesn't differ much from the users' demographic information, with 26% of ratings having been made by female users and 74% by male users.

This however means that male users are slightly more active in rating movies than female users, having a percentage increase from 71% to 74%.

| Occupations | Percentage (%) | |
|---|---|---|
| | Rating Information | Change from User Distribution |
| **Administrator** | 7.48 | -0.90 |
| **Artist** | 2.31 | -0.66 |
| **Doctor** | 0.54 | -0.20 |
| **Educator** | 9.44 | -0.63 |
| **Engineer** | 8.18 | +1.08 |
| **Entertainment** | 2.10 | +0.19 |
| **Executive** | 3.40 | +0.01 |
| **Healthcare** | 2.81 | +1.11 |
| **Homemaker** | 0.30 | -0.44 |
| **Lawyer** | 1.35 | +0.08 |
| **Librarian** | 5.27 | -0.14 |
| **Marketing** | 1.95 | -0.81 |
| **None** | 0.90 | -0.05 |
| **Other** | 10.65 | -0.48 |
| **Programmer** | 7.80 | +0.8 |
| **Retired** | 1.61 | +0.13 |
| **Salesman** | 0.86 | -0.41 |
| **Scientist** | 2.06 | -1.23 |
| **Student** | 21.96 | +1.18 |
| **Technician** | 3.51 | +0.65 |
| **Writer** | 5.54 | +0.77 |

Table 14 – Comparison between Users' and User Ratings' Distributions by User Occupation

Table 14 shows the user ratings by occupation and its difference with the system users' occupations.

As previously stated, the comparison of the occupation percentages between the user ratings and the system users allows to see which type of users are more active.

The occupation with more active users is that of Students, that has an increase in its' percentage of 1.18 percentile points, to 21.96%; followed by Healthcare workers, with an increase of 1.11 points; and Engineers with an increase of 1.08 points. The less active users' occupations are Scientists, with a decrease of 1.23 percentile points, to 2.06%; followed by Administrators, with a decrease of 0.9 points; and Marketing workers, with a decrease of 0.81 points.

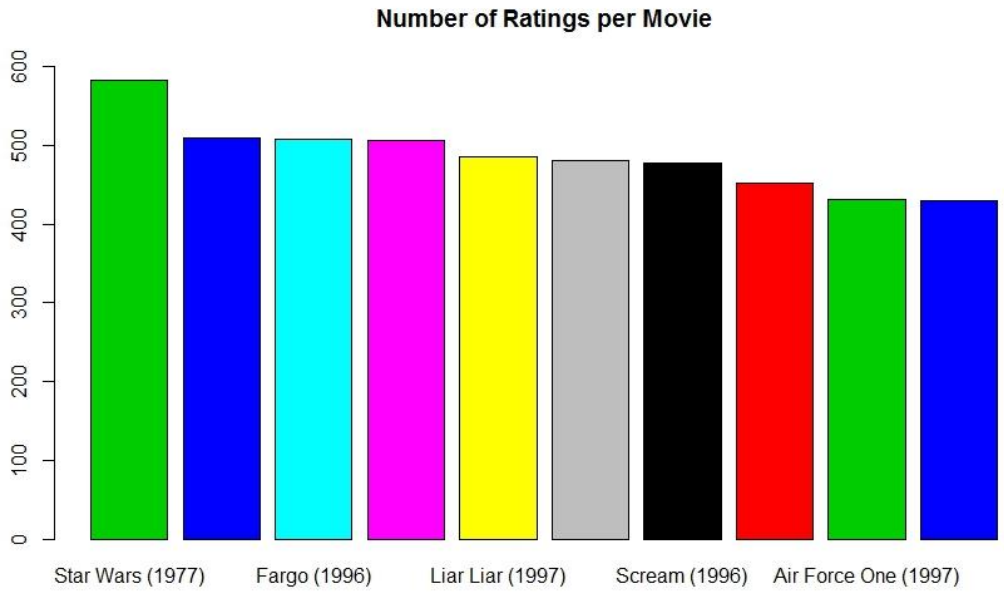


Figure 14 – Top 10 Movies with the Most User Ratings

Figure 14 shows the amount of ratings for the 10 most rated movies which are as follows: “Star Wars (1977)”, with 583 ratings; “Contact (1997)”, with 509 ratings; “Fargo (1996)”, with 508 ratings; “Return of the Jedi (1983)”, with 507 ratings; “Liar Liar (1997)”, with 485 ratings; “English Patient, The (1996)”, with 481 ratings; “Scream (1996)”, with 478 ratings; “Toy Story (1995)”, with 452 ratings; “Air Force One (1997)”, with 431 ratings; and “Independence Day (ID4) (1996)”, with 429 ratings.

There are 134 movies with only one rating. Some of those movies are: “All Things Fair (1996)”, “Angel on My Shoulder (1946)”, “Angela (1995)”, “Big Bang Theory, The (1994)” and “The Courtyard (1995)”, amongst others.

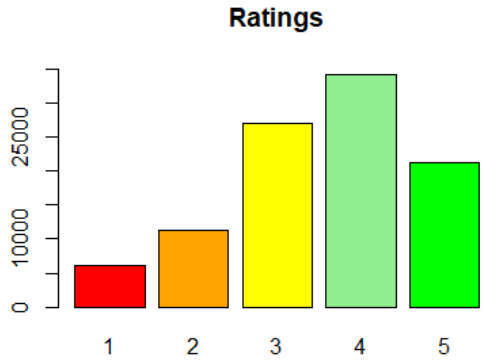


Figure 15 – User Ratings’ Distribution by Rating Value

Figure 15 shows the user ratings' distribution according to the rating value, with 1 being the worst evaluation and 5 the best.

The two lowest tiers of evaluation have the lowest percentage of ratings: around 6% of the ratings have a value of 1, while around 11% have a value of 2.

Ratings with a value of 4 are the most common, with around 34% of the ratings, followed by those with a value of 3, that represent 27% of the ratings.

And the highest tier of evaluation (i.e. a rating value of 5) represents around 21% of all ratings.

# 5 Implementation of the Solution

This chapter introduces the final solution as well as the various approaches used in the process of finding a hybrid algorithm capable of dealing with the Cold Start problem.

It is important to reiterate that the algorithm is split according to its two functions, clustering and recommending. This means that, the solution and the other approaches presented in this chapter will be divided in the same manner.

The nomenclature identifying the approaches shows the differences between each of them and its meanings are shown in Table 15.

| Nomenclature | Characteristics |
|---|---|
| Clustering AR-RG | - Uses clustering to filter the user data when creating the association rules (AR) model<br>- The user data attributes chosen for the clustering process include user demographics and the users' average rating per movie genre |
| Simple AR | - Uses all users' data when creating the association rules (AR) model |
| Eval. By Movie Title | - Comparison of the recommendations with the test set uses the movies' titles |
| Eval. By Movie Genre | - Comparison of the recommendations with the test set uses the movies' genres |

Table 15 – Characteristics of the Approaches

## 5.1 Formatting the Data

This section presents the process of formatting the data, as it is the same for all approaches.

The data from the “MovieLens 100K” dataset must be loaded and formatted, so that the recommender can find relevant patterns in it and extract useful information.

Several functions are used for that purpose:

- **getMovieLens100K_RatingData:** which loads the data pertaining to user ratings, namely the User ID, the Movie ID, the Rating and the Timestamp of the rating;
- **getMovieLens100K_GenreData:** loads the information regarding movie genres. It has all the genres’ names and is divided into two attributes, Genre ID and Genre;
- **getMovieLens100K_MovieData:** the data loaded by this function has the Movie ID, the Movie Title, the movie’s Release Date, the Video Release Date, the IMDb Link and several attributes with a binary value, corresponding to that movie’s genres. This means that all the genres of a particular movie have their respective binary values set to one (1), while the remaining are set to zero (0). This function also receives a dataframe with the genre information given by the previous function, which is used to name the columns representing the movie genres;
- **getMovieLens100K_UserData:** this function loads all user data, such as the User ID and the user’s Age, Gender, Occupation and Zip Code;
- **getMovieLens100K:** which executes all the previous functions, merges their results, removes unnecessary attributes like the user’s Age and Zip Code, the movie’s Release Date, Video Release Date and IMDb Link and derives new attributes, like the users’ Age Group, the movie’s Release Year, the Rating Year, the Years since Release and the movie’s Novelty.

An example of a row from the dataset returned by ‘getMovieLens100K’ is represented in Figure 16.

```
> user_ratings.data[1,]
  Movie ID User ID Rating Age Gender Occupation   Age Group      Movie Title unknown Action Adventure Animation Childrens Comedy Crime Documentary Drama
1        1       1      5  24      M technician Young Adult Toy Story (1995)       0      0         0         1         1      1     0           0     0
  Fantasy Film-Noir Horror Musical Mystery Romance Sci-Fi Thriller War Western Release Year Rating Year Years since Release Novelty Timestamp
1       0         0      0       0       0       0      0        0   0       0         1995        1997                   2  Recent 874965758
```

Figure 16 – Row from the ‘user_ratings.data’ dataframe

The function ‘getAgeGroup’ derives the Age Group of a user from its Age attribute, classifying it into one of five possible values: Child, Teenager, Young Adult, Adult and Elderly.

The creation of this new attribute was necessary because the Age attribute was numeric, meaning it would be harder to detect patterns. In situations where there is a variable with dozens of possible values, the most commonly used approach is to discretize it in order to reduce the number of distinct values for that variable.

The attributes Release Year, Rating Year, Years since Release and Novelty were not used in the approaches presented in this chapter but were kept as future work could involve using them in the clustering and recommendation processes.

## 5.2 Clustering AR-RG, evaluation by Movie Genre

This approach is the proposed solution for the project documented in this report, which will be compared with the other approaches presented in this chapter.

This is an Association Rules-based method that uses clusters found with the users' demographics and the Mean Rating per Genre, with weights. This method's evaluation compares movie genres when comparing the recommended movies with the ones in the test set.

After the data is formatted, this method uses the users' demographic data, as well as the mean of all the ratings of the user, per movie genre, to find and create the optimal number of clusters.

The function 'setUserGenrePreferences' returns the given dataframe with the data as shown in Figure 16, only replacing the attributes of genres that have a value of 1 with the value of the respective rating. After removing the Rating attribute, the dataframe is then used to calculate the mean rating per genre for each user, with the function 'getUserAverageByGenre', that calculates the mean of all genre columns for each user, which is used to create the dataframe in Figure 17. This is the dataframe used to find the different users' clusters.

```
> user_ratings.with_genre_mean[1,]
  User ID Gender Occupation   Age Group unknown    Action Adventure Animation Childrens   Comedy     Crime Documentary    Drama    Fantasy  Film-Noir
1       1      M technician Young Adult       0 0.9225092 0.4538745 0.1476015  0.202952 1.166052 0.3173432  0.08856089 1.549815 0.02583026 0.01845018
     Horror   Musical    Mystery   Romance    Sci-Fi  Thriller       War    Western
1 0.1660517 0.1402214 0.06642066 0.6383764 0.6346863 0.6937269 0.3394834 0.08118081
```

Figure 17 – Row from the 'user_ratings.with_genre_mean' dataframe

The next step in implementing the clustering algorithm is the mapping of the data which leads to the creation of a dissimilarities' matrix.

When creating this matrix, the users' demographic data gets a greater weight (i.e. a weight of 2) than the remaining attributes, which get a weight of 1. The weight for all demographic attributes is higher to show their importance in the definition of the users' clusters.

The distance metric used when calculating the dissimilarities matrix was the Gower distance, mainly because the data used has mixed attributes, which made the use of other metrics impractical.

The 'NbClust' R package **(Charrad et al., 2014)** was used for calculating the number of clusters for the model. It applies thirty (30) different indices for determining the number of clusters and proposes the best clustering scheme from the different results obtained in order to find the ideal number of clusters for the clustering process.

In this case, for the data with the weights, it decided that the ideal number of clusters was six (6).

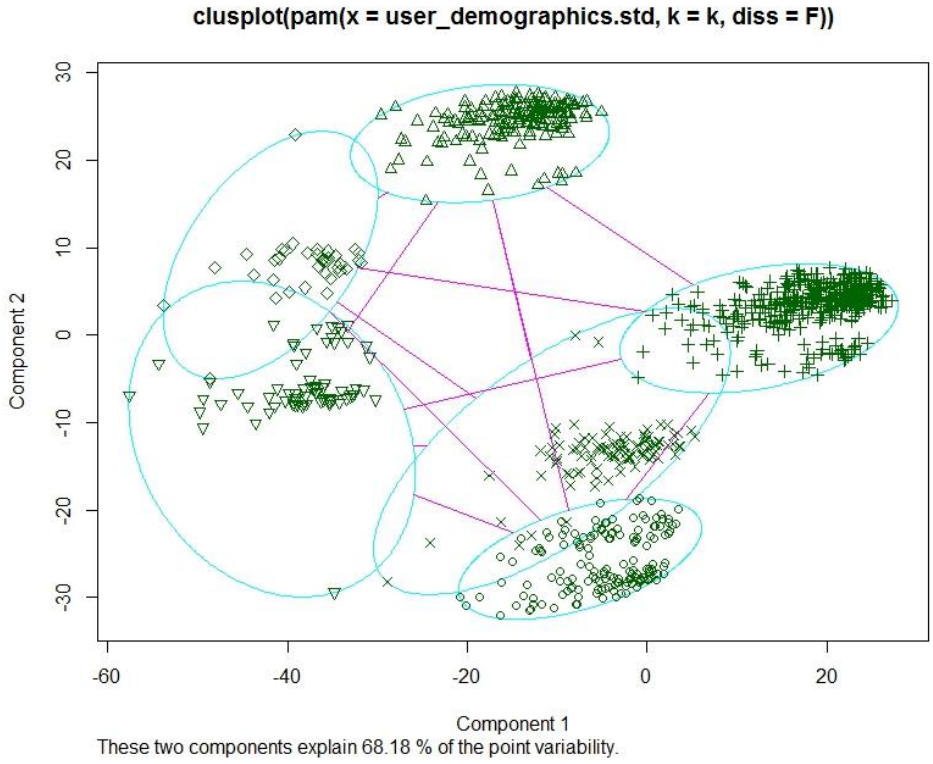


Figure 18 – Clusters of the Proposed Solution with Weights, for k=6

Figure 18 shows the graphical representation of the solution with weights' six clusters.

After the clustering process is complete, a dataframe with users' demographic information and respective cluster is created, resulting in the dataframe from Figure 19.

```
> user_demographics.with_cluster[1,]
  User ID Gender Occupation   Age Group Cluster
1       1      M technician Young Adult       1
```

Figure 19 – Row from the 'user_demographics.with_cluster' dataframe

The users' distribution into the different clusters is shown in Figure 20.

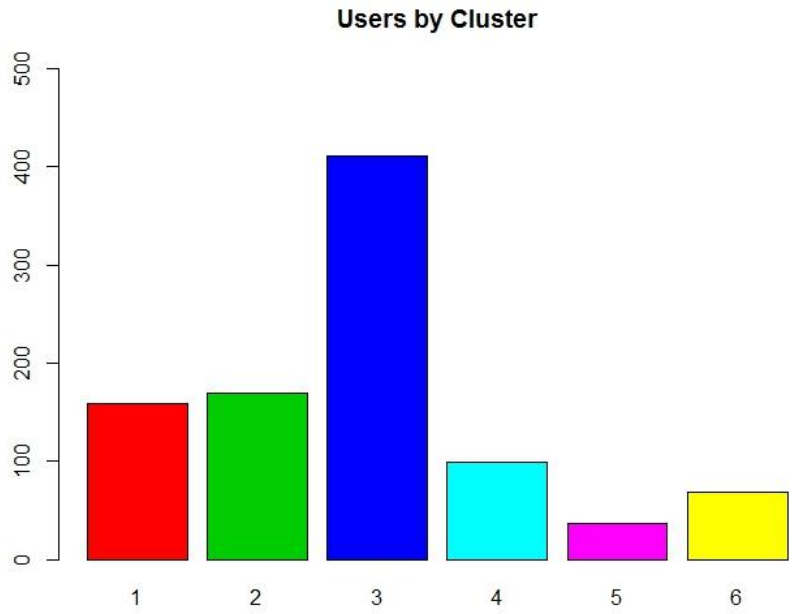


Figure 20 – Users' Distribution by Cluster

The cluster with the most users is Cluster 3 with 411 users, followed by Cluster 2 with 170 users and Cluster 1 with 159 users. Next come Cluster 4, which has 99 users, Cluster 6 with 68 users and finally Cluster 5 with 36 users.

Afterwards, the process of creating and training an association rules' model begins and it starts with the creation of one dataframe for each different cluster of users, which will be used to reduce processing needs of the following process.

As the project documented in this report aims to create an algorithm capable of dealing with the Cold Start problem, the next step is identifying the CS users in the dataset.

In most of the analysed papers, the CS users are those with 20 or less movie ratings, so we will follow that same assumption in this testing phase. In the dataset there are 32 users with 20 or less movie ratings, and their user id's are stored in the vector 'all_cs_users'.

The function ‘evaluate.AssociationRules’ is then used to train and test the model. It is coded in a way that enables its use to also train and test association rules’ models in additional situations: not taking into account the clustering information to filter the data; and evaluating the results by either comparing the movies’ titles or their genres.

This function receives a CS user’s ID, a number that defines the amount of movies recommended to be used, a variable that is either “clustering” – to use the clustering information to filter the data – or “simple – that uses all the data – and another variable that defines the comparison made when creating the model’s confusion matrix, that can either be “Movie Title” – that will compare the movies’ titles – or “Movie Genre” – that will compare the movies’ genres.

The function ‘evaluate.AssociationRules’ starts by getting the CS user’s demographic information as well as the cluster it belongs to. As this solution uses clustering to filter its data, it then checks the CS user’s cluster and gets the demographic information of all other users with the same cluster. For that, it uses the function ‘getTopMovies.byCluster’ that returns all the movies rated by those same users, with the average rating for each of those movies, only taking into account the movie ratings made by the given users . This approach only uses the movies, so the average rating values are not saved.

With the list of every movie rated by the users of the same cluster as the CS user in question, the algorithm proceeds to get every rating of those same users and creating the dataframe ‘cs_user.movie_ratings’ with all the ratings of those same users. It is from this dataframe that the association rules are derived.

Table 16 shows the variation in the support values for baskets of transactions of each cluster.

| Cluster | Min. | 1st Qu. | Median | Mean | 3rd Qu. | Max. |
|---|---|---|---|---|---|---|
| **1** | 0.006329 | 0.012660 | 0.044300 | 0.079670 | 0.113900 | 0.575900 |
| **2** | 0.005917 | 0.011830 | 0.041420 | 0.079300 | 0.106500 | 0.656800 |
| **3** | 0.002439 | 0.009756 | 0.039020 | 0.072250 | 0.100000 | 0.646300 |
| **4** | 0.010200 | 0.020410 | 0.051020 | 0.081940 | 0.112200 | 0.632700 |
| **5** | 0.028570 | 0.028570 | 0.057140 | 0.104900 | 0.142900 | 0.600000 |
| **6** | 0.014930 | 0.029850 | 0.059700 | 0.081880 | 0.104500 | 0.567200 |

Table 16 – Support of Baskets from each Cluster

The value chosen for the confidence threshold was that of 80% because of the importance of the rules being more consistent. When choosing the support threshold for the model several combinations where tried.

| | Cluster 1 | Cluster 2 | Cluster 3 | Cluster 4 | Cluster 5 | Cluster 6 |
|---|---|---|---|---|---|---|
| Support | 22% | 24% | 21% | 24% | 23% | 15% |
| Qty. Rules | 1,420 | 1,470 | 1,791 | 1,661 | 607 | 427 |

Table 17 – Support and Respective Number of Rules by Cluster

Table 17 shows the results of the function 'getPrunedRules', that tries different values for the support threshold while it attempts to get a number of association rules closest to the number of distinct movies in the dataset (i.e. around 1,600). It starts with a support value of 15% and then creates a ruleset using the Apriori algorithm. If the number of rules is above 15,000 it continues to the next loop because the pruning process becomes very CPU-heavy and takes a long time to complete. However, if the number of rules is below 15,000 the algorithm then proceeds to prune said rules. If the pruned ruleset has more than 1,800 rules in it, the algorithm continues to the next loop as its objective is finding a pruned ruleset as close to the number of distinct movies as possible. When it finds a pruned ruleset with less than 1,800 rules, it returns those rules.

As explained in the Methodology section, the proposed solution's model evaluation will use leave-one-out validation which was chosen because the models will be evaluated CS user by CS user and, as each CS user has 19-20 ratings, the training and testing sets will be using the five-fold cross validation method, with one fold used for model testing and the remaining four used for model training.

The five-fold cross validation is implemented next, with the the division of the data into five folds. The rest of the 'evaluate.AssociationRules' function is then run five times, each with a different fold in the testing set and the remaining four in the training set.

After the data division into training and test sets, the algorithm uses the movies in the training set in conjunction with the association rules to find the movies to recommend. This process uses the function 'get_rules' that returns all rules that have the given movie's title in their LHS or RHS.

It then extracts all the movies from those rules that are not the one given to the function, as well as the respective rules' lift, with which it creates a dataframe with each movie recommended and the lift from the rule it was extracted from.

Then, the final dataframe is sorted from highest lift value to lowest, with the highest lift for each movie being kept and duplicates with a lower lift being removed, and the movies in the top *n* rows are set as the recommended movies.

Afterwards, and as this solution compares the movies' genres, the function 'getGenreList' is used to get both the genres of the recommended movies and the ones from the test set.

The genres recommended by the association rules are then compared with the genres in the test set and the confusion matrix is created. In this phase only the most basic values are defined, such as the true positives, false positives, true negatives and false negatives.

Then a dataframe is returned with these values associated to the CS user given to the function: one row for each iteration of the five-fold cross validation.

The algorithm runs the function 'evaluate.AssociationRules' for each CS user and creates a dataframe with all the returned information, which is then used firstly to calculate the accuracy, precision, recall and f1-measure for each user and then get the average of those values to firstly find each CS user's mean values for the evaluation metrics, and then this solution's final evaluation metrics.

## 5.3 Clustering AR-RG, evaluation by Movie Title

This approach is similar to the proposed solution except for the evaluation comparison: while the proposed solution compares the movies' genres, this approach compares the movie titles when calculating the confusion matrix's metrics.

After the final dataframe being sorted, its duplicates being removed and the movies in the top n rows being set as the recommended movies, this approach compares the recommended movies' titles, rather than its genres, with the ones from the test set and the confusion matrix is then created.

The following process is the same, with the algorithm running the function 'evaluate.AssociationRules' for each CS user and then creating a dataframe with all the

returned information, that is used to calculate the evaluation metrics, first per CS user, and then the its final value, by getting the mean for each metric from all CS users.

## 5.4 Simple AR, evaluation by Movie Genre

This approach is very similar to the proposed solution but its main difference occurs in the use of clustering to filter the data when creating the association rules: unlike the proposed solution that uses clustering to filter its data, this approach does not.

While the proposed solution gets only the movies rated and user ratings made by users in the same cluster as the CS user, this approach uses all movies and all user ratings, except the CS user's, in the creation of the association rules.

Similarly to the proposed solution, the value chosen for the confidence threshold was 80% and the ideal support threshold was also found to be mainly 21% through the use of the 'getPrunedRules' function, which also returned the pruned rules used.

Afterwards this approach is the same as the proposed solution, comparing the movies' genres, creating the confusion matrix for each CS user in each iteration of the five-fold cross validation, and calculating the accuracy, precision, recall and f1-measure, first for each user's iteration, then getting a mean of each measure of each iteration for each user, and finally calculating the mean of each metric to be defined as the approach's final metric.

## 5.5 Simple AR, evaluation by Movie Title

This approach is the same as the previous approach, except in the evaluation method. Just like the difference between the proposed solution and 'Clustering AR-RG, eval. by Movie Title', their main difference lies in the fact that this approach uses the movies' titles, rather than their genres when comparing the recommended movies with the ones in the test set.

# 6 Solution Evaluation and Experiments

This chapter presents the results of the various metrics used to compare the solution with the presented alternatives, as well as the hypotheses to be evaluated, the experiment methodology and tests that were conducted.

## 6.1 Methodology

This section presents the confusion matrix and resulting metrics of the proposed solution's evaluation and its comparison with the other approaches introduced.

As stated previously, the initial confusion matrix for the CS users has five rows per CS user, each corresponding to a different iteration of the five-fold cross validation process.

The confusion matrix has columns for the standard measures: TP, TN, FP and FN; but it also has two other columns: Test Ratings, which refers to the amount of items in the test set; and Recommendations, which represents the amount of items in the recommendations.

After the creation of the confusion matrix comes the calculation of the evaluation metrics defined in the previous sections: Accuracy, Precision, Recall and F1 measure.

In a first step, these measures are calculated for each CS user's fold iteration, followed by the calculation of the mean of each measure for each CS user, resulting in the dataframe structure shown in Figure 21.

```
> head(cs_user.recommender_metrics)
  User.ID  Accuracy Precision    Recall F1.Measure
1      36 0.9969934 0.6363636 0.9333333  0.7567568
2     895 0.9981960 0.8000000 0.9411765  0.8648649
3      19 0.9975947 0.6666667 0.9230769  0.7741935
4      34 0.9963921 0.6250000 0.7692308  0.6896552
5      93 0.9969934 0.7777778 0.8235294  0.8000000
6     143 0.9969934 0.5833333 1.0000000  0.7368421
```

Figure 21 – Evaluation Metrics for the CS Users in the Proposed Solution

The solution's final metrics are then derived from these values by calculating their mean, which results in the values in Figure 22.

```
> cs_user.recommender_evaluation
                                  Method  Accuracy  Precision    Recall F1 Measure
1 Clustering AR-RG, eval. by Movie Genre 0.9960196 0.45893288 0.8919330 0.58465026
```

Figure 22 – Proposed Solution's Evaluation Metrics

The values for Accuracy should be dismissed as they are heavily influenced by the number of TN, which refers to all movies that were not recommended while also not in the test set. The amount of recommendations isn't much higher than ten, which means that there are always over 1,600 movies classified as TN.

The models' Precision is 45.89%, which means that around 46% of the recommendations made are correct. And Recall is around 89%, which means that 89% of movie recommendations will be considered by these CS users.

As Precision and Recall are the metrics used to compare the different approaches, the f1-measure can be instead used, as it represents the harmonic mean of these two metrics.

Table 18 shows the different approaches and their respective metrics making it easier to compare them.

| **Method** | **Accuracy** | **Precision** | **Recall** | **F1 Measure** |
|---|---|---|---|---|
| **Clustering AR-RG, eval. by Movie Genre** | 0.9960196 | 0.45893288 | 0.8919330 | 0.58465026 |
| **Clustering AR-RG, eval. by Movie Title** | 0.9923341 | 0.03468510 | 0.1214747 | 0.05206174 |
| **Simple AR, eval. by Movie Genre** | 0.9960914 | 0.45744781 | 0.8375979 | 0.57108556 |
| **Simple AR, eval. by Movie Title** | 0.9928483 | 0.01711111 | 0.0652381 | 0.02632555 |

Table 18 – Evaluation Metrics per Approach

Comparing the proposed solution with 'Clustering AR-RG, eval. by Movie Title' will show the difference in the model's metrics when the movies' genres are used in the recommendation process instead of the movies' titles.

As it is a broader concept, it may be a better approach for situations like these, where there isn't an abundance of available user data.

The use of movies' genres in the evaluation process shows an increase of more than thirteen-fold in Precision, seven-fold in Recall and eleven-fold in the F1 Measure.

And comparing the proposed solution with 'Simple AR, eval. by Movie Genre' shows the difference in the metrics when clustering is used to group the users into clusters and then filter the data accordingly when creating the association rules.

The use of clustering to filter the data has a marginal increase in Precision of 0.15, while it increases Recall by 5.43. And as these two metrics are both important, the best metric to compare the approaches would be the harmonic mean of both, the F1 Measure, which is increased by 1.36 when clustering is used.

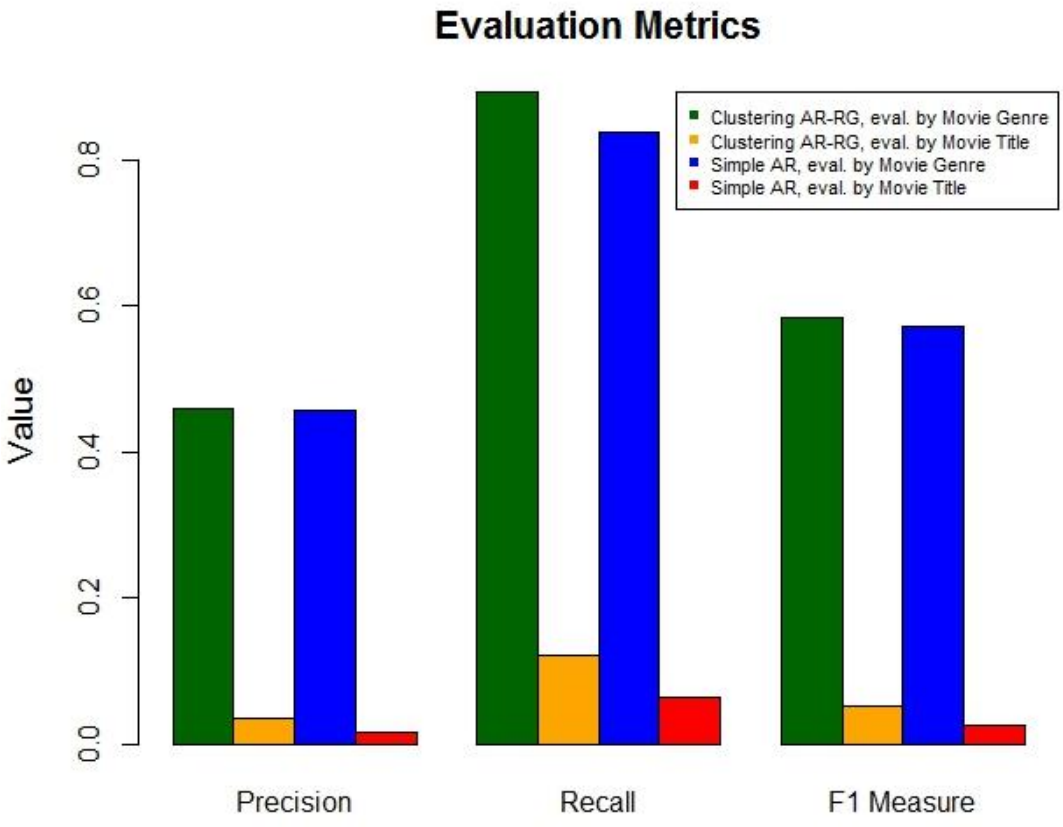


Figure 23 – Evaluation Metrics for the Alternative Approaches

## 6.2 Comparison with the Papers' Approaches

In this section a comparison of the proposed solution is made with the alternative approaches presented in this paper as well as with each of the methods presented in the papers analysed in the Related Work section. This comparison is first made through the use of statistical tests and then is made side-by-side through the use of tables.

It is important to note that some papers give only one value for each evaluation metric, making it difficult to use a statistical test to compare their approach with the proposed solution. These papers will be evaluated in the section Comparison Summary but will not be evaluated through statistical tests.

As mentioned in the previous section, the hypotheses to be tested will be that the proposed solution performs better than the presented alternatives. This means that the null hypothesis (i.e. $H_0$) for all comparisons is that both approaches are equal.

Two types of statistical tests will be used to attempt to disprove the null hypothesis: Student's t test for paired samples, to compare the proposed solution with the alternative approaches presented in this paper; and Welch two sample t test, to compare the proposed solution with the methods presented in the analysed papers.

The probability threshold (α), below which the hypothesis will be rejected, will be 5%.

Also, the comparison between the approaches is made with the 't.test' function of R, which has both types of t test, for paired and unpaired samples.

### 6.2.1 Student's t test for paired samples

#### 6.2.1.1 Clustering AR-RG, evaluation by Movie Title

When comparing the proposed solution with this approach as the only variables to be compared would be the Precision and Recall, the F1 Measure is used, as it is a harmonic mean of the two.

The degrees of freedom value for this test is 30 and the t-value is 45.314. The p-value was found to be 1. $726481e^{-29}$, which makes the probability of the difference in the data being due to sampling error lower than the probability threshold, which means the null hypothesis can be rejected in this instance and the proposed solution has better performance than "Clustering AR-RG, eval. by Movie Title".

#### 6.2.1.2 Simple AR, evaluation by Movie Genre

For the same reasons specified in the previous subsection, the F1 Measure is used for the comparison of the proposed solution with this approach.

This test's value of the degrees of freedom is 29 and the t-value is 1.3583. The p-value is 0.0924244, which is greater than the probability threshold. This means that the null hypothesis cannot be rejected in this instance and that the performance of the proposed solution and "Simple AR, eval. by Movie Genre" are likely to be equivalent.

#### 6.2.1.3 Simple AR, evaluation by Movie Title

The F1 Measure is also used in the comparison of this approach with the proposed solution for the reasons specified in the previous subsections.

The degrees of freedom for this test are also 29, the t-value is 50.387 and the p-value is $4.309288e^{-30}$, which means that the null hypothesis can be rejected and that the proposed solution's performance is better than that of "Simple AR, eval. by Movie Title".

### 6.2.2 Welch two sample t test

The papers whose methods are not being compared due to the lack of compatible evaluation metrics are **(Lika et al., 2014)**, **(Luiz et al., 2015)**, **(Aleksandrova et al., 2016)** and **(Wei et al., 2017)**.

#### 6.2.2.1 "Context-Aware Personalization Using Neighborhood-Based Context Similarity"

In this test, the proposed solution is compared with the method presented in **(Otebolaku & Andrade, 2017).**

In the test results presented by the authors of this paper, they calculated the average Precision for different numbers of recommendations. The values for the average precision used in this comparison will be those when the number of recommendations is ten (10), as the number of recommendations used in the proposed solution is also ten (10).

As the authors of this paper don't show the metrics for each user, the values for the average Precision used will be those of the mean average Precisions of each context.

This test has 4.2196 degrees of freedom, a t-value of -0.039932 and the p-value is 0.5150171, which means that the null hypothesis cannot be rejected in this instance, and that the performance of the proposed solution and the approach of **(Otebolaku & Andrade, 2017)** are likely equivalent.

#### 6.2.2.2 "Using Association Rules to Solve the Cold-Start Problem in Recommender Systems"

This test compares the proposed solution with the approach documented in **(Shaw et al., 2010)**.

As the authors of this paper use several different methods when expanding the user profiles, the values chosen for comparison were those for the ReliableBasis with HRR (RBHRR) because it showed the highest values. The evaluation metric chosen was of the F1 Measure because the other metrics used were Precision and Recall.

This test has 30.136 degrees of freedom and a t-value of 52.938. The p-value is $1.346033e^{-31}$, which means that the null hypothesis can be rejected and that the performance of the proposed solution is better than the one from the approach of **(Shaw et al., 2010)**.

### 6.2.3 Comparison Summary

In this subsection the proposed solution's results are compared side-by-side with the results presented in the various papers analysed in Related Work.

Table 19 shows the results of both the proposed solution and the method proposed by the authors of **(Otebolaku & Andrade, 2017)**. In this paper the authors use the average Precision to evaluate the use of contexts in the recommendation process and the mean average Precision to evaluate the method's performance.

| Methods / Precision Values | Proposed Solution | (Otebolaku & Andrade, 2017) |
|---|---|---|
| **Best Performance** | 0.61 | 0.68 |
| **Worst Performance** | 0.34 | 0.20 |
| **Average Performance** | 0.46 | 0.64 |

Table 19 – Comparison of the Proposed Solution with **(Otebolaku & Andrade, 2017)**

When comparing the results of both approaches, it is important to note that the values of the proposed solution come from the CS users and their individual values, while the values

from **(Otebolaku & Andrade, 2017)** come from five different contexts, with five neighbourhood sizes each. While the proposed solution shows higher values for the worst performance, its best and average performances are lower than the approach from **(Otebolaku & Andrade, 2017)**. Although the approach's best performance is only slightly better than the proposed solution, the difference in the average performance is much higher.

This is most likely due to the fact that the proposed solution uses leave-one-out validation, which leads to a higher variation. It may also be because of the higher number of users compared with context values for neighbourhood sizes.

There is not enough information to safely assume that one approach is better than the other, even though the approach in **(Otebolaku & Andrade, 2017)** has better results. Other metrics such as the Recall and the F1 Measure should be used to better compare the two methods. As seen in the previous section, the statistical test results also show that the performance of both approaches is likely equivalent.

| | Precision | Recall | F1 Measure |
|---|---|---|---|
| **Proposed Solution** | 0.46 | 0.89 | 0.584650 |
| **(Yanxiang et al., 2013)** | 0.61 | - | - |
| **(Shaw et al., 2010)** | 0.00819 | 0.0754 | 0.0148 |
| **(Bobadilla et al., 2011)** | 0.46 | 0.84 | 0.594462 |

Table 20 – Comparison of the Proposed Solution with the Remaining Approaches

The proposed solution's Precision is 46%, which is lower than the approach in **(Yanxiang et al., 2013)** which has a Precision of 61%. However the authors of **(Yanxiang et al., 2013)** don't give much information other than this value for the Precision, making it difficult to compare further with the proposed approach. As previously stated, the Recall and the F1 Measure should be used to better compare these methods.

The approach in **(Shaw et al., 2010)** has significantly lower values for Precision, Recall and the F1 Measure than the proposed solution. This is likely due to the fact that the authors are attempting to solve the CS problem by increasing the amount of association rules found for a CS user with few ratings, rather than increase the model's performance.

The comparison of the proposed solution with the approach in **(Bobadilla et al., 2011)** is made through the F1 Measure as both approaches have the values for Precision and Recall.

It is also important to note that the users are considered CS users in the same situation for both approaches (i.e. those that have twenty or less ratings available) and that the approach in **(Bobadilla et al., 2011)** uses a bigger version of the MovieLens dataset, the "MovieLens 1M". The use of a larger dataset gives the paper's model a better chance for finding patterns in the data and get overall better results.

As Table 20 shows, **(Bobadilla et al., 2011)** has the same Precision as the proposed solution, but a lower Recall. The F1 Measure is slightly higher for **(Bobadilla et al., 2011)** which means it is slightly better than the proposed solution.

# 7 Conclusions

This chapter contains the conclusions of this document, followed by the future work section where some possible improvements are presented.

## 7.1 Conclusions

The User Cold Start situation is a serious problem in recommenders which can lead to the system's loss of new users. This increases the importance of recommenders capable of dealing with this problem.

The algorithm documented in this report aims to deal with this situation and its development required the implementation of two components: one for clustering, that assigned users to the groups that best characterised them; and one for association rules, that used the users in the same cluster as the target for the recommendations, looking for the best movies to recommend.

The implementation of the clustering component used the users' demographic data and the mean rating of each user for all movies with a specific movie genre. When creating the clusters an increased weight was given to the demographic attributes due to their importance when characterising a user.

The component that creates the association rules' model filters the data used for its creation according to the cluster of the user to which the system is trying to make recommendations.

However, the filtering process increases the runtime of the algorithm, which was found when comparing the algorithm's runtime with the runtime of its variant that does not use clustering.

Another limitation of the algorithm comes from the method that chooses the support threshold and returns the pruned association rules, which is heavy on the processing power and has a considerable runtime.

The algorithm documented in this report shows that clustering has marginal improvements when used to filter the information for the creation of association rules.

The solution shows a marginally higher Precision but it has a higher Recall value, which means it was able to make more interesting recommendations to the users. It also shows a slightly higher F1 Measure value, which implies a slightly better overall performance.

Although this algorithm does not have overall better results than the methods in the papers analysed in this report, it is a simpler approach to most of them. It is also able to get better results in some of the evaluation metrics, such as Recall.

## 7.2 Future Work

Further improvements on the solution documented in this report could include a method that makes better recommendations when given a set of movie genres, possibly taking into account the movie's popularity or novelty.

Other improvements for this solution could focus on its optimisation in order to reduce the processing time and power required to run the algorithm.

# References


**(Wirth and Hipp, 2000)** Wirth, R., and Hipp, J. CRISP-DM: Towards a standard process model for data mining. In Proceedings of the 4th International Conference on the Practical Applications of Knowledge Discovery and Data Mining, 2000, pp. 29–39.

**(Lika et al., 2014)** Lika, B., Kolomvatsos, K., & Hadjiefthymiades, S. (2014). Expert Systems with Applications Facing the cold start problem in recommender systems. Expert Systems With Applications, 41(4), 2065–2073. https://doi.org/10.1016/j.eswa.2013.09.005

**(Park et al., 2012)** Park, D. H., Kim, H. K., Choi, I. Y., & Kim, J. K. (2012). Expert Systems with Applications A literature review and classification of recommender systems research. Expert Systems With Applications, 39(11), 10059–10072. https://doi.org/10.1016/j.eswa.2012.02.038

**(Shambour & Lu, 2015)** Shambour, Q., & Lu, J. (2015). Journal of Computer and System Sciences An effective recommender system by unifying user and item trust information for B2B applications. Journal of Computer and System Sciences, 81(7), 1110–1126. https://doi.org/10.1016/j.jcss.2014.12.029

**(Wu et al., 2015)** Wu, D., Zhang, G., & Lu, J. (2015). A fuzzy preference tree-based recommender system for personalized business-to-business e-services. IEEE Transactions on Fuzzy Systems, 23(1), 29–43. https://doi.org/10.1109/TFUZZ.2014.2315655

**(Gorakala & Usuelli, 2015)** Gorakala, S. & Usuelli, M. (2015). Building a recommendation system with R (1st ed.). Birmingham: Packt Publishing Ltd.

**(Gower, 1971)** Gower, J.C. General Coefficient of Similarity and Some of Its Properties, Biometrics 27, pp857-874 1971

**(Hartigan, 1975)** Hartigan, John A Clustering Algorithms, Wiley New York, NY 1975

**(Jain, 2010)** Jain A. K. Data clustering: 50 years beyond K-means. Pattern recognition letters, 31(8), 651-666, 2010.

**(Vapnik, 1998)** V. Vapnik. Statistical Learning Theory. Springer, N.Y., 1998.

**(Quinlan, 1986)** Quinlan, J. R. (1986). Induction of decision trees. Machine learning, 1(1), 81-106.

**(Agrawal et al., 1993)** Agrawal R., Imielinski T., Swami A. N. Mining association rules between sets of items in large databases. In Proceedings of the 1993 ACM SIGMOD International Conference on Management of Data, 207-216, 1993.

**(Dudani, 1976)** Dudani, S. A. (1976). The distance-weighted k-nearest-neighbor rule. IEEE Transactions on Systems, Man, and Cybernetics, (4), 325-327.

**(Mitchell, 1997)** Mitchell, T. M. (1997). Artificial neural networks. Machine learning, 45, 81-127.

**(James et al., 2013)** James, G., Witten, D., Hastie, T., & Tibshirani, R. (2013). An introduction to statistical learning (Vol. 112). New York: springer.

**(Ding et al., 2003)** Ding, C., He, X., Husbands, P., Zha, H., & Simon, H. (2003). PageRank, HITS and a unified framework for link analysis. In Proceedings of the 2003 SIAM International Conference on Data Mining, pp. 249-253.

**(Breiman, 1996)** Breiman, L. (1996). Bagging predictors. Machine learning, 24(2), 123-140.

**(Breiman, 2001)** Breiman, L. (2001). Random forests. Machine learning, 45(1), 5-32.

**(Freund et al., 1999)** Freund, Y., Schapire, R., & Abe, N. (1999). A short introduction to boosting. Journal-Japanese Society For Artificial Intelligence, 14(771-780), 1612.

**(Lu et al., 2015)** Lu, J., Wu, D., Mao, M., Wang, W., & Zhang, G. (2015). Recommender system application developments : A survey. Decision Support Systems, 74, 12–32. https://doi.org/10.1016/j.dss.2015.03.008

**(Adomavicius & Tuzhilin,** Adomavicius, G., & Tuzhilin, A. (2005). Toward the Next Generation of

**2005)** Recommender Systems : A Survey of the State-of-the-Art and Possible Extensions, 17(6), 734–749.

**(Luiz et al., 2015)** Luiz, A., Pereira, V., & Raul, E. (2015). Knowledge-Based Systems Simultaneous co-clustering and learning to address the cold start problem in recommender systems. Knowledge-Based Systems, 82, 11–19. https://doi.org/10.1016/j.knosys.2015.02.016

**(Aleksandrova et al., 2016)** Aleksandrova, M., Brun, A., Boyer, A., & Chertov, O. (2016). Identifying representative users in matrix factorization-based recommender systems : application to solving the content-less new item cold-start problem. Journal of Intelligent Information Systems. https://doi.org/10.1007/s10844-016-0418-3

**(Wei et al., 2017)** Wei, J., He, J., Chen, K., Zhou, Y., & Tang, Z. (2017). Collaborative filtering and deep learning based recommendation system for cold start items, 69, 29–39. https://doi.org/10.1016/j.eswa.2016.09.040

**(Otebolaku & Andrade, 2017)** Otebolaku, A. M., & Andrade, M. T. (2017). Context-Aware Personalization Using Neighborhood-Based Context Similarity. Wireless Personal Communications, 94(3), 1595–1618. https://doi.org/10.1007/s11277-016-3701-2

**(Yanxiang et al., 2013)** Yanxiang, L., Deke, G., Fei, C., & Honghui, C. (2013). User-based Clustering with Top-N Recommendation on Cold-Start Problem. https://doi.org/10.1109/ISDEA.2012.381

**(Shaw et al., 2010)** Shaw, Gavin & Xu, Yue & Geva, Shlomo. (2010). Using Association Rules to Solve the Cold-Start Problem in Recommender Systems. 340-347. 10.1007/978-3-642-13657-3_37.

**(Bobadilla et al., 2011)** Bobadilla, Jesús & Ortega, Fernando & Hernando, Antonio & Bernal, Jesús. (2011). A collaborative filtering approach to mitigate the new user cold start problem. Knowledge Based Systems - KBS. 26. . 10.1016/j.knosys.2011.07.021.

**(Kaufman & Rousseeuw, 1990)** Kaufman, L., & Rousseeuw, P. J. (1990). Finding groups in data: An introduction to cluster analysis. New York: Wiley.

**(Gupta & Goel, 2016)** Gupta, S. & Goel, S. (2016). Handling User Cold Start Problem In Recommender Systems Using Fuzzy Clustering, (June).

**(Gupta et al., 2013)** Pankaj Gupta, Ashish Goel, Jimmy Lin, Aneesh Sharma, Dong Wang, and Reza Bosagh Zadeh. WTF: The who-to-follow system at Twitter, Proceedings of the 22nd international conference on World Wide Web.

**(Charrad et al., 2014)** Charrad M, Ghazzali N, Boiteau V, Niknafs A. Nbclust: An R Package for Determining the Relevant Number of Clusters in a Data Set. J Stat Softw. 2014; 61:1–36.

**(1)** [Support Vectors]. (n.d.). Retrieved February 23, 2017, from http://www.saedsayad.com/support_vector_machine.htm