Vaal University of Technology

Recommender System for Mobile Subscriber Provisioning

Thesis

Submitted for the degree Magister Technologiae in Information Technology

In the Department of Information and Communications Technology,

Faculty of Applied and Computer Sciences

Elias Mbongeni Sibanda

Student Number: 210046910

Supervisor: Professor Tranos Zuva

Co-supervisor: Mr. Varghese Thomas

April 2018

DECLARATION BY CANDIDATE

"I hereby declare that the dissertation/thesis submitted for the degree M-Tech: Information Technology, at Vaal University of Technology, is my original work and has not previously been submitted to any other institution of higher education. I further declare that all sources cited or quoted are indicated and acknowledged by means of a comprehensive list of references"

Elias Mbongeni Sibanda

ACKNOWLEDGEMENTS

Let me first thank my supervisor Prof T. Zuva for introducing me and guiding me through this field of research. Without you I would never have had the love for recommender systems. Your contribution has made a meaningful change in my life and I will forever be indebted to you. Secondly, let me thank my girlfriend Lerato Moeletsi and daughter Neo Annie Moeletsi for sacrificing our time together and understanding when I had to go and conduct experiments. Thank you very much.

I also thank my colleagues who encouraged me to work hard and the fellow students who provided guidelines when I was struggling with certain tools and concepts. Thank you, guys.

I thank God and my family that pushed me to complete this dissertation even on days when I felt demotivated. A big thank you to the Vaal University of Technology for creating a conducive environment for us to learn, as well as National Research Foundation for providing the funds and making this possible. I will forever be grateful for the opportunity.

Lastly, I thank my late father (Jacob Sibanda) for always encouraging me to study and to complete things once I have started them. It is these words that kept me going during difficulties.

CONTENTS

LIST OF FIGURES

ABSTRACT

Mobile phone recommendation systems are of great importance for mobile operators to achieve a profit. In a user-derived market, the number of contract users and contract phones is especially significant for mobile service operators. The tremendous growth in the number of available mobile cellular telephone contracts necessitates the need for a recommender system to help users discover suitable contracts on the basis of their usage patterns. Recommender systems recommend items to users and their primary purpose is to increase sales and recommend items that are predicted to be suitable for individual users. There are two commonly used techniques in developing recommender systems including collaborative- and content-based filtering. Recommender systems make their recommendations based on data that is available on the system. These systems have gained popularity over the years and they have been adopted in many domains. In this study a recommender system for mobile subscriber provisioning was developed using a hybrid *J48* and *k-means* algorithms. The *J48* algorithm was used for classifying subscribers per usage stream and then *k-means* was used to cluster all the subscribers of similar usage patterns. The algorithms were selected after being compared with other algorithms and the two performed best in their categories. The clustering algorithm, *k-means,* was able to cluster the sample data as follows: Cluster 0 contained 48% (1621) of the subscribers cluster 1 contained 42% (1423) subscribers, cluster 2 contained 8% (272) subscribers and lastly cluster 3 contained 74 subscribers representing 2% of the population and the run time of *k-means* is faster than that of *EM*. The classification algorithm *j48* performed at an average of 99.98% for correctly classifying instances and this was higher than the Naïve Bayes, zeroR and MLP algorithms. The developed recommender system was able to successfully recommend contract packages to subscribers. A precision-recall curve was produced, and it showed good performance of the system. This study successfully highlighted the challenges in recommender systems, and showed that a hybrid system was better able to recommend products to the mobile subscribers.

CHAPTER 1

1 INTRODUCTION

1.1 INTRODUCTION

Recommender systems automate the process of recommending products, services or information items (hereinafter referred to as items) to consumers (referred to as subscribers) based on several types of data concerning users, items and previous interactions between users and items (Trewin, 2000). Deshpande et al. (2007) have revealed that there seems to be no single algorithm that is best for this purpose and relative performances of different algorithms are largely domain- and data-dependent.

This dissertation reports on the development of a mobile recommender system to assist TELecommunications COmpanies (TELCOs) in making product recommendations. Recommender systems are a subclass of what rating a user may give to a certain item. They are widely used in the movie, shopping and other domains. In the movie domain, recommendations are made on how other users have rated the item and the types of movies that the user has been interested previously. The primary purpose of Recommender systems' is to recommend to a user, items that the user may be interested in, with a likelihood that the user may not be interested in the recommended item.

In a web-based environment, recommender systems recommend items to a user based on item ratings and similarity. Recent studies have shown that using a Recommender System (RS) can lead to increased sales volumes in the short term and in the long term or help to increase sales diversity by directing customers to other parts of the available product catalog (Fleder and Hosanagar, 2009). This technology has been successful for web users in providing targeted item recommendations but only a few have been designed for mobile users (Ricci and Nguyen, 2007).

Recommender systems use several different technologies. These items can be classified into two broad groups. Content-based (CB) systems which examine properties of the items recommended and Collaborative filtering (CF) systems which recommend items based on similarity measures

between users and/or items. In a recommender system application, there are two classes of entities which are referred to as users and items. Users have preferences for certain items, and these preferences must be teased out of the data (Koren et al., 2009).

For a recommendation to be possible, the researcher proposes that a user inputs his/her mobile number and the recommender system then recommends all suitable items based on factual information retrieved from the database. After a user recharges only one Short Message Service (SMS) should be sent with recommendations. This has the potential to be used by mobile subscribers because it eliminates sending many SMSs to the user and recommending packages that are not relevant. It is important that when recommendations are made from a mobile subscriber recommender system, the recommendation is based on Call Data Records (CDR) for that subscriber. The use of CDR will enable the recommender system to produce valuable results since the information retrieved is factual and the user is most likely to react to the recommendation, because the recommendation will be personal (based on user data) and not general (based on a group information). This approach will eliminate erroneous recommendations of products that the user cannot afford.

One of the longest-standing challenges in recommender systems is how to deal with the cold start problem for new items (Riedl and Riedl, 2013). Rather than passively consuming a set of recommended items, users often want to be in control of their interaction with the items in a system. The recommendations are often best used as guides through a complex item space, letting the user choose in what directions to move at each step (Ricci,2010).

Mobile devices were first invented in 1973. The development of mobile devices was initially not meant for ordinary consumers, but for high profile business people. They were expensive and did not perform well. However, the devices served the primary purpose, which was to send a message across from one subscriber to another. It has been over forty years now since the invention of the first mobile device and there has been a transition on how the devices are currently used. Ten years later the first mobile phones went on sale. The primary purpose of the first devices was voice and for decades TELCOs generated their revenue from voice. However, this has changed with the availability of the internet and the introduction of instant messaging platforms like Mxit, WhatsApp, Imo etc. In the early 1980's there was no use for mobile recommender systems due to the limited services and capability of mobile devices. Currently voice is no longer the main revenue generating activity for TELCOs. Data has gained popularity and has taken over as a revenue

stream. It is important to note that voice is still in use by mobile subscribers and it has not faded. Therefore, mobile recommender systems are expected to make recommendations based on a criterion that will help increase the accuracy.

1.2 STATEMENT OF THE PROBLEM

Current mobile telecom networks store large amounts of data including the duration of the call, the time the call was executed, the subscriber payment plan (Pre-paid or Post-paid) etc. in their databases. There is an ever-increasing complexity of understanding behavioral patterns of subscribers' preferences which render the traditional recommendation approaches not efficient in meeting customer demands. A new recommendation paradigm is required that uses techniques that exploit user-specific information based on time, usage, interests, etc., rather than common techniques to prioritize recommendations.

One of the greatest challenges faced by recommender systems is the lack of recommendation accuracy. It is evident from Ricci et al. (2006) that this is caused by the ambiguity in querying of information and or the structure of the database. Adding to the challenge is the gaining popularity of different kinds of devices that use different communication standards. This has made it difficult for mobile subscriber recommender systems to be accurate in recommending items to users, leading to recommendation errors. A user might not have a device that is internet enabled and that user then receives a recommendation that has data bundles in it. That kind of recommendation does not address the user's needs and the user is likely not to react to the recommendation.

1.3 RESEARCH QUESTION

How can a mobile subscriber recommender system be developed in such a way that it takes into consideration usage patterns of the mobile subscriber to make a recommendation?

To answer the research question the following sub-questions were answered:

- What recommender systems methods and techniques are currently available?
- How can a mobile subscriber recommender system be developed in such a way that it addresses the needs of a mobile subscriber (user)?
- How to measure the effectiveness and efficiency of the developed recommender system?

1.4 AIM/ OBJECTIVES OF THE PROJECT

The primary research objective is to explore how current recommender system algorithms can be optimized using content and collaborative filtering for mobile subscribers.

1. To compare different recommender systems algorithms from the literature.
2. To highlight the challenges in recommender systems.
3. To develop a mobile subscriber recommender system.
4. To measure the effectiveness of the proposed recommender system.

1.5 SIGNIFICANCE OF THE STUDY

The use of CDR based recommendation is novel and little research has been done in this area, considering that most recommender systems are based on user ratings and user history. However, this study is expected to make a major contribution towards how TELCOs recommend products to users. This study is also expected to benefit e-commerce sites by encouraging them to purchase user data from TELCOS. This project is expected to also contribute in highlighting the problems, challenges and other issues that are faced by current recommender systems for mobile subscribers. The perception of how mobile subscriber recommender system algorithms are developed in the future is expected to be changed.

1.6 LIMITATIONS

Although this research is being carefully prepared, the researcher is aware of the limitations and shortcomings that might arise. The limitations and shortcomings are outlined below:

The research was conducted based on data that is gathered from one Operational Company (OPCO) with a larger subscriber base. This research was also conducted over a period of two (2) years, and ten thousand (10 000) subscribers were used. Also considered is that some subscribers may move to a different network or stop using the OPCO services and the future usage might not be available for comparisons. Ten thousand subscribers might be a low number for mobile subscribers to be considered for this study. Another limitation is the fact that the data is massive and some users have dual-sim enabled mobile devices. They only use the mobile carrier for

receiving calls and not for other revenue generating activities (SMS, GPRS, VOICE). This is a limitation because it might produce low Revenue Generating Subscribers (RGS) that might lead to less recommendations. In addition, the OPCO will provide historic data for security reasons.

1.7 RESEARCH DESIGN

This study followed a qualitative research approach and literature exploration and experiments were conducted. The data was collected from the OPCO. The data was then loaded into a Java data manipulation tool, pre-Processed, analyzed, and interpreted by the researcher. The pre-processed data was thus used in experiments to produce recommendations to the subscriber. The proposed design was expected to derive results after the data has been manipulated using data manipulation tools. Lastly the recommendations were displayed to the subscriber.

1.8 THESIS LAYOUT

Chapter 2. Review of related works on recommender systems, mobile recommender systems and discussion based on theoretical framework for this research work.

Chapter 3. Reviews literature on clustering and classification as well as evaluations of recommender systems are discussed.

Chapter 4. The methodology followed for this study and algorithms proposed are discussed.

Chapter 5. Presents the experiments, results and the interpretation of experimental results.

Chapter 6. Conclusion, Contribution and Future Work. The conclusion of the research, the achievements, shortfalls and future work is discussed.

CHAPTER 2

2 MOBILE RECOMMENDER SYSTEMS

Mobile devices are gradually increasing in popularity since the prices for these devices are gradually decreasing and people can afford to own them (Smith and Brown, 2005). It is clear from Polatidis and Georgiadis (2013) that the use of cellular phones as well as the rapid growth of the internet has generated an information overload problem. However, in a mobile recommender system space the usage of mobile devices brings about a new phenomenon and changes the focus of recommendations to individuals, thus making personalization a key factor of these mobile devices. It is evident from Woerndl et al. (2007) that in a mobile situation, information personalization is more difficult for the reason that there are constrains in mobile devices regarding displays and bandwidth etc. However, it can be said that personalization is not a new research topic within the recommender systems theory because Amazon.com, provides a personalized web page to each individual user (Polatidis and Georgiadis, 2013).

Mobile recommender systems increase loyalty by taking personal information as inputs from the user and generating recommendations locally or in a distributed environment. It directs the predictions in a form of a recommendation to the interface that the user is using. With the hasty development of mobile computing technologies, various kinds of mobile applications gained popularity (Gavalas and Economou, 2011). As a groundbreaking technology, mobile computing enables access to information anytime, anywhere, even in surroundings with scarce physical network connections. Amid others, the operative use of mobile technology in the field of mobile tourism has been actively studied. Along this line, mobile RSs (i.e. RSs tailored to the needs of mobile device users) represent a relatively recent thread of research with numerous potential application fields (e.g., mobile shopping, advertising/marketing and content provisioning) (Ricci, 2011). For instance, Yang et al. (2008) proposed a location-aware recommender system that accommodates customers' shopping needs with location-dependent vendor offers and promotions. Yuan and Chao (2010) introduced a framework which enables the creation of tailor-made campaigns targeting users according to their location, needs and devices' profile.

Due to the adoption of mobile devices many studies have been conducted in the mobile recommender systems space. Yang et al. (2008) studied recommender systems that considers the

location of the mobile user and proposed a location-aware recommender system that considers a user's shopping needs with a location-dependent vendor's offers. However, other researchers such as Dunlop et al. (2004), Setten et al. (2004) and Tung and Soo (2004) states that only a few recommender systems are designed for mobile users and very few existing mobile recommender systems are conversational. Ricci and Nguyen (2005) illustrated a mobile recommender system called MobyRek which has been designed to run on a mobile phone with limited input from the user, thus reducing user interaction and making the system proactive. Proactivity for recommender systems is a good utility.

Baltrunas et al. (2012) further argues that context is crucial in mobile recommender systems, in their study of Context Aware Recommender Systems (CARS), they study the relationship between an item rating and context for example an outdoor restaurant can be rated 5 in summer because it's hot then be given a lower rating in winter because customers feel cold when visiting the restaurant.

Polatidis and Georgiadis (2013) argued that although recommender systems are in every personal computer and mobile device, there are many factors that users of mobile devices take into consideration and avoid their use. These factors are related to privacy. Additionally, the internet era has made it possible for social networks and its offshoots that involves swapping large amounts of data daily, these can be used in recommender systems to improve personalization.

Implementations of recommender systems in literature.

Entree is a restaurant recommender system that practices case-based reasoning techniques to select and rank restaurants (Burke, 2002; Bridge et al., 2005). The interaction with these systems works in the following manner: a user adds as input an entry point, the input can either be a known restaurant or a set of criteria; then the user is shown restaurants that are alike. Furthermore, the interaction between the user as well as the system is in a manner that is a dialogue, critiquing the system's suggestions and interactively refining the search until an acceptable option is achieved (Burke, 2002).

Items that can be recommended to the user are represented by a set of features, also called *attributes* or *properties*. For example, in a movie recommendation application features that

describe a movie are: actors, directors, genres, subject matter, etc. Thus, ensuring that each item is described by the same set of attributes and there is an acknowledged set of values the attributes may take, the item is represented by means of structured data. In this case, many algorithms can be used to learn about a user profile thus eliminating the cold start problem for many recommender systems (Kumar and Sharma, 2016).

Informed Recommender utilizes consumer product reviews to make recommendations. The system converts consumers' opinions into a structured form by using a translation ontology, which is exploited as a form of knowledge representation and sharing. Sarwar et al. (2001b) studied weighted hybrid recommender systems which were defined as systems in which the score of a recommended item is computed from the results of all the available recommendation techniques present in the system. The P-Tango system uses a hybrid model that initially gives collaborative and content-based recommenders equal weight but progressively adjusts the weighting as predictions about user ratings are confirmed or disconfirmed. A weighted hybrid has several benefits one of which is that all the system's capabilities are based on the recommendation process in a way that is straightforward and easy to make recommendations (Burke, 2002). However, this technique has an implicit assumption that the relative value of the different techniques is uniform across possible items.

Terveen and Hill (2001) demonstrated that content information can lead to improved recommendations. However, the content should be encoded in a manner that is appropriate (Funakoshi and Ohguro, 2000). The *Fab* system tackles issues of content-based filtering and social filtering. In the *Fab* system, content information is sustained by two types of agents: *user agents* as well as collection agents; user agents are associated with individuals and *collection agents* are those that are associated with sets of documents (Terveen and Hill, 2001).

Research on the use of c*lique-based* and *feature-based* models for mobile subscribers has been demonstrated (Greene et al., 2010). A clique is a set of subscribers whose usage patterns are almost the same without considering data volumes or revenue but in the usage manner. A clique can also be used to study subscriber preferences and enhance the recommendation engine thus adding more items to the recommendation list.

2.1 ISSUES AND GENERAL CHALLENGES FACING MOBILE DEVICES AND MOBILE RECOMMENDER SYSTEMS

Most of the issues that are highlighted when designing recommender systems for mobile users refer to mobile devices in the class of mobile phones and sim-card enabled tablets. It is imperative to note that recommendations on small screen devices can be difficult and frustrating for end-users. It is evident that users are able to read and understand information on small interfaces; however the size of the screen has an impact on performance (Jones and Marsden, 2005). In web-based recommender systems for mobile users, a user might have to scroll down to find the item on recommendation. However, the longer the user takes to scroll the lower the chances of the item being clicked by the customer (Yeung et al., 2012). In addition, a user on a small screen is less effective in completing an assigned task when compared to users with large screens.

Another issue associated with mobile recommender systems is data costs. Most mobile recommender systems are run online and a user must have data to interact. The cost of data often impacts how much time a user will spend on a site. Due to the limitations stated above, information search browsing, and in particular item recommendations, is a problem on mobile devices. Likewise, it is difficult for users to process the result lists returned especially on small screens. Church and Smyth (2008) scrutinized the variances between browsing and search behavior on the internet using mobile devices as compared to the using of personal computers. They revealed that browsing remains to be dominant for mobile information access; however, they also stated that search is gaining popularity as an alternative to information access especially in relation to certain types of mobile handsets and information needs. Due to the above reasons, many search models have been designed specifically for mobile devices and mobile recommender systems.

Publicly-accessible adaptive systems such as collaborative recommender systems present a security problem. Attackers, who cannot be readily distinguished from ordinary users, may inject biased profiles to force a system to "adapt" in a manner advantageous to them. Such attacks may lead to a user losing trust in the objectivity and accuracy of the system. Recent research has begun to examine the vulnerabilities and robustness of different collaborative (Chee et al., 2001) recommendation techniques in the face of "profile injection" attacks (Bamshad et al., 2007). Before making recommendations to users', security must be in place and a communication channel

should be secured so that the recommendation is not tampered with and reaches the target in its original state.

It is evident from Chakraborty and Karforma (2013) that recommender systems are vulnerable to different types of profile-injection attacks where a number of fake user profiles are inserted into the system to influence the recommendations made to the users. However, when the data is analyzed offline injected profiles can be detected before data is processed further thus making offline data manipulation worth considering for non-real-time recommender systems.

2.1.1 COLD START ISSUES

The "cold-start" issue depicts circumstances in which a recommender system cannot make significant recommendations because of an underlying lack of ratings. This issue can fundamentally corrupt the framework execution. It can happen under three situations: new user, new item and new group (Riedl and Riedl, 2013):

New User. When a user first registers with a Collaborative Filtering (CF) service, no rating is available for that specific user. Thus, predictions cannot be personalized. The cold-start issue may be solved in several ways. For example, by:

- Forcing the user to rate a few items before allowing that user to use the service.
- Displaying the global average until there is enough ratings by the user to personalize the recommendations.
- Asking the user for demographic information.

New Item. This is particularly more troublesome for new items because they will not be recommended till someone gives the item a rating and therefore it is unlikely that users will rate an item that is not recommended to them. However, in domains where there may be many unrated items that are very good numerous methods can be used, including:

- recommending items through non-CF techniques such as content analysis or metadata
- using a selection criterion and requesting users to rate unrated items

New group. A huge cold-start problem is bootstrapping another group. On the off chance that another administration's esteem is in its customized CF suggestions, at that point without ratings it might not have adequate different ratings in this manner not retain users sufficiently long to develop evaluations. The most well-known arrangement is to give rating motivating forces to a little "bootstrap" subset of the group before welcoming the whole group to utilize the service. Different methodologies are used to keep up users' enthusiasm through exchange service (Kumar and Sharma, 2016).

2.1.2 COMPARISON OF COLLABORATIVE FILTERING AND CONTENT BASED FILTERING

Both collaborative filtering and content-based techniques have their own strengths and weaknesses. The combination of the two techniques allows the strength of the one technique to overcome the weakness of the other technique; hence they are usually combined and a hybrid is formed.

Table 1. Is a comparison of the different techniques.

Table 1. Comparison between collaborative filtering and content based

Focus area	Collaborative filtering	Content based
No need for domain knowledge	Yes	Yes
Adaptive	Yes	Yes
Quality improves as new data is loaded	Yes	Yes
Implicit feedback adequacy	Yes	Yes
Impacted by cold start problem	Yes	No
Susceptible to new subscriber problem	Yes	Yes
Susceptible to new product problem	No	Yes
Quality of the recommendation depends on the size of the data	Yes	Yes

Both CF and Content Based Filtering (CBF) techniques have no need for domain knowledge. This means. Both content based and collaborative filtering are of adaptive nature and can build recommendations with implicit feedback and no domain knowledge (Kumar and Sharma, 2016).

However, both techniques improve as there is more data. CF is susceptible to new products which is a problem that CBF addresses. The more data there is on the system the better the quality of the recommendation.

2.2 KNOWLEDGE BASED RECOMMENDER SYSTEMS

Knowledge based recommender systems contain knowledge about the product and the need that the product satisfies. The recommendation is based on the functional knowledge about the product and the user (Ricci, 2011). One of the systems that adopted the method is the Personal Logic recommender system which is a system that bargains a dialog that effectively guides the user through a discrimination tree of product features (Trewin, 2000). The knowledge is then used for the recommendations that the system produces.

2.3 COLLABORATIVE FILTERING

Collaborative filtering (CF) is rooted in what people have been engaged with for a very long time, i.e. making recommendations from past experiences. A typical example would be students discussing a book that they have enjoyed in the past. If the student has enjoyed reading similar books in the past the others will also have interest in reading the book. Chandramouli et al. (2011) argued that collaborative filtering is a popular and heavily used recommendation method. The collaborative filtering method recommends the items to the target user on the foundation of historical penchants of other users with a comparable sense of taste (Schafer et al., 2007b). CF works similarly to word of mouth because it uses the opinion of others. CF techniques are classified into two categories, namely, memory based methods and model based methods (Liu et al., 2016).

- **memory-based methods** - exploit the complete user database to make recommendations.
- **model-based methods**- first fit a model based on the user database and then use the fitting to make a recommendation.

Even though collaborative filtering is a good technique it has challenges when new items are added onto the system. One of the well-known implementations of collaborative filtering in the movie domain is Movielens. Basically, a user rates a movie using a rating scale from 1-5, where 1 is an indication that the movie is bad and 5 indicates that it is extremely good thus making 3 the average.

Collaborative filtering systems are used by users for one or more of the following tasks:

- Finding items that the user might like.
- Getting advice on a specific item.
- finding items that add to a whole group, not just an individual
- getting help with domain specific tasks

The two main objectives of any collaborative filtering system are prediction and recommendation.

- **Prediction:** a recommender system need to be prepared to offer alternatives to users and present them to the user
- **Recommendation:** this part of the system is more dependent on the prediction. Firstly, the system has to predict what the "rating" on the item would be before recommending the item to the user (Sarwar et al., 2001b).

2.4 RECOMMENDER SYSTEM ALGORITHMS

Many studies have been conducted on the provision of context-aware information services such as personalized news delivery (Wang et al., 2010; Yeung et al., 2012).Mobile information recommendation is becoming very popular due to the growing diversity, availability and use of mobile information services (Yeung et al., 2012). However, most systems are user driven and require users to express their interests and input the keywords. Collaborative filtering methods encompasses probabilistic and non-probabilistic methods. It can be said that non-probabilistic models are expansively used by practitioners and Probabilistic models have been widely used in the machine learning community (Schafer et al., 2007a).

Non-probabilistic algorithms: it is evident from Schafer et al. (2007a) that the most well-known CF algorithms are nearest neighbor algorithms. However, nearest neighbor algorithms are not limited to neighbor algorithms. There are also graph-based algorithms (Sarwar et al., 2001a), neural networks (Shams and Haratizadeh, 2016), and rule-mining algorithms (Lin et al., 2001).

Neighborhood algorithms are computationally expensive. For example, to accurately get the distance between two neighbors the whole table must be scanned, and the distance must be calculated for different items. There are many studies in the literature that try to reduce the shortcomings of neighbor algorithms. The studies include clustering, which is used to identify

neighbors in a quicker way (Sarwar et al., 2001a). Mutually *k-means* clustering, and hierarchical divisive and agglomerative clustering can segment users/items into clusters. However, another challenge in employing clustering is that clustering schemes employ distance functions, like Pearson correlation, to form the clusters and measure distance from a cluster (Schafer et al., 2007a).

Cho et al. (2002) also stated that Collaborative filtering identifies customers (neighbors) whose interests are similar to those of a given customer and recommends products neighbors of a given customer. However, despite their success their widespread use has exposed two major limitations (Godfrey, 2007). The first is related to sparsity. The number of ratings already obtained is very small compared to the number of ratings that need to be predicted because typical collaborative filtering requires explicit non-binary user ratings for similar products. As a result, collaborative-filtering based recommendations cannot accurately compute the neighborhood and identify the products to recommend.

Algorithms to find the neighborhood usually require very long computation times that grows linearly with both the number of customers and the number of products. With millions of customers and products of real world situations, existing collaborative-filtering based recommendations suffer serious scalability problems (Sarwar et al., 2001b).

2.5 TRUST IN RECOMMENDER SYSTEMS

The Trust-Aware Recommender System (TARS) is the recommender system that suggests the meaningful information to the users on the root of trust. Trust is the measure of readiness to be certain in a user based on their aptitude (e.g. goodness, strength, ability) and conduct within a precise context at a given time. It is a directional relationship from the trustor – the user that assesses its trust on the target user – to the trustee – the user that is the target of the trust evaluation. Massa and Avesani (2007) stated that an empirical appraisal on *Epinions.com* dataset showed that recommender systems that make use of trust information are the most effective in terms of accuracy while preserving a good coverage. This is especially evident on users who provided few ratings.

In the pursuit to embody trust in recommender systems a technique emerged to deal with the quality assessment in open environments and to ask users to explicitly specify which other users

they trust. For example, on Epinions.com, a site where users can review products, users can also specify which other users they trust i.e. “reviewers whose reviews and ratings they have consistently found to be valuable”. This then helps with creating trust nodes.

Trust-based recommender systems (Massa and Avesani, 2007) is an developing turf to bargain users personalized item recommendations (e.g., movies, books, etc.) based on historical ratings given by users and the trust relationships among users (the users can be friends on social media). In general, RSs will recommend items based on past experiences and assume that the information is reliable but, in reality, this assumption is untrue and thus makes the evaluation of content provided by users an important issue (Avesani et al., 2005). Current studies reveal that current trust metrics cannot deliver substantial performance and indicate that future metrics should be designed wisely (Bamshad et al., 2007).

Trust is a complex term and therefore for recommender systems it is coiled differently. It can be said that trust is mostly defined as correlated with analogous likings towards items normally rated by more than one user (Pitsilis and Marshall, 2004a; Papagelis et al., 2005: Lathia et al., 2008). It is evident from Guo et al. (2014) that trust has properties including Asymmetry, Transitivity, Dynamicity and Context Dependence.

Asymmetry. Trust is subjective and personal. Therefore, user A can trust user B and user B does not necessarily trust user A.

Transitivity. This is the most important property of trust that is heavily used in trust-based recommender systems. It says if user u trusts v, and v trusts p, it can be inferred that user u trusts p to some extent. It is in line with real life situations in which people tend to trust the friend of a friend rather than a stranger. Predictive performance is improved by improving or increasing more trusted friends.

Dynamicity. Generally, trust is built cautiously and over time. Another commonplace is that trust is hard to establish but easy to destroy.

Context Dependence. Trust is context-specific. A user that is trusted in one domain may not necessarily be trusted in another domain.

2.5.1 TRUST METRICS

There are numerous trust metrics that are put forward to compute contained trust using ratings from users. These are mainly based on the instinct that the users whose ratings are similar or close to each other, tend to be frank (Jamali and Ester, 2009). For discussions to be facilitated an introduction of several notations is necessary. Thus, Denote U, I *and* R as the users, items and ratings, respectively. For ease, the symbol u, v are held in reserve for users and i, j for items; hence $r_{u,t}$ represents a valuation given by user u on item i. Let I_u be the set of items rated by user u, and $t_{u,v}$ be the trustworthiness of user v towards user u.

Five (5) trust metrics denoted by **M1-M5** are elaborated as follows:

M1- Lathia et al. (2008) stressed the worth of acquiring scores provided by other users. This entails that a user who makes available opposite ratings is a trustworthy user as compared to a user who is not eager to take part in sharing views. Trust is defined as the average of the values provided by users over all items rated.

$$t_{u,v} = \frac{1}{|I_{u,v}|} \sum_{t \in Iu,v} (1 - \frac{|r_{u,t} - r_{v,t}|}{r_{\max}}). \qquad 1$$

where $I_{u,v} = I_u \cap I_v$ is the set of commonly rated items by user's u and v, and $r_{\max}$ is the maximum rating scale predefined by a recommendation system. According to Equation 1. It is frivolous to find that trust is symmetric, i.e., $t_{u,v} = t_{u,v}$ and there is no consideration of time and contextual information.

M2- Papagelis et al. (2005) defined trust through user similarity computed by Pearson correlation coefficient (PCC)

$$S_{u,v} = \frac{\sum_t (r_{u,t} - r_u)(r_{u,t} - r_u)}{\sqrt{\sum_t (r_{u,t} - r_u)^2} \sqrt{\sum_t (r_{u,t} - r_u)^2}} \qquad 2$$

where $S_{u,v}$ is the similarity between users u and v, and trust is assigned as similarity, i.e., $t_{u,v} = s_{u,v}$.

More generally, researchers lean towards the use of a threshold to figure out if a user with a specific similarity is trustworthy. To support this, Yuan et al. (2010) used threshold similarity to form binary trust ideals without lack of generality and proposed the formalization below:

$$t_{u,v} = \begin{cases} S_{u,v} & if\ s_{u,v} > \theta s,\ |I_{u,v}| > \theta I; \\ 0, & otherwise; \end{cases} \quad 3$$

where θs , $\theta 1$ are the threshold of the user similarity and the number of co-rated items, respectively. One characteristic of similarity measures is symmetry, i.e., $S_{u,v} = S_{u,v}$. Hence, the trust based on PCC is also symmetric, In addition, Castro Sotos et al. (2009) in their study hypothesized that PPC is not transitive unless under stringent conditions i.e. $S_{u,v} > 0,707$ when users are extremely correlated. Thus to enable the transitivity of trust it is obligatory to set similarity threshold $\theta s = 0.707.$ Further, Guo et al. (2013) reported that PCC is not reliable when the length of rating vectors is short; hence, the threshold $\theta 1$ is to ensure that the computed PCC value is more dependable.

M3 Hwang and Chen (2007) simplified Resnick's prediction formula and computed a predicted rating using a simple version of only based on a single user:

$$P_{u,i} = \overline{r}_u + (r_{v,i} - \overline{r}_v), \quad 4$$

where $\overline{r}_u \ and \ \overline{r}_v$ are the mean ratings of users u and v, respectively. The trust score is then derived by averaging the prediction error on co-rated items:

$$t_{u,v} = \frac{1}{|I_{u,v}|} \sum_{t \in Iu,v} (1 - \frac{|P_{u,i} - r_{u,i}|}{r_{\max}}). \quad 5$$

where $P_{u,i}$ is the predicted rating for a single user and $r_{\max}$ is the maximum rating scale predefined by a recommendation system.

Shambour and Lu (2012) adopted the same strategy but computed trust based on Mean Squared Distance (MSD).

$$t_{u,v} = \frac{|I_{u,v}|}{|I_u \cup I_v|}\left(1 - \frac{1}{|I_{u,v}|}\sum_{t \in Iu,v}\left(\frac{|P_{u,t} - r_{u,i}|}{r_{\max}}\right)^2\right). \quad (6)$$

The user who has trust values that are above the threshold λ, i.e., $t_{u,v} > \lambda$ is viewed as a trusted neighbor. It is noted that both Equations 5 and 6 results in symmetric trust. However, both equations do not consider the dynamics and context property of trust.

M4 O'Donovan and Smyth (2005) considered a rating provided by others as accurate if the absolute difference between the foreseen rating $P_{u,i}$ and the ground truth, $r_{u,i}$ is less than a threshold $\in$:

$$correct(r_{u,i}, r_{v,i}) \Leftrightarrow |P_{u,i} - r_{u,i}| \leq \in, \quad (7)$$

where $P_{u,i}$ is given by Equation 4. Then two kinds of trust are defined using the notion of correctness: profile-level and item-level trust. The former trust is defined as the ratio of correct ratings over all the ratings provided to generate predictions:

$$t_{u,v} = \frac{|CorrectSet(v)|}{\mathrm{Re}\,cSet(v)}, \quad (8)$$

where $CorrectSet(v)$ represents the set of correct ratings provided by user v, and $\mathrm{Re}\,cSet(v)$ denotes the set of recommendations that user v has been involved in.

The item-level trust of a user for a certain item to be consistent with other metrics, trust is only considered at user level, that is, profile-level trust rather than item-level trust. Since the absolute value is adopted in Equation 7, the proposed trust metric is too symmetric. Note that setting a low

value of $\in$ is not in favor of trust formation whereas setting a high value will tend to treat other users equally trusted.

M5 Pitsilis and Marshall (2004b) adopted the subjective logic (Jøsang, 2001) to define trust. The uncertainty is redefined as the inability to make accurate predictions:

$$u_v = \frac{1}{|I_{u,v}|} \sum_{t \in Iu,v} (1 - \frac{|P_{u,i} - r_{u,i}|}{r_{\max}} \qquad 9$$

where u_v is the illustration of ambiguity towards user v, and $P_{u,i}$ is derived from equation 4. Then the belief and disbelief are defined as:

$$bv = \frac{1}{2}(1 - u_v)(1 + s_{u,v});$$
$$dv = \frac{1}{2}(1 - u_v)(1 - s_{u,v}); \qquad 10$$

where $s_{u,v}$ is the parallel between users u and v computed by Equation 2. Hence, it satisfies the requirement for subjective logic: $b_v + d_v + u_v = 1$. The belief b_v is used as the direct trust that user u has on user v, i.e., $t_{u,v} = b_v$. The benefit of this metric is that it considers disbelief rather than only taking belief into consideration.

The TARS notations can be found in Table 2.

Table 2. Notations that are used in the TARS

Symbol	**Explanation**
i	Item
a	Current user
u	Recommender
r_a	Average rating of the current user
r_u	Recommender's rating on the item
$r_{u,i}$	current user's rating on item
$w_{a,u}$	Current user's weight to the recommender
$P_{a,i}$	Predicted rating for the current user on item

Figure 1 shows the architecture of TARS where the inputs are trust matrix and rating matrix and the outputs are predicted rating. The rating prediction mechanism of the conventional TARS model is like that of CF. The difference of opinion is that CF weights each recommendation based on the active user's similarity with the recommender, while TARS weighs each recommendation based on the active user's trust on the recommender.

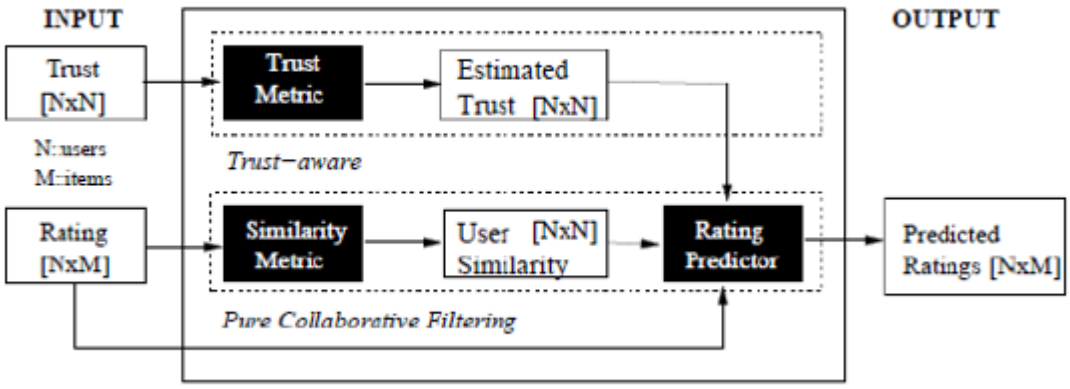

Figure 1. The rating prediction mechanism of conventional TARS of a pure collaborative filtering.

The predicted rating for the active user on the item can be calculated as in equation 11 and the Current user's weight to the recommender can be found on equation 12.

$$p_{aj} = \overline{r}_a + \frac{\sum_{u=1}^{k} w_{a,u}(r_{u,i} - \overline{r}_u)}{\sum_{u=1}^{k} w_{a,u}} \qquad 11$$

$w_{a,u}$ can be calculated as given in equation 12:

$$w_{a,u} = \frac{d_{\max} - d_{a,u} + 1}{d_{\max}}, \qquad 12$$

where $d_{\max}$ is the Maximum Allowable Propagation Distance (MAPD) between users of the recommender system. The value of MAPD is present by the administration of TARS. $d_{a,u}$ is the active users' (a) trust propagation distance to the recommender (u).

In TARS, the trust propagation distance refers to the number of hops in the shortest trust propagation path from the trustor to the executor.

As shown in the prediction mechanism of the conventional TARS model, MAPD is the fundamental parameter for the rating prediction. However, the works of TARS did not provide any mechanism to set MAPD. just arbitrarily select some value for this extremely important parameter.

For example, in Massa and Avesani (2009), the authors randomly set the value of MAPD as 1,2,3 and 4 to conduct different experiments of TARS. They did not consider the relationship between the value of MAPD and the scale of TARS. On one hand, if the value of MAPD is set too small, TARS might lose some valuable recommendations. On the other hand, the computational complexity of constructing trust networks for TARS is $o(k^{d_{\max}})$, in which k is the number of trust stated per user, and $d_{\max}$ is the value MAPD, so if the value of MAPD is set too high the computational of TARS increases exponentially. Spontaneously, the optimized value of MAPD for TARS should have some relationship with the topology of the trust network and optimizes the conventional TARS model based on the topology of the trust network (Massa and Avesani, 2009).

2.6 SIMILARITY IN RECOMMENDER SYSTEMS

Below is a representation of three (3) types of similarities that are applicable in recommender systems depending on the specific objective. These three are, namely, cosine-based similarity, adjusted cosine-based similarity and correlation-based similarity.

2.6.1 COSINE-BASED SIMILARITY

In cosine-based similarity two items are thought of as two vectors in the m dimensional user-space. The similarity between them is measured by computing the cosine of the angle between these two vectors. Formally, in the $m * n$ ratings matrix below the similarity between items i and j, denoted by $sim(i, j)$ is given by:

$$sim(i, j) = \cos(\bar{i}, \bar{j}) = \frac{\bar{i}.\bar{j}}{\left\|\bar{i}\right\|_2 * \left\|\bar{j}\right\|_2} \qquad 13$$

where "." Denotes the dot-product of the two vectors and sim(i,j) is the similarity between the item i and item j.

2.6.2 CORELATION-BASED SIMILARITY

In this case, the similarity between two items i and j is measured by computing the $Pearson - r$ correlation $corr_{i,j}$. To make the correlation computation accurate, co-rated cases (i.e., cases where the users rated both i and j *) must* be isolated, as depicted in equation 14. Let the set of users who both rated i and j are denoted by U then the correlation similarity is given by:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_i})^2}\sqrt{(R_{u,j} - \overline{R_j})^2}} \quad 14$$

where $R_{u,i}$ denotes the rating of user u on item *i*, R_i is the average rating of the i-th item and $R_{u,j}$ denotes the rating of user u on item j, R_j is the average rating of the j-th item.

2.6.3 ADJUSTED COSINE SIMILARITY

One vital difference between the similarity computation in user-based CF and item-based CF is that in case of user-based CF the similarity is computed along the rows of the matrix but in case of the item-based CF, the similarity is computed along the columns, i.e. each pair in the co-rated set corresponds to a different user. Computing similarity using basic cosine measure in item-based case has one important drawback-the differences in rating scale between different users are not considered. The adjusted cosine similarity offsets this drawback by subtracting the corresponding user average from each co-rated pair. Formally, the similarity between items i and j using this scheme is given by:

$$sim(i,j) = \frac{\sum_{u \in U}(R_{u,i} - \overline{R_i})(R_{u,j} - \overline{R_j})}{\sqrt{\sum_{u \in U}(R_{u,i} - \overline{R_j})^2}\sqrt{(R_{u,j} - \overline{R_j})^2}} \quad 15$$

where $\overline{R_u}$ is the average of the $u-th$ user's ratings and $\overline{R_j}$ is the mean average rating given by user j.

2.7 PREDICTION COMPUTATION

The most crucial step in a collaboration filtering system is the generation of the output interface in terms of prediction. Once the set of related items based on the similarity measures is isolated, the next step is to consider the target user's ratings and adopt a method to obtain predictions.

Sarwar et al. (2001b) in their study considered these two techniques namely, weighted sum and regression.

2.7.1 WEIGHTED SUM

As the name implies, this method computes the prediction on an item of i for a user u by computing the sum of the ratings given by user on the items similar to i. Each rating is weighted by the corresponding similarity $s_{i,j}$ between items i and j. Formally, using the notion shown in equation 16 we can denote the prediction $p_{u,i}$ as

$$p_{u,i} = \frac{\sum_{all\ similaritems.N} (s_i, N * R_u, N)}{\sum_{all\ similaritems.N} (|s_i, N|)} \quad 16$$

where $R_{u,}N$ is similar item N's ratings values, i is the target items and N is the similar item.

Basically, this approach tries to capture how the active users rate similar items. The weighted sum is scaled by the sum of the similarity terms to make certain that the prediction is within the predefined range.

2.7.2 REGRESSION

This approach is similar to the weighted sum methods but instead of directly using the ratings of similar items it uses an approximation of the ratings based on the regression model. In practice, the similarities computed using cosine or correlation measures may be misleading in the sense that two ratings vectors may be distant (in Euclidean sense) yet may have very high similarity. In that case using the raw ratings of the "so-called" related items may result in poor prediction. Sarwar et al. (2001b) stated that the basic idea is to use the same formula as the weighted sum technique, but instead of using the similar item N's "eaw" ratings values $R_{u,}N's$ this model uses their approximated values $R_{u,}N$ based on a linear regression model. If the respective vectors of the target items i and the similar item N by R_i and R_N are denoted, the linear regression model can be expressed as:

$$\bar{R}_N = \alpha \bar{R}i + \beta + \in \quad 17$$

where α *and* β are the regression model parameters that are determined by going over both the rating vectors and $\in$ is the error of the regression model.

2.8 HYBRID ALGORITHMS

Hybridization is the combination, or the use of all, features from different algorithms in order to make a recommendation. While each of the algorithms described above is dominant in their respective fields, the need for hybridization is gaining more attention (Massa and Avesani, 2009; Ghazanfar and Prugel-Bennett, 2010). Moreover hybrid algorithms have been categorized according to their design (Jannach et al., 2010).

Combining two or more recommendation methods is called hybridization. Hybridization is not a novel concept in recommender systems but it has been applied differently in theory. Woerndl et al. (2007) stated that a hybrid recommender system is a combination of different recommender systems to improve the retrieval of information. In their study, they combine collaborative filtering with other recommendation methods to account for the complexity in context. Different combinations can be fused together to achieve a hybrid for example: weighted, switching, mixed, feature combination or augmentation or cascading (Burke, 2002).

Below is a discussion of some of the algorithms found in literature:

2.8.1 MONOLITHIC

A monolithic algorithm is an algorithm that integrates unique features from each algorithm. It could possibly use every feature that an algorithm has or just support a number of them. How a monolithic algorithm works is shown in Figure 2.

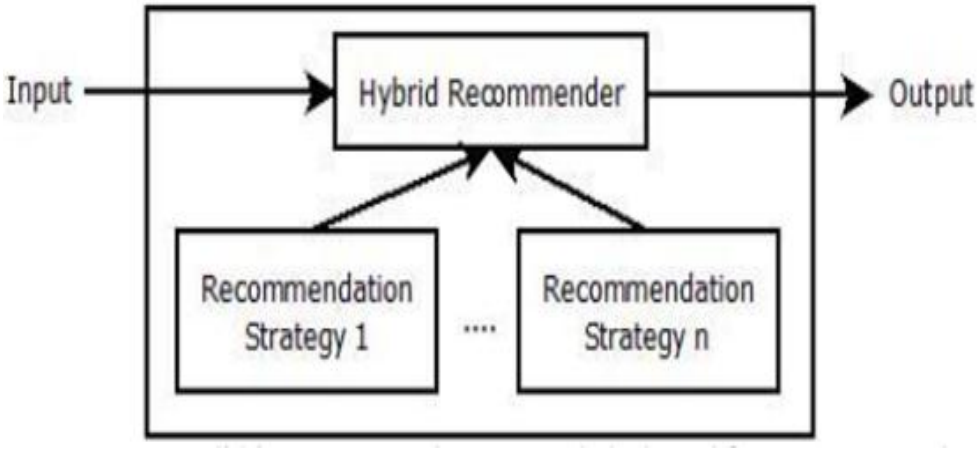

Figure 2. Monolithic Recommender Approach (adapted from Jannach et al., 2010)

2.8.2 PARALLELIZED

A parallel algorithm runs all the predefined algorithms in parallel, then takes the output from each one and passes it to a predefined hybridization stage. A pictorial view of the algorithm is shown in Figure 3.

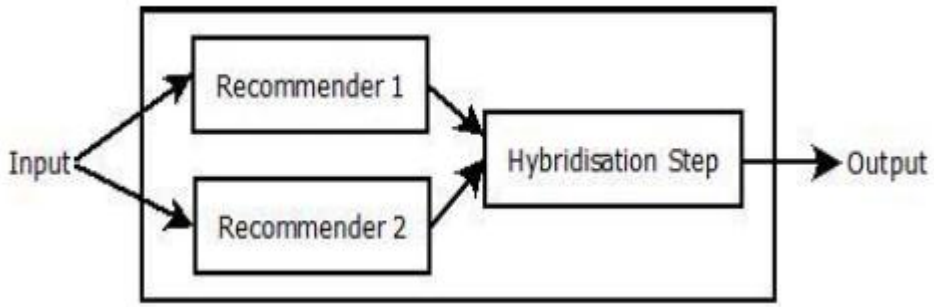

Figure 3. Parallel Recommender Approach (adapted from (Jannach et al., 2010)

2.8.3 PIPELINED

A pipelined algorithm runs each algorithm one after another, in sequence and takes the output from the previous and uses it as the input for the next one until the required outcome has been achieved. A pictorial representation of the algorithm is found in Figure 4.

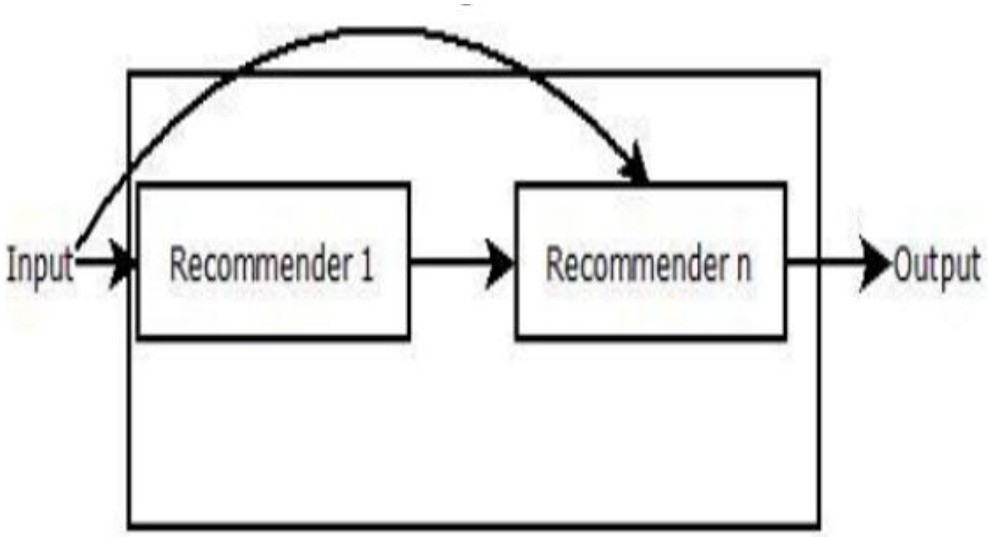

Figure 4. pipelined Recommender Approach (adapted from Jannach et al., (2010))

2.9 RATING MATIX FOR SIMILARITY COMPUTATION

Table 3. Example of a rating matrix for three (3) users

User	Item 1	Item 2	Item 3
James	3	2	4
Lucy	3	4	4
Andrew	5	4	No rating

As seen in Table 3, a user-item matrix allows for the creation of a neighborhood that can be used to create similarity between users for whom we wish to generate recommendations. The user-item matrix is usually sparse since most users do not rate every item or may have used the specific item.

Sparsity can be an issue that can lead to weak recommendations. Two common processes that can reduce the use of sparsity and improve the satisfaction of the recommender systems are:

1. **Default Voting:** setting an appropriate rating for all items that have not been rated. The cost of applying a default rating is low (Adomavicius and Tuzhilin, 2005).
2. **Pre-processing using averages:** Scans through missing items and either averages user's votes or averages the item votes and placing the resulting average as the rating value for the missing user-item matrix entry (Bhaidani, 2008). Looking at Table 2. Andrews' ratings average to 4.5 therefore his rating for item 3 can be pre-processed

2.9.1 NEIGHBORHOOD FORMATION

Generating a neighborhood involves calculating the similarity between the given users within the user-item matrix. Similarity will be used to generate a recommendation for a specific user.

The algorithm folows these steps:

1. Compare the similarity between all users with the active user.
2. Select n Users that have the highest similarity to build a neighbourhood
3. Compute the prediction based on this similarity matrix.

Within the user-based recommendation systems, the similarity between two users is calculated using the Pearson's correlation coefficient. Pearson's correlation (also called the Pearson's product moment correlation after Karl Pearson's) has become a standard way of calculating correlation. Similarity between users u_i *and* u_k are calculated below:

$$sim_{ik} = corr_{ik} = \frac{\sum_{j=1}^{i}(r_{ij} - \overline{r_j})(r_{kj} - \overline{r_k})}{\sqrt{(r_{ij} - \overline{r_i})^2 \sum_{j=1}^{i}(r_{kj} - \overline{r_k})^2}} \quad 18$$

where i is the total number of items, r_{ij} is the rating given by j, and $\overline{r_i}$ is the mean average rating given by user i.

Using Table 3, calculations can be computed and similarities between the users can also be computed, thus enabling prediction. Calculating the similarity between user Lucy and James is calculated as follows using Equation 18:

$$\overline{r_{james}} \frac{3+2+4}{3} = 3 \qquad \overline{r_{lucy}} \frac{3+4+4}{3} = 3.666$$

$$sim_{james-lucy} = \frac{\sum_{j=1}^{3}(r_{james-j} - \overline{r_{james}})(r_{lucy-j} - \overline{r_{lucy}})}{\sqrt{\sum_{j=1}^{3}(r_{james-j} - \overline{r_{james}})^2 (r_{lucy-j} - \overline{r_{lucy}})^2}}$$

$$sim_{james-lucy} = \frac{(3-3)(3-3.666) + (2-3)(4-3.666) + (4-3)(4-3.666)}{\sqrt{((3-3)^2 + (2-3)^2 + (4-3)^2)((3-3.666)^2 + (4-3.666)^2 + (4-3.666)^2)}}$$

$$sim_{james-lucy} = \frac{-4}{\sqrt{43}}$$

$$sim_{james-lucy} = -0.610$$

Table 4. Example of a similarity calculation between three users

User	James	Lucy	Andrew
James	1	0.25	-0.610
Lucy		1	-0.539
Andrew			1

The formula presented in equation 18 can be used to calculate the similarity and the results are displayed in Table 4.

2.9.2 PREDICTION GENERATION

The prediction is a numerical value that represents a predicted opinion of the active user about a specific item. The prediction for a user-based collaborative filtering algorithm needs both the user-item matrix (Table 1) and the similarity matrix (Table 2).

Equation 19 shows how a prediction represents the predicted opinion for the active user u_a about item i_j.

$$pr_{aj} = r_a + \frac{\sum_{j=1}^{3}(r_{ij} - \overline{r_i}) \times sim_{al}}{\sum_{j=1}^{3}|sim_{al}|} \qquad 19$$

where r_{ij} represents the rating that user u_i gives item i_j and $\overline{r_i}$ represents the average rating for user u_i. Equation 19 retrieves similarity sim_{ik} from Equation 18 which represents the similarity between the active user u_a and user u_i (Bhaidani, 2008).

This prediction formula is based on a weighted average of similarities between the active user and all other users combined with the average of the active users' other ratings.

Using Table 2 and Table 3 and based on the ratings found in these two tables, Andrews rating on Item 3 can be achieved using the calculation below.

$$\overline{r_{Andrew}} \frac{5+4}{2} = 4.5$$

$$\Pr_{andrew-item3} = \overline{r_{Andrew}} + \frac{\sum_{t=1}^{2}(r_{i,3} - \overline{r_i}) \times sim_{andrew-i}}{\sum_{t=1}^{2}|sim_{andrew-i}|}$$

$$\Pr_{andrew-item3} = \overline{r_{Andrew}} + \frac{(r_{lucy-item3} - \overline{r_{lucy}}) \times sim_{andrew-lucy} + (r_{james-item3} - \overline{r_{james}}) \times sim_{andrew-james}}{|sim_{andrew-lucy}| + |sim_{andrew-james}|}$$

$$\Pr_{andrew-item3} = 4.5 + \frac{(4-3.666)*-0.539 + (4-3) \times 0.610}{|-0.539| + |-0.610|}$$

$$\Pr_{andrew-item3} = 4$$

Based on the above calculation it is evident that the recommender based algorithm has predicted that Andrew will rank item 3 with a rating of 4.

2.10 USING TIME SERIES IN RECOMMENDER SYSTEMS

Time series can be used as a tool to identify profile injections in recommender systems. However, it can also be used as a measurement for other aspects like measuring the frequency of occurrence over time. Burke et al. (2006) stated that the insertion of a rating that contains a certain profile denotes a time-series. They argue that a profile that is constructed at a speed that is too fast for a human being (for example, 100 ratings in a single minute) is sure to be a profile injection attack. Similarly, ratings on an item can also represent a time series, should an item represent a rapid rating at high speed that could also signal that the item is nuked, thus making a time series an important identifier in nuked profiles. Time series differ in how they are represented. A time series can be constructed as a stochastic or a moving average; the way it is represented is solely dependent on the purpose of use for the time series.

A time-series can only be seen at a predetermined number of periods, and in that occasion the underlying sequence of random variables $(X_1, X_2, ... X_n)$ is just a *n*-dimensional haphazard variable (also known as a random vector). However, it is convenient to allow the number of observations to be infinite. In that case $(\chi_{t,t} = 1,2,3...)$ is named a stochastic progression. Stochastics are very good for identifying trends; hence a time series becomes a useful tool for recommender systems. The information on how a time-series has been used in this study can be found in Chapter 4.

2.11 ITEM BASED AND CONTENT RECOMMENDATION METHODS

There are two major factors that come into play when talking about how user-based recommender systems compare to item-based recommender systems- the quality of the results and the performance results. In an experiment performed by the GroupLens research group, it was concluded that based on building a user-based and item-based recommender system, the item-based recommender system provided better quality of results with a lower MAE (Mean Accuracy Error) than the user-based recommender system (Sarwar et al., 2001b)

2.11.1 ITEM-BASED RECOMMENDATION

This form of collaborative filtering is computed based on item relations and not user relations. An item based collaborative filtering algorithm looks at the set of items that an active user has rated

and computes how similar the set of items are to the target item. The item-based algorithm thus utilizes the similarity and computes the prediction based on weighted averages of the active user's ratings on those similar items (Sarwar et al., 2001b).

Item-based collaborative filtering and prediction algorithm follow the same process as the user-based CF. Item-based collaborative filtering trails the same three processes: representation, neighborhood formation, and prediction generation. Within representation item-based CF uses the user-item matrix seen in Table1. The neighborhood formation creates a similarity matrix. This similarity matrix uses the Pearson Correlation Coefficient; however, it determines the correlation between two items.

$$sim_{jk} = \frac{\sum_{j=1}^{i}(r_{ij} - \overline{r_j})(r_{ik} - \overline{r_k})}{\sum_{l=1}^{l}(r_{ij} - \overline{r_i})^2 \sum_{j=1}^{i}(r_{kj} - \overline{r_k})^2} \qquad 20$$

Equation 20 is used to find the similarity between items j and k, where r_{ij} *and* r_{ik} are the ratings that items j and k have been given by user i. The calculations are done over l users who have rated both items.

Using equation 20, a similarity matrix similar to Table 3 can be developed in which all the items will have a similarity value to each other.

Finally, a prediction can be generated through the use of a weighted sum in which a prediction can be generated a prediction pr_{aj} for user a and item j. This equation captures how the active user rates the similar items. The weighted sum is scaled by the sum of the similar items (Sarwar et al., 2001b).

$$pr_{aj} = \frac{\sum_{k=1}^{l} r_{ok} \times sim_{jk}}{\sum_{k=l}^{l} \left| sim_{jk} \right|} \qquad 21$$

where k=1,2,…and l is the total number of items taken from the neighborhood.

2.11.2 CONTENT-BASED RECOMMENDER SYSTEMS

Content-based recommendation systems or content-based filtering is based on textual information such as documents. These items are typically described with weights and keywords. Using nearest neighbor functions or clustering methods can allow the recommendation system to analyze the keywords, document content and then use it as a basis to recommend a suitable item. This will also be based on the items characteristics. The different techniques used in this method of information filtering follow two approaches: Heuristic-based and model-based methods. Heuristic based methods include KNN algorithms and clustering methods while model-based methods include Bayesian filtering, artificial neural network, and clustering. Content based filtering systems are limited by their content. If they lack sufficient keywords or overspecialization problems, they yield weak recommendations (Shishehchi et al., 2011).

2.12 CHAPTER SUMMARY

In this chapter the literature on mobile devices was reviewed on how they have increased popularity and how their use has rapidly changed over the year. Literature on recommender systems, domains in which recommender systems have been employed and how they have been employed has been explored. Furthermore, an intensive exploration of challenges that are facing recommender systems daily has been explored in this chapter. Recommender system challenges revealed that collaborative filtering systems are impacted by cold; however, by implementing a hybrid it is possible to overcome the challenges. Different approaches, namely, collaborative filtering, content-based approaches were discussed.

Different algorithms were discussed in this chapter. The chapter also outlined how trust is relevant in recommender systems and how it plays a role in the use of recommender systems. This chapter also highlighted different similarity measures and how they are relevant in the study. Having studied the different approaches, hybrid algorithms were also explored. The next chapter focuses on clustering algorithms as well as classification algorithms.

CHAPTER 3

3 DATA CLUSTERING, CLASSIFICATION AND RECOMMENDER SYSTEMS EVALUATIONS

3.1 CO-CLUSTERING

Co-clustering is mostly used because of its low computational cost as compared to other collaborative filtering approaches. It is evident from Banerjee et al. (2007) that co-clustering of rows and columns has rapidly become a powerful data analysis technique. Co-clustering is also known as bi-clustering (Cheng and Church, 2000). Clustering is a learning technique that does not need to be supervised while it learns the latent structure of the data matrices. In this case the data matrices are of two objects, namely, subscribers and products in literature. A large number of clustering algorithms such as *k-means* and agglomerative clustering have been studied (Gosh, 2003). It can be said that co-clustering has received recognition and attention in practical applications such as simultaneous clustering of documents in text mining (Takamura and Matsumoto, 2003). In the document retrieval and search engines co-clustering has been used to group documents that contain the same words, enhancing the retrieval rate and accuracy of the retrieval system.

Collaborative filtering using co-clustering

In this co-clustering approach, there are three main steps that needs to be taken into consideration.

1. Calculating the average of the matrix
2. Assigning the row cluster
3. Obtaining the row cluster.

3.2 CLUSTERING

The choice of clustering similarity metric is critical for training high-quality clustering solutions. In domains where more than one similarity metric are appropriate, several approaches have been proposed for combining multiple similarities using machine learning techniques (Cohen and Richman, 2002; Bilenko and Mooney, 2003; Bilenko et al., 2005; Chen et al., 2009). Other metric learning approaches use optimization techniques to learn a similarity metric from labeled examples

directly (Xing et al., 2003; Davis et al., 2007). Clustering is a procedure of information mining in which comparable objects are assembled into clusters. Clustering methods/systems are broadly implemented in various fields like information retrieval, image processing, etc. (Mahmud et al., 2012). There are two sorts of methodologies in clustering: various leveled also known as hierarchical and partitioning. In various leveled, the clusters are consolidated in light of their nearness or how close they are. This blend is anticipated when additional process prompts unwanted clusters. In partition clustering approach, one dataset is isolated into clear number of little sets in a solitary emphasis (Dunham, 2006). The exactness and nature of clustering depends on how the calculations are actualized and their capacity to discover concealed information.

Cluster analysis (Xia and Xi, 2007) groups objects (perceptions, events) based on the information found in the data portraying the items or their connections. The objective is that the items in a group shall be similar to one other and unique in relation to (or inconsequential to) the objects in other groups. The more prominent the resemblance (or homogeneity) inside a group, and the more noteworthy the uniqueness between groups, the better or more particular the clustering. Knowledge discovery is broadly utilized as a part of different domains like retail and telecommunication industry, fraud detection, spatial data analysis and other scientific applications (Virmani et al., 2015).

An imperative inquiry is how to choose what constitutes great clustering, since it is normally agreed that there is no outright 'best' set of rules which would be applied in order to come up with the best way of clustering (Shukla et al., 2012; Singh et al., 2008). Therefore, the client must supply the foundation that best suits their specific needs, and the consequence of the clustering calculation can be deciphered in various ways.

3.1 CLUSTERING ALGORITHMS

3.1.1 K-MEANS

The *K-means* algorithm is generally picked over other clustering algorithms as it is extremely proficient in preparing vast/large datasets. It often terminates at a local optimum and generates tighter clusters than hierarchical clustering, especially if clusters are globular. It is a famous algorithm because of its detectable speed and effortlessness. *k-means* utilizes the idea of Euclidean distance to figure the centroids of the clusters. This strategy is less successful when new

informational indexes are included and have no impact on the deliberate separation between different information objects. The computational density of *k-means* algorithm is high (Mahmud et al., 2012; Sharma et al., 2012). Similarly, *k-means* is incapable of handling noisy data and absent values. To overcome this, data pre-processing is necessary to ensure that the data is clean and does not have noise. Normalization is used to eradicate laid off data and ensures that decent quality clusters are produced which can improve the efficacy of clustering algorithms. So it becomes an essential step before clustering because Euclidean distance is highly sensitive to the changes in the differences (Patel and Mehta, 2011). Amid flat clustering methods, *k-means* is the most broadly used and is considered the most significant algorithm (Schütze et al., 2008). The algorithm attempts to classify all elements into *k* clusters by lessening the average squared Euclidean distance to the closest centroid.

This measure effectively calculates the distance between two elements in Euclidean space by calculating an absolute difference between the elements for every dimension:

$$\left|\vec{x}-\vec{y}\right| = \sqrt{\sum_{i=1}^{M}(x_i - y_i)^2} \qquad 22$$

Initially, the *k* centroids are positioned arbitrarily, or according to some rule of parting. Over several reiterations, each element is then assigned to a centroid, or cluster, and a new position for each centroid is calculated. The algorithm is complete after *N* reiterations. This is the point where the centroids do not move, or when the algorithm falls below a certain threshold of improvement.

The complexity of the *k-means* clustering algorithm is O(*IKNM*), where *I* is the fixed amount of repetitions, *K* the number of clusters, *N* the number of elements, and *M* the dimensionality of the data.

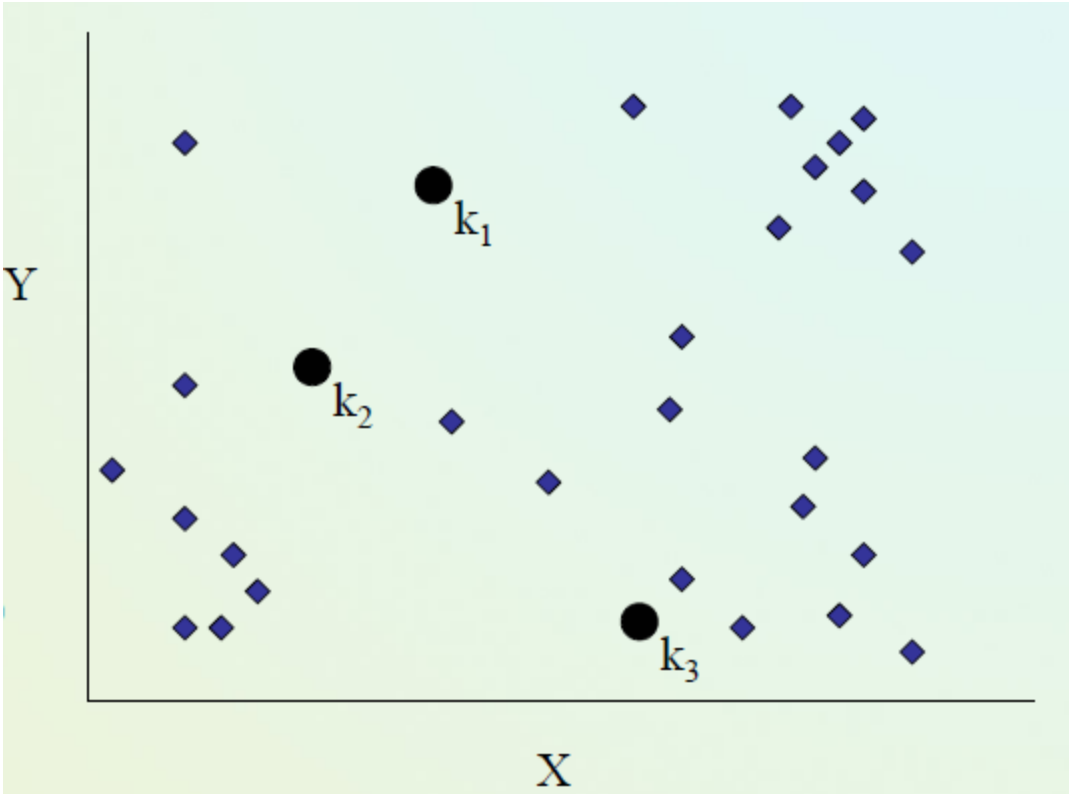

Figure 5. K-means example representing cluster centroids

Figure 5. Shows how the *k-means* algorithm works according to the above image k=3 and the black circles show centroids while the square boxes depict the clustered instances.

3.1.2 EM CLUSTERING

Like the *k-means* clustering method, EM for Gaussian Mixtures needs the number of clusters (*k*) to be stated. The covariance matrices are each set as the identical sloping matrix, where the j^{th} element on the slope is equal to the variance of the j^{th} dimension of the entire dataset. Likelihood circulations for cluster assignments are computed for every single data point, based on the initial parameter approximations (means and covariance matrices). Using these likelihood circulations, EM finds the probable log-likelihood function and computes new parameter estimate that maximizes that log-likelihood function. With the new parameter estimates, new probability distributions for cluster assignments are calculated and therefore a new expected log-likelihood function is derived. New parameter estimates are once again obtained by maximizing the new expected log-likelihood function. This is iterated until the enhancement in the parameter estimates is reduced to a point whereby it falls below a certain threshold (Sharma and Sharma, n.d).

3.2 CLASSIFICATION ALGORITHMS

Classification complications target to find common characteristics that specify the group to which each instance fits. This can be utilized both to recognize the existing data and to predict how new cases will perform. Data mining produces classification models by examining the data that is already classified and inductively discovering a predictive pattern. The existing cases may be derived from ancient databases. They may also be a result of an experiment in which a model of the entire database is tested in the real world and the fallouts used to design a classifier. Sometimes an expert is required to classify a sample of the database, and that sample is used to create the model which will be applied to the entire database (Rajput et al., 2011). Different classification algorithms are applied and used with different datasets, some of these algorithms are discussed below:

3.2.1 MULTI-LAYER PERCEPTRON

A Multi-layer Perceptron (MLP) is a class of feedforward artificial neural network. MLP consists of at least three layers of nodes. Except for the input nodes, each node is a neuron that uses a nonlinear activation function. MLP utilizes a supervised learning technique called back-propagation for training. It can distinguish data that is not linearly separable. Multilayer perceptrons are sometimes colloquially referred to as "vanilla" neural networks, especially when they have a single hidden layer (Hastie et al., 2009).

Artificial Neural Network (ANN) is a unified group of nodes by means of mathematical approaches to process information. It is a self-adaptive system, which can change its construction based on the internal or external influences. Multiple ANN models have been developed and the most prevalent one is the Multi-Layer Perceptron (MLP) feed forward network (Kavzoglu and Mather, 2003). MLP consists of many layers. The most widely used structure was the three-layer structure, due to its capability to solve most image classification problems. The multiple layers include one input layer, one hidden layer, and one output layer as on shown in Figure 6.

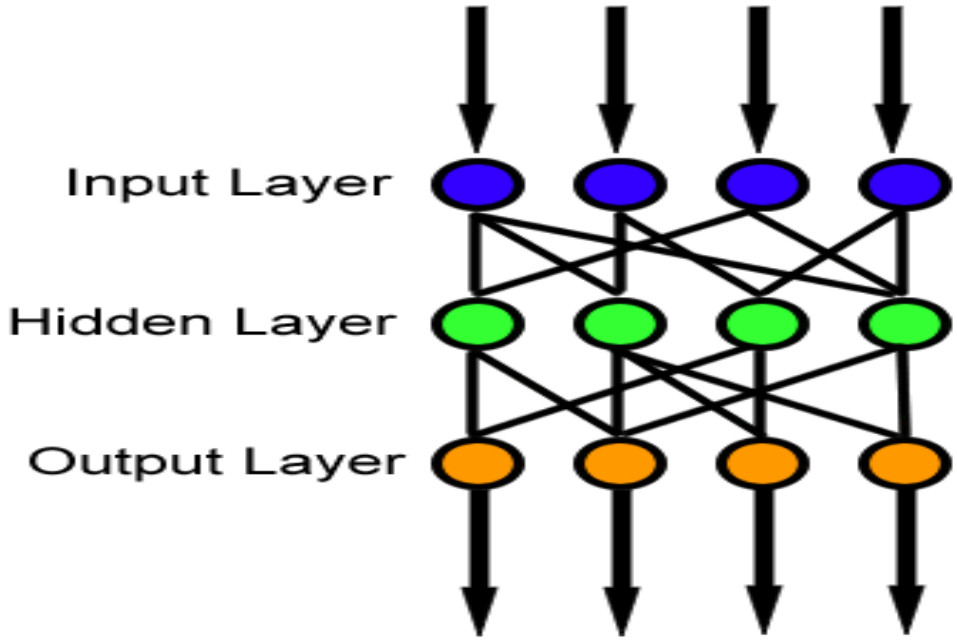

Figure 6. Illustration of how a MLP nodes look like.

Each layer is composed of artificial neurons. It is visible in Figure 6 that all the nodes are linked with each other, excluding the nodes in the same layer. The input layer, the hidden layer and the output layer are used for data input, processing, and output, respectively. The downside of this algorithm is that creating a neural network is time consuming.

3.2.2 LOGISTIC REGRESSION

Logistic regression, by use of a linear combination of independent variables, is a statistical technique used to predict the possibility of occurrence of an event, i.e. its probability (Jung et al., 2014). However, it is evident that the algorithm yields low accuracy with high-processing speed, indicating that classification using only a logistic algorithm cannot guarantee the accuracy of the results (Jung et al., 2014). For this purpose, logistic algorithms need to be used in conjunction with other algorithms to validate the results.

3.2.3 JRIP

JRip also known as (RIPPER) is one of the straightforward and most popular algorithms. It examines classes in increasing size and it generates an initial rule set using incremental reduced error. JRip proceeds by treating all the samples of a specific ruling in the training data as a class,

thus discovering a set of rules that conceal all the members of the class. This process is iterated until all classes have been covered (Rajput et al., 2011).

3.2.4 J48

J48 is a tree classifier. A tree is moreover a leaf node labeled with a class, or a structure containing a test, then linked to two or more nodes also known as subtrees (Shepperd and Kadoda, 2001). To classify some instance, first there is a need to identify its attribute-vector and apply this vector to the tree. The tests are done on the attributes, reaching one or other leaf, to complete the classification process, as illustrated in Figure 7.

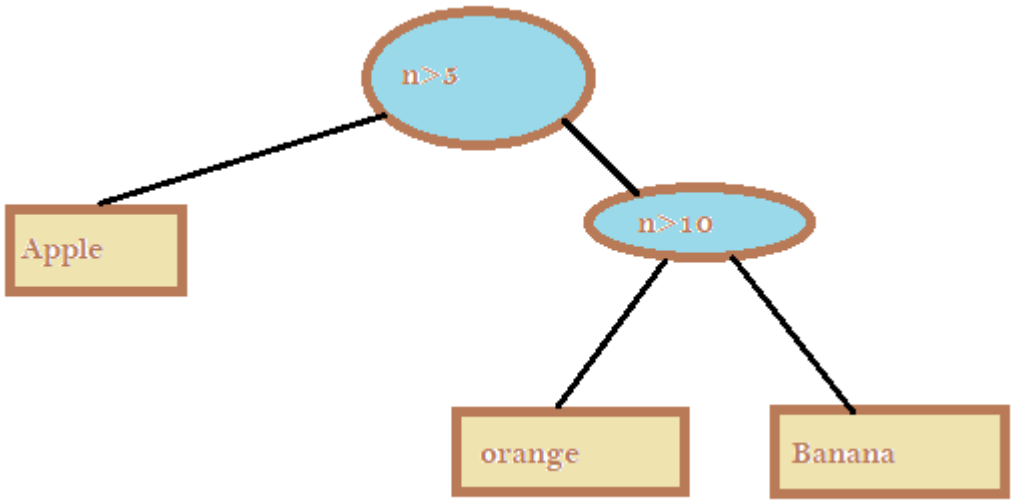

Figure 7. Simple example of a tree classifier classification process.

Let us consider this example above and

let n=5

Then n>5 = true

n>10 =true

therefore, the above will be classified as a banana.

3.3 EVALUATING RECOMMENDER SYSTEMS

Evaluation is an exceptionally significant and monotonous task in data recovery or information retrieval. There are numerous retrieval models, systems and algorithms in writing so to be able to select the best among many, and enhance there is a need to assess them. One approach to assess is to gauge the viability of the algorithms (Zuva, 2012).

There are many ways to evaluate recommender systems and this part is a discussions about ways in which different recommender systems are assessed and the types of results produced by each evaluation method. It is evident that the amount of information in the world accessible to human beings has increased far more quicker than anyone can ever be able to process it, thus requiring the employment of strategic calculations on how information is filtered to narrow it down and how it is made accessible to people in a more eloquent way (Sarwar et al., 2001a). Massa and Avesani (2007) discovered that the most widespread and widely used technique for evaluating Recommender Systems is based on a technique called leave-one-out. Leave-one-out is an offline technique that can be run on a historical dataset and encompasses concealing one rating and then trying to predict it with a specific algorithm

3.3.1 ACCURACY IN RS

There are multiple ways to evaluate the predictive quality of recommender systems. However, accuracy is one of the most commonly measured metrics. Accuracy is a measure of how effective the system's predictions are to real results. A communal way to measure accuracy is by using a statistical accuracy metric called Mean Accuracy Error (MAE) (Herlocker et al., 2004).

3.3.1.1 MEAN ABSOLUTE ERROR

MAE is measured for items that the user rated and it can be computed using equation 23 (Vozalis and Margaritis, 2003).

$$MAE_i = \frac{\sum_{j=1}^{n_i} \left| ar_{ij} - r_{ij} \right|}{n_i} \qquad 23$$

where n_i is the number of items the user has rated r_{ij} is the representation of predictions generated for the chosen items and ar_{ij} represents the actual ratings provided by the user for those chosen items.

It can be said that in RS research, the use of several types of procedures for evaluating the quality of a recommender system have been adopted and used and can be mainly categorized into two classes (Ziegler et al., 2005).

Statistical accuracy metrics: these metrics are used to evaluate the accuracy of a system by comparing the numerical recommendation scores against the actual user ratings. Mean Absolute Error (MAE) between ratings and predictions is a widely used metric. MAE is a measure of the deviation of recommendations from their true user-specified values. For each ratings-prediction pair $< p_i, q_i >$ this metric treats the absolute error between them, i.e., $|p_i - q_i|$ equally. The MAE is computed by the first summing these absolute errors of the N corresponding ratings-prediction pairs and then computing the average. Formally MAE is calculated as:

$$MAE = \frac{\sum_{i=1}^{N} |p_i - q_i|}{N} \qquad 24$$

The lower the MAE, the more accurately the recommendation engine predicts user ratings. Root Mean Squared Error (RMSE), and correlation are also used as statistical accuracy metric.

Decision support accuracy metrics evaluate how effective a prediction engine is at helping a user select high-quality items from the set of all items. These metrics assume the prediction process as a binary operation-either items are predicted (good) or not (bad). With this observation, whether an item has a prediction score of 1.5 or a five-point scale is irrelevant if the user only chooses to consider predictions of 4 or higher.

Another way to quantity unbiased accuracy is by computing the R-score which is a measure based on the assumption that the value of a recommendation declines exponentially with the position of an item. The score for a user *u,* choosing and item *i* at position j is computed as follows (Myrberg, 2016).

$$Ru = \sum_{u} \sum_{j} \frac{\max(r_{uij} - d, 0)}{2^{(j-1)/(\alpha-1)}} \quad 25$$

where r_{ui} refers to the rating of a user u for item i. Here, the rating r_{ui} is 1 if the user selected the item and 0 if not. A higher R-score refers to a better ranking of the item. d is a task-dependent neutral rating and α is a half-life parameter which controls the exponential decline of the rating value.

All recommender systems, whether they are in the e-commerce world or any other application have these attributes in common according to Vozalis and Margaritis (2003):

1. They all consume inputs,
2. They all have a goal and,
3. They all produce an output.

There is significantly no best algorithm and every algorithm depends on the purpose and the nature of the recommender system (Herlocker 2004). Identifying the best algorithm for a given purpose has proven challenging, in part because researchers disagree on which attributes should be measured, and which metrics should be used for each attribute (Herlocker, 2004). According to Moreira et al. (2015) offline evaluation is usually done by recording the items users have interacted with, hiding some of this user-item interaction and training algorithms on the remaining information to assess the accuracy.

The accuracy of a recommender system cannot be based on the manipulation, whether it is online or offline. Whether a recommender system is accurate or not depends on the recommendation algorithm and it can be argued that the correct selection of the recommendation variables is essential for every kind of recommender system. The recommendation accuracy can be measured by how often a user accepts a recommendation; however, the disadvantage is that if data is studied offline the accuracy cannot be measured against the user's immediate response but the test set (Schafer et al. 2007).

It is evident from Beel et al. (2013) that offline evaluations are the most common evaluation methods for research paper recommender systems. This does not validate the accuracy, nor does

it standardize offline evaluation of recommender system data. The study of Beel et al. (2013) is contradictory for the offline and online evaluation. Beel et al. (2013) argued that offline evaluations are meant to identify the most promising recommendation approaches. These most promising approaches should then be evaluated in more detail with a user study or an online evaluation to identify the best approaches.

Although there are several published evaluations of recommender systems, most of them measure accuracy. Researchers have discovered that accuracy is not the only criteria of interest and in other cases it may be the least important measure. Below are some of the evaluations that can be considered besides measuring accuracy.

3.3.2 LEARNING RATE

This is a measure of how fast the CF system can become effective in predicting the taste of users as new data comes in. Under normal circumstances these are calculated per-user, as a measure of the number of ratings that a user has to provide prior to getting high quality personalized predictions (Schein et al., 2001).

3.3.3 COVERAGE

Coverage is the percentage of items/users in which the system can make predictions. It is also possible to calculate variations such as the percentage of items that have the potential of being recommended to users, since performance optimizations in recommendations could prevent certain items from not being recommended (Sarwar et al., 2000).

3.3.4 USER SATISFACTION METRICS.

The above metrics are only a sample. However, there are many more metrics that can be calculated provided that the researchers could present a system to users, and measure how users see the system. This is achievable by surveying the users. Good examples include that by Swearingen and Sinha (2001).

3.4 EVALUATING UNRANKED RECOMMENDER SYSTEM RETRIEVAL RESULTS

The most commonly used and important elementary measures for information retrieval effectiveness are precision and recall. Precision can be defined as the portion of recovered items

that are relevant to all recovered items or the prospect given that an item is recovered it will be pertinent and recall can be defined as the fraction of relevant items that are retrieved to relevant items in the entire database (Manning et al., 2008). The value of recall to users indicates the ability of the system to find relevant items and precision indicates the ability to output top ranked relevant items. In the real world, the user of a system is concerned about relevant items retrieved. Thus the measures of precision and recall focuses on the evaluation of the relevant output of the system. Low and high values would indicate bad and good performance of the system, respectively. Under normal circumstances, as the number of retrieved items increases the precision decreases and recall increases.

There are other measures that are derived from them. F-measure is also a known measure derived from precision and recall measures. This is a scalar amount that trade off precision against recall which is the weighted harmonic mean of precision and recall (Zhou and Yao, 2010). More information on F-measure can be obtained in Powers (2011).

3.4.1.1 EVALUATION OF RANKED RETRIEVAL RESULTS

This section describes techniques for evaluation of ranked information retrieval results that use precision and/or recall measures. Some of these techniques include Precision-Recall curve (P-R-curve), R-precision, Mean Average Precision (MAP) and Precision at ***k.*** Most current systems present ranked results and therefore to use the precision and recall measures there is need to pair them at each given position. Precision at ***k*** and R-precision can be used. Precision at ***k*** reduced as **P@k** is the precision calculated at a cut-off point k. This measure does not take into consideration recall. It is criticized because relevance items for a query partake a ration of influence on **P@k** but are ignored. To improve this issue R-precision measure was introduced. In this measure the number of relevant items is known and that becomes the cut-off point. The formula is given in equation 26 below:

$$R-\Pr ecision=\frac{1}{n(\mathrm{Re})}\sum_{k}^{n(\mathrm{Re})}\mathrm{Re}_{k} \qquad 26$$

The Receiver Operating Characteristics (ROC) curve is also useful in information retrieval systems for performance evaluation. The ROC curve always moves from the bottom left to the top right of the graph. Performance of a model is represented as a point in ROC curve. A good system produces results that generate a graph that climbs steeply on the lift side.

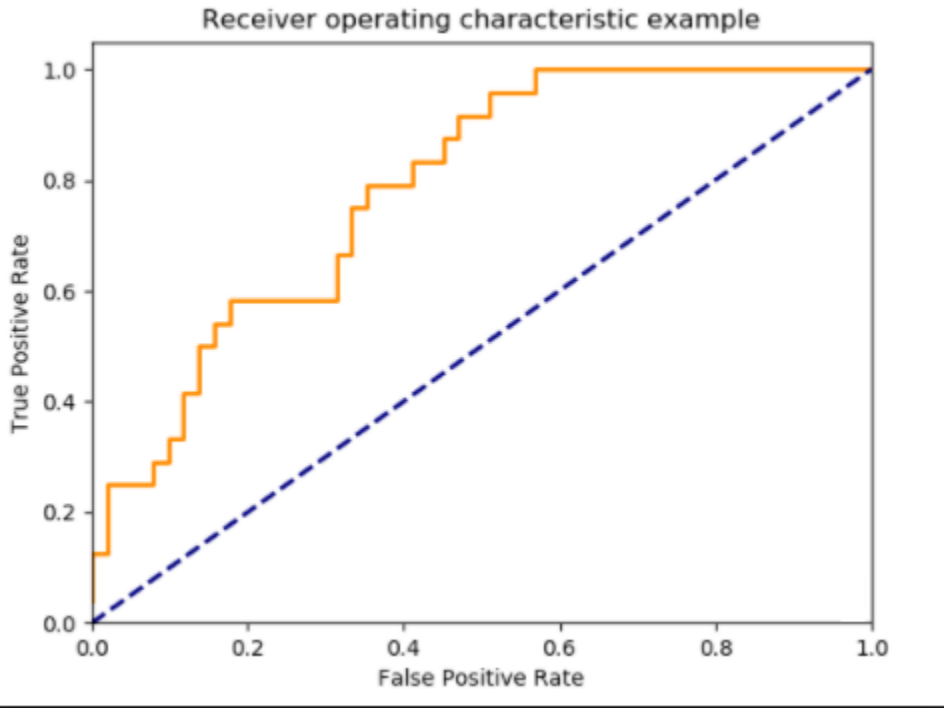

Figure 8. graph illustrating Receiver operating characteristic example

Non-graphical evaluation techniques related to precision and/or recall include MAP which has gained popularity among the Text Retrieval Conference (TREC) members (Manning *et al.*, 2008). MAP is one of the various ways of combining precision and recall into a single scalar value measure which is defined as an average of the average precision value for a set of queries.

$$MAP = \frac{1}{n(\mathrm{Re})} \sum_{k=1} \mathrm{Re}_k \frac{\sum_{i=1}^{k=1} \mathrm{Re}_i}{k}$$

27

where *n(Re)* is the number of relevant items, Re_k and Re_i take 0 or 1 indicating not relevant or relevant at position *k* and *i*, respectively.

3.5 CHAPTER SUMMARY

In this chapter various clustering methods, including co-clustering and how it applies for rows and columns have been discussed. An exploration of the hierarchical and partitional algorithms has been done and it was shown that *k-means* is the simplest partitional algorithm indicating the reason for wide use. The clustering algorithms including *J48* and JRip (RIPPER) have been discussed. These classification algorithms perform differently but they are all used for the sole purpose of classification.

Furthermore, in this chapter a brief discussion of how clustering is different from classification was provided. Visual illustrations and an example of how a tree classifier works was also provided in this chapter. Lastly, literature on evaluation methods for recommender systems unranked and ranked results evaluations were discussed including R-precision. Graphical representations of ROC in its acceptable state was illustrated. The next chapter the methodology has been discussed and how this study has used *J48* together with *k-means* in preparation for the recommendation system.

CHAPTER 4

4 RESEARCH METHODOLOGY

4.1 INTRODUCTION

In this section, an introduction of the research methodology and the techniques that are used in recommender systems is done. Research methodology is a methodical analysis of the procedures applied to a field of study. It encompasses the notional analysis of the methods and principles associated with a knowledge body. Archetypally, it can be classified into quantitative or qualitative techniques (Ishak and Alias, 2005). This study follows a qualitative approach whereby literature exploration, development of the recommender systems and evaluation was done. The recommender system in this study can be viewed as a client-server architecture, and therefore the system was evaluated both on the client-side as well as on the server-side.

Recommender systems can be classified into four distinct categories, that is, content based, collaborative filtering, knowledge based and hybrid. Hybrid systems employ two or more recommendation methods. In this study, a hybrid system has been employed which includes collaborative filtering for usage patterns and subscriber classification as well as content based by using subscriber usage history. In application, content-based filtering systems can be implemented in two diverse ways or grouped in two different classes, namely, memory based, and model-based algorithms as described in chapter 2. For this project, model-based algorithms have been used because they reduce runtime complexities. Subscriber nearest neighbor algorithms have been used to generate recommendations. Based on the subscriber usage patterns and cluster assignments the nearest neighbor has been used to obtain suitable recommendations.

This study follows the steps as illustrated in Figure 9. The data was collected, then converted into the correct format, after which it was pre-processed. After completion of the above steps the data was then classified, and prediction took place. The data was then loaded into the recommender system and the recommendations computed. After recommending to the subscriber the last steps involve evaluating the system and analyzing the results.

Figure 9. Illustration of how the methodology flows

4.2 HYBRID RECOMMENDER SYSTEMS

Hybrid recommender systems employ more than one technique because deficiencies of one method can overcome with the strength of the other. It is evident from Burke (2002) that the most commonly used collaborative filtering is combined with some other technique in an attempt to avoid the ramp-up problem. Some of the hybrid combinations include weighted, switching, mixed, feature combination, cascade, meta-level etc. Mixed recommender systems represent

recommendations from different recommenders simultaneously. Feature combination combines features from different recommendation data sources into a single recommendation algorithm. Meta-level uses models learnt by another recommender system an input to the next one. In this study, two separate sets of recommendations were generated and merged them together to develop a final set of recommendations.

4.2.1 COLLABORATIVE FILTERING IN SUBSCRIBER RECOMMENDER SYSTEMS

Mobile subscriber data was collected from the OPCO and used in experiments to predict the future usage patterns and recommendations. The system performance results were evaluated using precision and recall. A comparison was done between the recommendation generated and the usage patterns of the subscribers. Mobile operators have Base Transceiver Stations (BTS) located at different areas to collect subscriber data. The BTS sends data wirelessly to different systems and data is collected from the BTS through these systems. Subscribers are grouped through their usage patterns.

4.2.1.1 MEASURING USER SIMILARITY

The subscribers have been clustered according to how they use certain items, that is, by checking how they use DATA, VOICE and SMS. In this case a utility matrix that was proposed in Table 10 has been used. The focus is to see how the users are using these categories and compare them with a user that is using a postpaid item. The system is made a recommender system by using the data from *weka* and predicting what the subscriber usage for a certain item would be then recommend a product based on the predicted usage similarity.

After correctly classifying the subscribers as well as obtaining satisfactory results on all datasets, the data from weka was then used as the source data to simulate the recommender system. Using the results from weka *J48* classification algorithms, the recommender system was able to recommend the products to the subscribers based on the classification as well as the usage rating.

4.2.1.2 SUBSCRIBER BASED NEAREST NEIGHBOUR

In subscriber based nearest neighbor collaborative filtering algorithm, the prediction of a possible contract is based on the usage patterns of the subscriber in comparison to the contract users. There

must be similarity between the two subscribers for a recommendation to be possible. The crucial part of subscriber nearest neighbor will be that the subscriber is to identify the nearest neighbor of the target subscriber. The formula for Pearson correlation coefficient similarity is given in Equation 28.

$$Subscribersim(u_t, u) = \frac{\sum_{i \varepsilon I_{u,ut}} (R_{u,i} - \bar{R}_u)(R_{u_t i} - \bar{R}_{u_t})}{\sqrt{(R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{i \varepsilon I_{u,ut}} (R_{u_t,i} - \bar{R}_{u_t})^2}} \quad 28$$

where $Subscribersim(u_t, u)$ is a representation of the similarity between subscriber u and u_t, $I_{uu_t} = I(u) \cap I(u_t)$ means the product is used by both the subscribers and u , u_t , and $R_{u,i}$ are the usage totals for product i used by subscriber u and u_t respectively, R_u and R_{u_t} represents the average usage amounts over a time of users u and u_t, respectively.

4.2.1.3 PRODUCT NEAREST NEIGHBOUR

Product based nearest neighbor algorithms works the same as subscriber based nearest neighbor; however, the difference lies significantly on the product. The recommendation is made purely on the basis of the product nearest neighbor. The similarity is calculated based on the product that is common between two different subscribers. The formula for adjusted based cosine which is mostly used is given in equation 29 below:

$$productsim(i_t, j) = \frac{\sum_{u \varepsilon U_{t,j}} (R_{u,i} - \bar{R}_u)(R_{u_t j} - \bar{R}_{u_t})}{\sqrt{\sum_{u \varepsilon U_{t,j}} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \varepsilon U_{t,j}} (R_{u,j} - \bar{R}_u)^2}} \quad 29$$

where R_u and $R_{u,j}$ represents the usage amounts of subscriber u on product j. $\bar{R}_u$ is the mean of the u^{th} subscriber's usage amount over a certain time and $u_{i,j}$ represents all the subscribers that have used product i and j.

The prediction for product based nearest neighbor algorithm for subscribers is carried out using equation 30 below.

$$P_{\substack{(u,j) \\ product-based}} = \frac{\sum_{i \varepsilon R_{ut}} prodsim(i,j) * R_{ut,j}}{\sum_{i \varepsilon R_{ut}} prodsim(i,j)} \qquad 30$$

Where $prodsim(i,j)$ is the product similarity between product i and product j and $R_{ut,j}$ is the usage amount of user *ut* on product *j*.

If the predicted future usage amount is within the product range, then the product is recommended to the subscriber.

4.2.2 TIME SERIES

An element of any product and any Subscriber is the time of the day at which the Subscriber interacts with the mobile device. Different pricing options are applied to prepaid users depending on the time of the day that a transaction/interaction occurs. A method to evaluate usage based on time is thus derived. As part of the methodology a measure when most of the transactions are performed is computed and come up with a time series of events for a postpaid subscriber as well as for a prepaid subscriber. The time series then becomes a crucial factor of the similarity, in such a way that similarity can mean that the two or more subscribers have similar usage patterns at similar times of the day. For example subscriber A and subscriber B can both make more calls during off peak hours (peak and off-peak hours) are found in Table 5.

Table 5. Illustration of peak and off-peak times

Period	**Time**	**Day**
Peak period	07h00-20h00	Monday-Friday
Off-Peak period	20h00-07h00	Monday-Friday
Off-Peak period	00h00-24h00	Saturday, Sunday and public holidays

To obtain a time series, the usage data is divided into different time intervals and for each interval the usage totals are computed for each stream, be it DATA, VOICE or SMS. An analysis of how similar the usage patterns are at different time intervals was done.

4.2.2.1 USING TIME INFORMATION IN RECOMMENDER SYSTEMS

Having time stamped usage data brings in the prospect to check and verify at what time of the day most usage happens and verifies that the prescribed product is suitable with the usage times of the subscriber. It is evident from Min and Han (2005) that using the ratings of a user on products of a particular category enables one to compute ratings value on the category; thus in this instance the usage on a specific category will average the usage on that category as shown in equation 31.

$$cat\_r_{u,cat} = \sum_{i \in cat} \frac{r_{u,i}}{|r_{u,i}|} \qquad 31$$

where $cat\_r_{u,cat}$ is the categorical usage of a postpaid subscriber to the product category cat. In order to obtain a time series, the usage data are divided into different time intervals, and for each interval the category usage averages are computed. As soon as a time series is constructed, many of the existing methods of time series analysis may be used. Thus this study incorporated the method proposed in Min and Han (2005) which identifies the moment when content drift occurs by analyzing how similar are the category usage at different time intervals. For this purpose, a

Pearson correlation coefficient on category usage of the same subscriber but on different time intervals is calculated as shown in equation 32.

$$AS(u,t_i,t_j)=\frac{\sum_{cat}(cat\_r_{u,cat,ti}-\overline{cat\_r_{u,ti}})(cat\_r_{u,cat,tj}-\overline{cat\_r_{u,tj}})}{\sqrt{\sum cat(cat\_r_{u,cat,t_i}-\overline{cat\_r_{u,ti}})^2}\times\sqrt{\sum cat(cat\_r_{u,cat,t_j}-\overline{cat\_r_{u,tj}})^2}} \quad 32$$

where $AS(u,t_i,t_j)$ is the auto similarity of subscribers of u between time intervals t_i and t_j, $cat\_r_{u,cat,t}$ is the category rating of user u on category cat during time interval t. Given a similarity threshold value $tsim$, if $AS(u,t_i,t_j) < tsim$, it can be stated that subscriber u changed his/her usage at time interval t.

Equation 33 has been used to obtain the trend equation using regression analysis.

$$y=a+bx \quad 33$$

b can be calculated as follows

$$b=\frac{n\sum xy-\sum x\sum y}{n\sum x^2-(\sum x)^2} \quad 34$$

where n is the sample size and $\sum xy$ is the sum of xy

a is calculated in equation 35

$$a=\frac{\sum y}{n}-b(\frac{\sum x}{n}) \quad 35$$

where $\frac{\sum y}{n}$ is the mean of y and $\frac{\sum x}{n}$ is the mean of x. Therefore the above equation can be represented as $a=\dot{y}-b\dot{x}$

where x is the instance number and y is the usage amount.

4.3 CONTENT BASED FILTERING

Content based filtering approaches recommend products to subscribers based on the content within the data. For subscribers, the usage content is compared with how the postpaid user content has been and the recommendation is based on sorely on that content. The postpaid data is similar to the prepaid data in content; however, for the products there are dissimilarities. A product may have minutes and data but no airtime (monetary value) and another product may have all three components. All products are represented by attributes profile, including whether the product has voice minutes, data-bundles, SMSs etc. The formula for cosine similarity is given in 36.

$$sim(x, y) = \frac{\sum x_i * y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \quad 36$$

where x and y are product vectors containing n elements, and $sim(x, y)$ measures the distance closeness.

With Euclidean dissimilarity, the distance between the products based on how different or dissimilar the products are can be obtained. The formula for Euclidean dissimilarity is given in equation 37.

$$dissim(x-y) = \sqrt{\sum_{i=1}^{n}(x_i - y_i)^2} = \|x-y\|_2 \quad 37$$

where x and y are product vectors of n elements and measures the distance apart.

This approach is known to be suitable with different domains, and thus works well in the mobile subscriber domain as well. These approaches are information retrieval based.

4.4 DATA CLASSIFICATION

Data classification is a form of collaborative filtering. To fulfill the objectives of this study the algorithms used were able to classify the different datasets. *J48* algorithm is used due to its consistency and has been demonstrated by (Jung et al., 2014).

4.4.1 DATASETS

After data cleaning and pre-processing, there were four different datasets. The original dataset from the OPCO had many empty columns that were unnecessary, and it also included keys which had to be changed into meaningful information and descriptions. The mobile subscriber data comes in different forms streams, although it is usually combined by OPCOs. For revenue reporting, the data still needs to be analyzed in isolation before combining them. This is necessary because although the streams are all revenue generating, they are different in terms of usage. The datasets are described in Table 6.

4.4.1.1 VOICE DATASET

The voice dataset contained the following parameters/attributes: subscriber, minutes, average minutes/day, group, deviation to average, usage group, usage Rating. Table 6 has details about all the attributes of the voice dataset.

Table 6. Voice dataset, parameter explanations.

Parameter	**Description**
Subscriber	A subscriber as defined on the system
Minutes	The number of minutes
average minutes/day	Average minutes per day
group	The group number in which the subscriber is categorized from the source data
deviation to average	The difference between the group average and the subscriber minutes
usage group	The usage classification for the subscriber
Usage Rating	The rating on a scale of 1-5 as per the usage matrix defined.

Figure 10 shows a sample of the data that was used as part of the training set after some of the attributes were removed using the weka remove function.

```
@relation 'Voice dataset-weka.filters.unsupervised.attribute.Remove-R5'

@attribute subscriber numeric
@attribute minutes numeric
@attribute 'average minutes/day'
@attribute group numeric
@attribute 'deviation to average' numeric
@attribute 'usage group' {'High Usage','Moderate Usage','Low Usage'}
@attribute 'Usage Rating' numeric {1,2,3,5,5}

@data
224660000000,600,42:51:26,1,179.4999,'High Usage',5
224660000400,600,42:51:26,1,179.5,'High Usage',5
224660001500,429,30:38:34,1,8.5,'High Usage',4
224660002300,365,26:04:17,1,55.5,'High Usage',4
224660004000,288,20:34:17,1,132.5,'High Usage',2
224660004200,288,20:34:17,1,132.5,'High Usage',2
224660004900,199,14:12:51,2,8.8,'Moderate Usage',1
224660005300,199,14:12:51,2,8.8,'Moderate Usage',1
224660005800,199,14:12:51,2,8.8,'Moderate Usage',1
224660008300,199,14:12:51,2,8.8,'Moderate Usage',1
224660008400,199,14:12:51,2,8.8,'Moderate Usage',1
224660008800,189,13:30:00,2,18.8,'Moderate Usage',1
224660010900,187,13:21:26,2,20.8,'Moderate Usage',1
224660011100,185,13:12:51,2,22.8,'Moderate Usage',1
224660011400,180,12:51:26,2,27.8,'Low Usage',1
224660012000,120,8:34:17,3,38.4,'Low Usage',1
224660012200,120,8:34:17,3,38.4,'Low Usage',1
224660012500,120,8:34:17,3,38.4,'Low Usage',1
224660012700,120,8:34:17,3,38.4,'Low Usage',1
224660016700,120,8:34:17,3,38.4,'Low Usage',1
224660018500,120,8:34:17,3,38.4,'Low Usage',1
```

Figure 10. Part of the voice subscriber dataset used as training data for the J48 algorithm.

4.4.1.2 SMS DATASET

The SMS dataset contained the following parameters/attributes: subscriber, NO_SMS, average SMS per day, group, deviation to average, Group AVG, usage group, Usage Rating. Table 7 shows the details about all the attributes.

Table 7. Parameter explanations on SMS dataset

Parameter	**Description**
Subscriber	A subscriber as defined on the system
NO_SMS	Total number of SMSs
average SMS per day	Average number of SMSs per day
Group	The group number in which the subscriber is categorized from the source data
deviation to average	The difference between the group average and the subscriber minutes
usage group	The usage classification for the subscriber
Usage Rating	The rating on a scale of 1-5 as per the usage matrix defined.
Group average	The average number of SMSs sent from the specific group

```
@relation 'sms dataset'

@attribute Subscriber numeric
@attribute NO_SMS numeric
@attribute 'average SMS per day' numeric
@attribute group numeric
@attribute 'deviation to average' numeric
@attribute 'Group AVG' numeric
@attribute 'usage group' {'High Usage','Low Usage','Moderate Usage'}
@attribute 'Usage Rating' numeric

@data
224660000000,540,39,1,131,409,'High Usage',5
224660000400,500,36,1,91,409,'High Usage',5
224660001500,420,30,1,11,409,'High Usage',5
224660002300,388,28,1,21,409,'High Usage',5
224660004000,360,26,1,49,409,'High Usage',5
224660004200,300,21,2,280,20,'High Usage',5
224660004900,299,21,2,279,20,'High Usage',5
224660005300,295,21,2,275,20,'High Usage',5
224660005800,292,21,2,272,20,'High Usage',5
224660008300,290,21,2,270,20,'High Usage',5
224660008400,238,17,2,218,20,'High Usage',4
224660008800,179,13,2,159,20,'Low Usage',3
224660010900,160,11,2,140,20,'Low Usage',3
224660011100,159,11,2,139,20,'Low Usage',3
224660011400,152,11,2,132,20,'Low Usage',2
224660012000,140,10,2,120,20,'Low Usage',2
224660012200,140,10,2,120,20,'Low Usage',2
224660012500,140,10,2,120,20,'Low Usage',2
224660012700,140,10,2,120,20,'Low Usage',2
224660016700,140,10,2,120,20,'Low Usage',2
224660018500,140,10,2,120,20,'Low Usage',2
```

Figure 11. Part of the SMS subscriber dataset used as training data for the J48 algorithm.

4.4.1.3 DATA DATASET

The DATA/GPRS dataset contained the following parameters/attributes: subscriber, MB, GB, AVG_MB/D, Group, deviation to average, Usage Rating, Group AVG. Table 8 has details about all the attributes.

Table 8. Parameter explanations on data dataset

Parameter	**Description**
Subscriber	A subscriber as defined on the system
MB	The number of megabytes
GB	The number of Gigabytes
AVG_MB/D	The average number of megabytes used by the subscriber.
Group	The group number in which the subscriber is categorized from the source data
deviation to average	The difference between the group average and the subscriber minutes
Usage Rating	The rating on a scale of 1-5 as per the usage matrix defined.
Group AVG	The average number of MB used by an individual subscriber on that group

In figure 12. Part of the DATA stream dataset that was used for after using the weka remove function can be found.

```
@relation 'data dataset-weka.filters.unsupervised.attribute.Remove-R9-10'

@attribute subscriber numeric
@attribute MB numeric
@attribute GB numeric
@attribute AVG_MB/D numeric
@attribute Group numeric
@attribute 'deviation to average' numeric
@attribute 'Group AVG' numeric
@attribute 'usage group' {'High Usage','Moderate Usage','Low Usage'}
@attribute 'Usage Rating' numeric {'1,2,3,4,5}

@data
224660865234,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224661258498,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224660001502,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224660002305,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224660004992,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224668004223,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224668004945,20480,20,1462.857143,1,7293.706162,7313.706162,'High Usage',5
224668005312,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668005889,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668008314,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668008489,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668008823,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668010945,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668011112,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668011411,15360,15,1097.142857,1,7298.706162,7313.706162,'High Usage',3
224668012892,13312,13,950.857143,1,7300.706162,7313.706162,'High Usage',3
224668122025,13312,13,950.857143,1,7300.706162,7313.706162,'High Usage',3
224668212517,13312,13,950.857143,1,7300.706162,7313.706162,'High Usage',3
224668012723,13312,13,950.857143,1,7300.706162,7313.706162,'High Usage',3
224668016778,13312,13,950.857143,1,7300.706162,7313.706162,'High Usage',3
```

Figure 12. Part of the DATA subscriber dataset used as training data for the J48 algorithm.

In table 9. A sample of the products that are loaded in the database is found. These products can be recommended to a subscriber.

Table 9. Part of the 100 products to be recommended to subscribers

Contract No	Price	Minutes	SMSs	DATA
1	R250	250	500	500
2	R350	800	limitless	2048
3	R189	50	25	2048
4	R150	120	300	300
5	R160	50	25	2048
6	R190	100	400	1024
7	R229	100	200	2048
8	R179	50	0	2048
9	R200	100	500	2048
10	R250	200	limitless	2048
11	R120	50	0	1024
12	R130	100	0	1024
13	R400	limitless	limitless	1024
14	R379	800	0	1024
15	R120	0	150	500
16	R250	700	100	1024
17	R229	300	200	500
18	R149	50	300	500

Table 9 shows part of the products that are on the database, these are some of the products that can be recommended to the mobile subscriber.

Relation: Combined Dataset

No.	Subscriber Numeric	MB Numeric	GB Numeric	AVG_MB/D Numeric	Data group Nominal	Data Rating Numeric	Monthly usage Numeric	Monthly GB Numeric	minutes Numeric	average minutes/day Nominal	group Numeric	deviation to average Numeric	CLUSTER_AVG Numeric	Voice group Nominal	Voice Rating Numeric	NO_SMS Numeric	average SMS per day Numeric	SMS group Nominal	SMS Ra Nume
1	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	600.0	42:51:26	1.0	179.4999	420.5	High	5.0	540.0	39.0	High	
2	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	600.0	42:51:26	1.0	179.5	420.5	High	5.0	500.0	36.0	High	
3	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	429.0	30:38:34	1.0	8.5	420.5	High	4.0	420.0	30.0	High	
4	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	365.0	26:04:17	1.0	55.5	420.5	High	4.0	388.0	28.0	High	
5	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	288.0	20:34:17	1.0	132.5	420.5	High	2.0	360.0	26.0	High	
6	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	288.0	20:34:17	1.0	132.5	420.5	High	2.0	300.0	21.0	High	
7	2.2466E11	20480.0	20.0	1462.857...	High	5.0	43885.71429	42.85714...	199.0	14:12:51	2.0	8.8	207.8	Moderate	1.0	299.0	21.0	High	
8	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	199.0	14:12:51	2.0	8.8	207.8	Moderate	1.0	295.0	21.0	High	
9	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	199.0	14:12:51	2.0	8.8	207.8	Moderate	1.0	292.0	21.0	High	
10	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	199.0	14:12:51	2.0	8.8	207.8	Moderate	1.0	290.0	21.0	High	
11	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	199.0	14:12:51	2.0	8.8	207.8	Moderate	1.0	238.0	17.0	High	
12	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	189.0	13:30:00	2.0	18.8	207.8	Moderate	1.0	179.0	13.0	Low	
13	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	187.0	13:21:26	2.0	20.8	207.8	Moderate	1.0	160.0	11.0	Low	
14	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	185.0	13:12:51	2.0	22.8	207.8	Moderate	1.0	159.0	11.0	Low	
15	2.2466E11	15360.0	15.0	1097.142...	High	3.0	32914.28571	32.14285...	180.0	12:51:26	2.0	27.8	207.8	Low	1.0	152.0	11.0	Low	
16	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
17	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
18	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
19	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
20	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
21	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
22	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	140.0	10.0	Low	
23	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	92.0	7.0	Low	
24	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	88.0	6.0	Low	
25	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	80.0	6.0	Low	
26	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	73.0	5.0	Low	
27	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	69.0	5.0	Low	
28	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	67.0	5.0	Low	
29	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	66.0	5.0	Low	
30	2.2466E11	13312.0	13.0	950.8571...	High	3.0	28525.71429	27.85714...	120.0	8:34:17	3.0	38.4	81.6	Low	1.0	66.0	5.0	Low	

Figure 13. The full dataset after pre-processing

Figure 13 shows the full dataset after it has been combined and pre-processed. The full dataset has all the attributes and streams.

Figure 14 shows the predicted contracts for subscribers using binary whereby true and false means that the contract has been correctly and not correctly predicted, respectively.

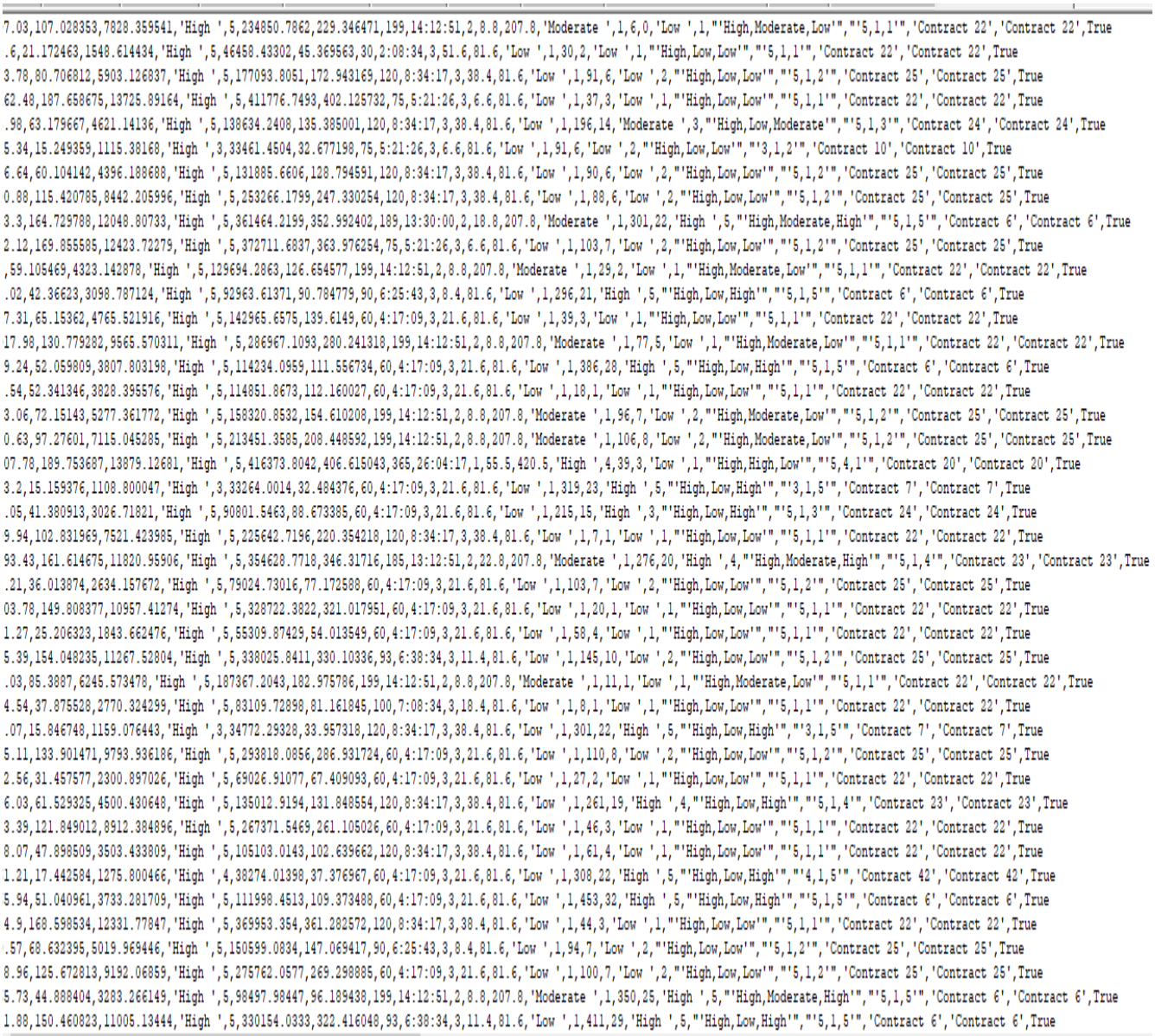

Figure 14. Combined dataset used for predicting the contracts based for subscribers

4.5 DATA CLEANING AND PREPARATION

All the four datasets were cleaned but still maintaining the data integrity before experimenting with them. After establishing which parameters to use for the algorithm, the data was then pruned. Each dataset contained over four thousand (4000) records. The reason for the separation of the datasets before evaluating the full data set is to ensure that each dataset is fully analysed and classified before it is combined into a single set. Also, in the mobile subscriber space the streams are always separated and only combined for reporting purposes.

The pre-processing stages consisted of four main steps:

I. Opening the csv file
II. Removing attributes that are not necessary for the experiment
III. Converting the file to .arff
IV. Manually choosing the usage group attribute as a class attribute.

These pre-processing steps were crucial to ensure that the appropriate data were used in experiments.

After pre-processing all the datasets namely, VOICE, SMS, DATA and the full or combined dataset, it was ensured that 30% of the entries were blank so that predictions can be formed. For the VOICE, SMS and DATA dataset, the algorithm was used to predict the usage class. There are only three (3) usage classes which are mainly ('High Usage', 'Low Usage', 'Moderate Usage').

It is imperative to note that every stream has been individually evaluated and the performance of the algorithms differ for every stream.

Table 10. Proposed utility matrix showing five (5) point scale

	SMS	DATA	VOICE
1	1-5	1MB-20.99MB	1-15:59
2	6-10	21MB-50.99MB	16:00-20:59
3	11-15	51MB-100.99MB	21-25:59
4	16-20	101MB-499.99MB	26-30:59Min
5	20+	500MB+	31:00+Min

Table 10 illustrates a proposed utility matrix that has been used together with the usage classification. For example, a subscriber might be classified as high usage for data but have a rating scale of four (4) and another similar subscriber might also be high usage for DATA, but the rating scale can be five (5). The utility matrix is used in conjunction with the usage classification.

Table 11. Example of subscriber data represented using the utility matrix in table 10

Subscriber	Voice	Data	sms
Subscriber 1	1	4	-
Subscriber 2	4	5	-
Subscriber 3	2	-	5
Subscriber 4	3	2	1
Subscriber 5	-	4	1
Subscriber 6	2	-	-
Subscriber 7	4	-	-
Subscriber 8	5	-	4
Subscriber 9	5	1	3
Subscriber 10	1	4	3
Subscriber 11	4	4	2
Subscriber 12	1	5	3
Subscriber 13	1	2	-
Subscriber 14	5	5	1
Subscriber 15	1	5	2
Subscriber 16	-	1	2

As shown in Table 11 a utility matrix has been employed to make the system function like a recommender system and use the matrix of 1-5 to recommend products that are within the usage range.

The recommender system in this study there were two classes of entries which are referred to as subscribers and products. Subscribers have a usage history for a certain product and the usage history is then represented as a number rating ranging from one to five (1-5) with five (5) being the highest and one (1) being the lowest and (-) representing that the subscriber has not made use of the product for the time range in question.

Usage ratings could be ignored and only focus on the usage classification (High Usage, Moderate Usage, Low Usage). However, to increase the accuracy of the recommendation engine the matrix was employed. Looking at Table 11 subscribers 4 and 5 both used VOICE, SMS and DATA but their usage levels are not comparable. Similarly, subscriber 8 and 9 both had a high usage for VOICE and SMS but both used little to or data.

4.5.1 COMBINATIONS AND PERMUTATIONS

The usage categories, namely, high usage, moderate usage and low usage can be represented as a combination of three strings (H, M, L). Permutations and combinations are part of a branch of mathematics called combinatorics, which involves studying finite, discrete structures. (Nelson, 2013). Permutations are specific selections of elements within a set where the order in which the elements are arranged is important, while combinations involve the selection of elements without regard for order. A typical combination lock, for example, should technically be called a permutation lock by mathematical standards since the order of the numbers entered is important; 1-2-9 is not the same as 2-9-1, whereas for a combination any order of those three numbers would suffice.

The generalized equation for a permutation can be written as:

$$n\,\mathrm{Pr} = \frac{n!}{(n-r)!} \qquad 38$$

where n is the number of items you are choosing from and r is the number of items.

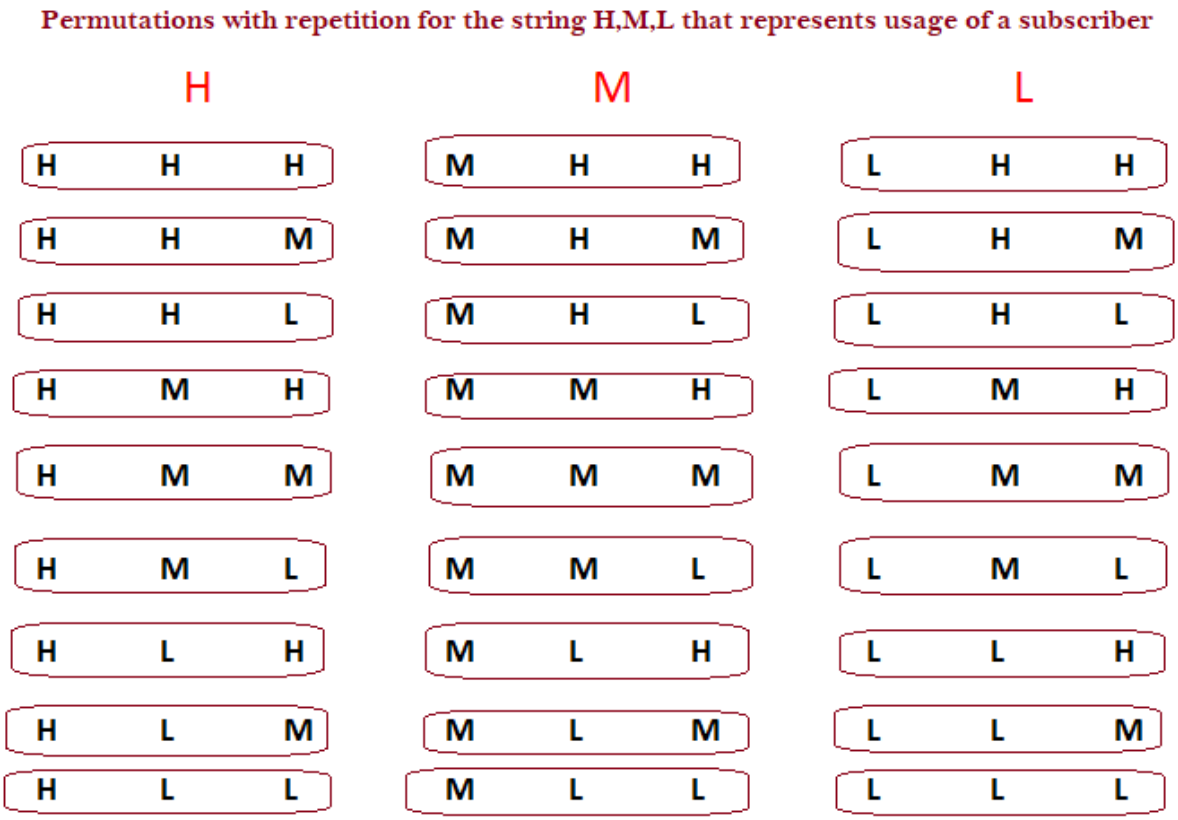

Figure 15. String permutation for usage.

To arrive at the permutations in Figure 15, a permutations generator with repetition has been used, which can be found on Dcode (2017). This is an essential part of this study because a subscriber can be categorized according to the above permutations. Two or more subscribers can be categorized in a similar way but still use different contracts. This is one of the reasons why the utility matrix in Table 10 was used together with the above permutations as illustrated in Figure 15.

Due to the nature of this study both classification and clustering have been proposed and therefore one algorithm was used for clustering and another used for classification. Classification was applied for individual streams and only after that was clustering incorporated for the full dataset. The algorithms used are explained below.

4.5.2 THE PROPOSED J48 DECISION TREE ALGORITHM FOR CLASSIFICATION

Classification is a way of building a model of classes from an arrangement of records that contain class labels. A decision Tree Algorithm is used to discover the way the characteristics vector acts for several instances. Likewise, on the bases of training instances the classes for the produced occurrences are found (Korting, 2006). This algorithm generates the rules for the prediction of the target variable (the target variable is selected by the user). With the assistance of the tree characterization algorithm the basic circulation of the data is effectively justifiable (Korting, 2006).

J48 is an extension of ID3. The additional features of *J48* are accounting for missing values, decision trees pruning, continuous attribute value ranges, derivation of rules, etc. *J48* can be used as an open source Java implementation of the C4.5 algorithm. There are several options associated with tree pruning for this algorithm. In case of potential over fitting pruning can be used as a tool for précising. In different algorithms the classification is performed recursively till each and every leaf is unadulterated so that characterization of the information ought to be as impeccable as could be allowed. This algorithm produces the tenets from which specific identity of that information is created. The goal of the *J48* algorithm is to continuously speculate on a decision tree until the point when it élites harmony of adaptability and exactness (Kaur and Chhabra, 2014).

Basic Steps in the Algorithm:

- In case the instances belong to the same class the tree represents a leaf and so the leaf is returned by labeling with the same class.
- The potential information is calculated for every attribute given by a test on the attribute. Then the gain in information is calculated and this would result from a test on the attribute.
- Then the best attribute is found on the basis of the present selection criterion and that attribute selected for branching (Korting, 2006).

4.5.3 THE PROPOSED DATA CLUSTERING USING K-MEANS ALGORITHM

The objective of data clustering, otherwise called cluster analysis, is to find the common grouping(s) of an arrangement of examples, focuses, or questions. Sharma et al. (2012) characterized cluster analysis as "a measurable arrangement procedure for finding whether the people (thus referred to as subscribers) of a populace fall into various gatherings by making quantitative examinations of different attributes."

The most well-known hierarchical algorithms are single-link and complete-link; the most popular and the simplest partitional algorithm is *k-means* (Jung et al., 2014). Since partitional calculations are favored in design acknowledgment because of the idea of accessible information, the scope here is centered on these calculations. *k-means* has a rich and varied history as it was autonomously found in numerous research papers and commercial projects. Even though *k-means* was first proposed more than 50 years back, it is still a standout amongst the most broadly utilized calculations for grouping. Simplicity of execution, effortlessness, proficiency, and observational achievement are the primary explanations behind its fame.

In data mining, *k-means* clustering (Gao and Shrinkage, 2009) is a method of cluster analysis which aims to partition *n* observations into *k* clusters in which each observation belongs to the cluster with the nearest mean. This results into a partitioning of the data space into Verona cells. *k-means* is one of the most straightforward unsupervised learning algorithms that makes provision for the notable grouping issue (Sharma et al., 2012). The procedure follows a straightforward and simple approach to arrange a given informational collection through a specific number of clusters (assume k clusters) fixed *a priori*. The fundamental thought is to characterize k centroids, one for each group. These centroids ought to be put cleverly since of various area causes diverse outcome. In this way, the better decision is to put them however much as could reasonably be expected far from each other. The subsequent stage is to take each direct having a place toward a given informational collection and partner it to the closest centroid.

At the point when no point is pending, the initial step is finished, and an early gathering age is finished. Now *k* new centroids must be recalculated as ban focuses of the groups coming about because of the past step. After getting these new *k* centroids, another coupling must be done

between similar informational collection focuses and the closest new centroid. A loop has been generated. As a result of the loop it can be seen that the *k* centroids change their position until they are well-ordered, and no more changes are done. As such, centroids do not move any more.

The algorithm is composed of the following steps:
1. Place K points into the space represented by the objects that are being clustered. These points represent initial group centroids.
2. Assign each object to the group that has the closest centroid.
3. When all objects have been assigned, recalculate the positions of the K centroids.
4. Repeat Steps 2 and 3 until the centroids no longer move. This produces a separation of the objects into groups from which the metric to be minimized can be calculated.

The system is a hybrid system therefore each recommendation process follows these simple steps below

I. Usage content characteristics are computed.
II. A time series is constructed.
III. Similarity with other subscribers is computed.
IV. Prediction computation.
V. Recommendation.

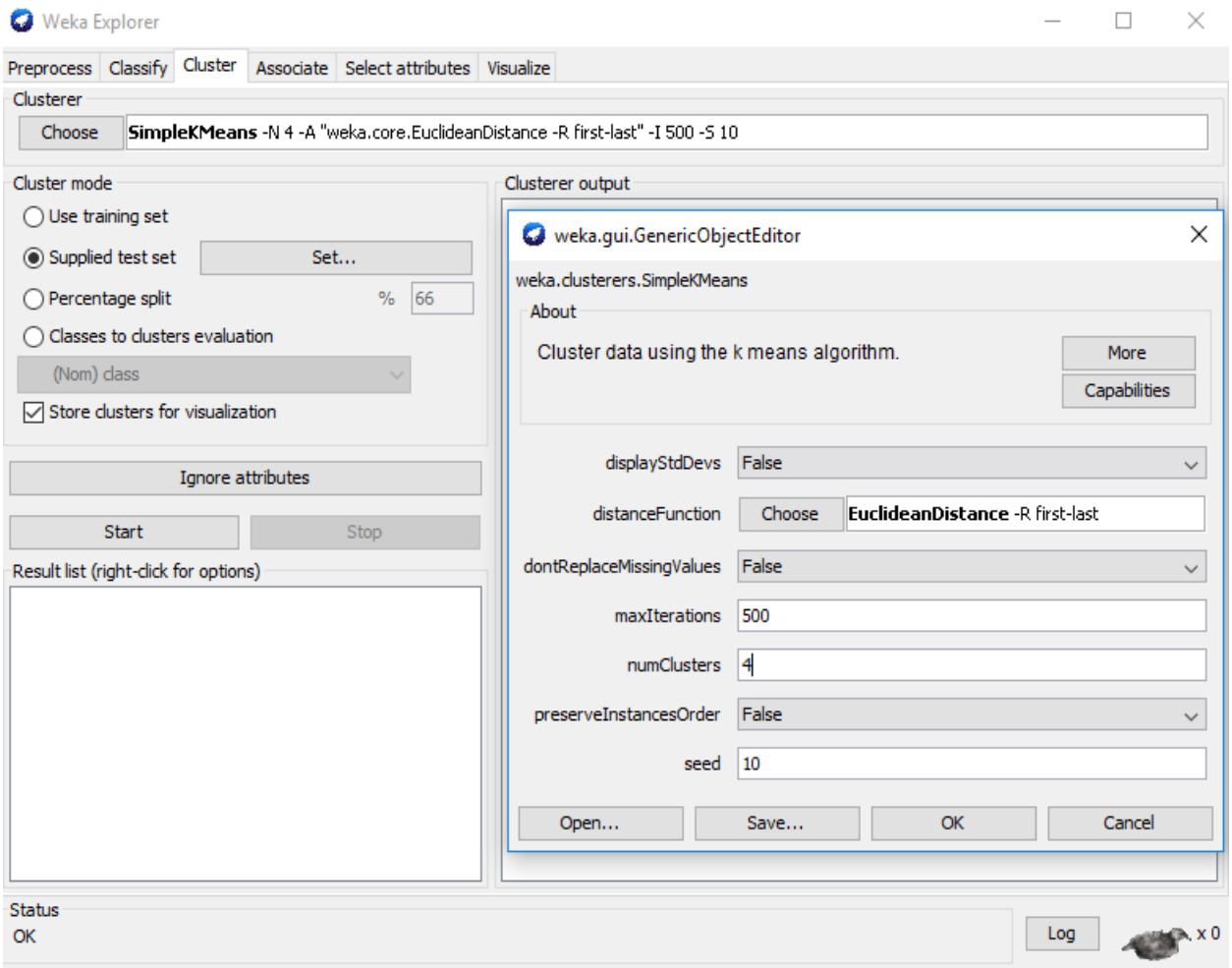

Figure 16. K-means algorithm in Weka showing the use of Euclidean distance function and number of clusters

The simple *k-means* algorithm supports four distance functions, this study used the Euclidean Distance function in this study as it can be seen in Figure 16.

The Euclidean Distance formula can be represented as follows

$$dist = \sqrt{\sum_{k=1}^{n} (P_k - q_k)^2} \qquad 39$$

where n is the number of dimensions (attributes) p_k and q_k are, respectively, the k^{th} attributes (components) of records p and q.

4.6 CLIENT-SIDE EVALUATION OF THE RECOMMENDER SYSTEM

There are several evaluation methods that can be applied and used in recommender systems. These methods differ depending on the goal that the researcher wishes to achieve. The evaluation of this study is two-fold: the first part of the evaluation is offline, on the server side and the other part is done online and on the client side. It is imperative to note that the recommender system in this study can be seen as a client server architecture whereby the client side is the subscriber mobile device and the server side is equipped with the database and recommendation system. used a graphical evaluation precision and recall in evaluating the system from the client side has been used.

To effectively evaluate the system precision and recall were calculated as measures of effectiveness. In this research, precision is defined as the total number of all relevant products divided by the total number of all retrieved products. Precision and recall is represented as follows.

$$Precision = \frac{A}{A+C} \quad 40$$

$$Recall = \frac{A}{A+B} \quad 41$$

where *A* is the number of relevant items recommended, *C* is the number of relevant items in the database and B is the number of all irrelevant items recommended.

To the user the scalar value of recall indicates the ability of the system to find relevant items as per query from the collection of different products. Precision demonstrates the capacity to yield pertinent products according to the query. When all is completed, the subscriber is keen on the pertinent prescribed products. Thus, the measures of precision and recall become measures of interest to the subscribers. The lower the value the more undesirable is the performance of the system. The Precision-Recall graph gives a visual performance of the recommender system and the graph is given in the next chapter. These performance measures were used to measure the recommender system.

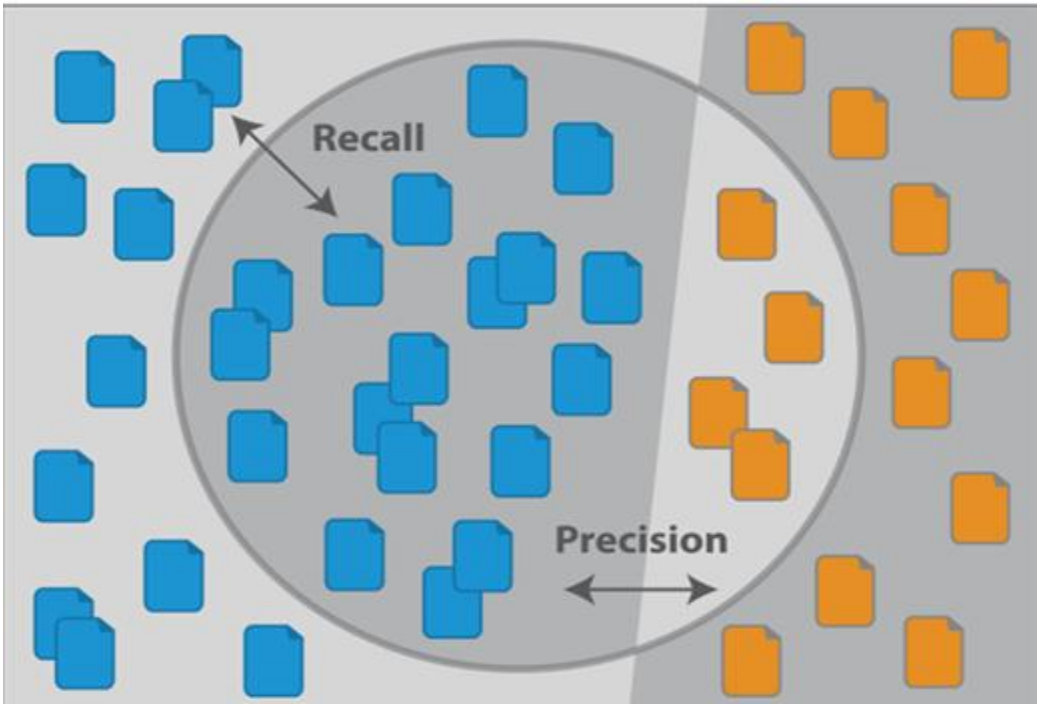

Figure 17. Pictorial representation of precision and recall.

Figure 17 shows the precision and recall in a pictorial manner. The orange components within the circle show a predictive value while the ones labelled recall show the fraction of relevant instances that have been retrieved.

4.7 CHAPTER SUMMARY

In this chapter the researcher has illustrated and explained the dataset because the mobile subscriber dataset is different from other datasets that are available from databases. This chapter has outlined how the classification together with clustering algorithms played a role in preparation for the recommendation system. An illustration of how the hybrid works using content as well as collaborative filtering approaches and the incorporation of a time series was also done. This chapter also summarised the steps of how to get to the recommendation in five (5) simple steps. A utility matrix was introduced, and its' relevance explained in conjunction with the combinations. The use of a Euclidean distance function was illustrated in this chapter and lastly an illustration of how precision and recall are calculated has been illustrated. The next chapter discusses experiments and results.

CHAPTER 5

5 EXPERIMENTS, RESULTS AND INTERPRETATION

This chapter describes and explains the experiments and all the work done in evaluating the recommender system including all the steps taken and how the evaluation was done on the system. The dataset for the mobile subscriber was also analyzed and the results are provided below. The results are separated into two: client-side results and server-side results. The purpose of these experiments are to:

- Measure effectiveness of the system
- Group similar subscribers
- Simulate the recommender system

Weka makes available an atmosphere for comparing learning algorithms, graphical user interface, comprehensive set of data pre-processing tools, learning algorithms and evaluation methods. Furthermore, Weka makes available the enactment of Regression, Clustering, Classification, Association rules and feature selection. As part of the experiment the classification algorithm *J48* which is an open source Java implementation of the C4.5 algorithm in Weka was used. The Weka algorithms can be directly applied to a dataset or called from a Java code (Hall et al., 2009). Weka also provides numerous means for loading data such as (ARFF) or (CSV) files. This is done before the data is loaded in the recommender system, as illustrated in chapter four (Figure 9). The data from weka then serves as the source to the recommender system.

SERVER-SIDE RESULTS

5.1 RESULTS FOR CORRECTLY CLASSIFYING THE SUBSCRIBER INSTANCES.

From the results in Table 12 it is evident that J48 has correctly classified all the voice subscribers with a 0% incorrectly classified instances. This high accuracy may be due the multiple subscribers with similar voice usage. However, during training of the algorithm the parameters were adjusted to ensure that the algorithm performs at an optimum level. DATA stream test set was also classified with 100% correctly classified instances with 0% incorrectly classified. SMS test set has a 99.98% correctly classified. Overall this high accuracy of the classifier was needed so that the

combinations can be applied properly and the recommendation accuracy to improve. The respective classifier trees are found in Figure 18, Figure 19 and Figure 20, respectively.

Table 12. Results for correctly classifying subscriber instances.

Dataset	**Correctly classified instances**	**Incorrectly classified instances (percentage)**
VOICE	100%	0%
DATA	100%	0%
SMS	99.9853%	0.0147%

J48 Tree classifier has given us the following tree for classifiying minutes.

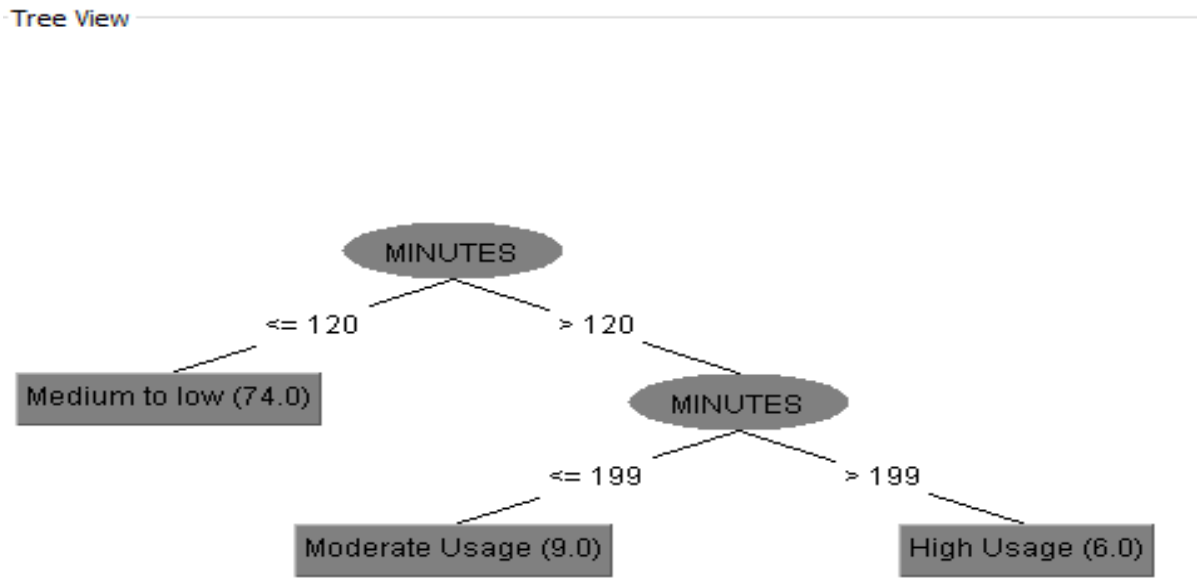

Figure 18. J48 Tree classifier for voice model

Figure 18 is shows that a subscriber with a usage of less that 120 minutes is classified as medium to low, and a subscriber that has a usage that is greater than 120 minutes but less than 200 minutes is classified as moderate usage and that with 200 minutes or more is classified as high usage.

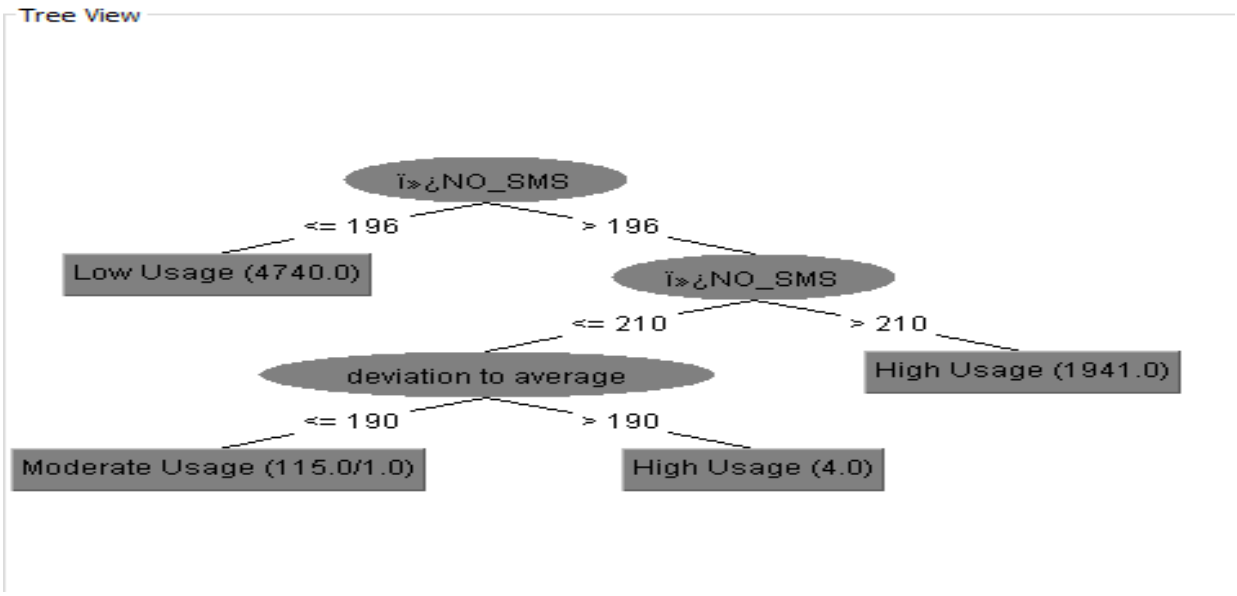

Figure 19. Tree visualizer for J48 algorithm on SMS dataset

Figure 19 shows how the *J48* algorithm has learned the data and how datasets have been classified. From all the attributes, the *J48* algorithm has used the No SMS attribute as well as the deviation average to classify the subscribers. The number of SMSs does determine the usage group, but it is uses the deviation to average attribute as well to see how far the usage is from the group average so that as new data comes through, it can adjust group averages and subscriber classification as well.

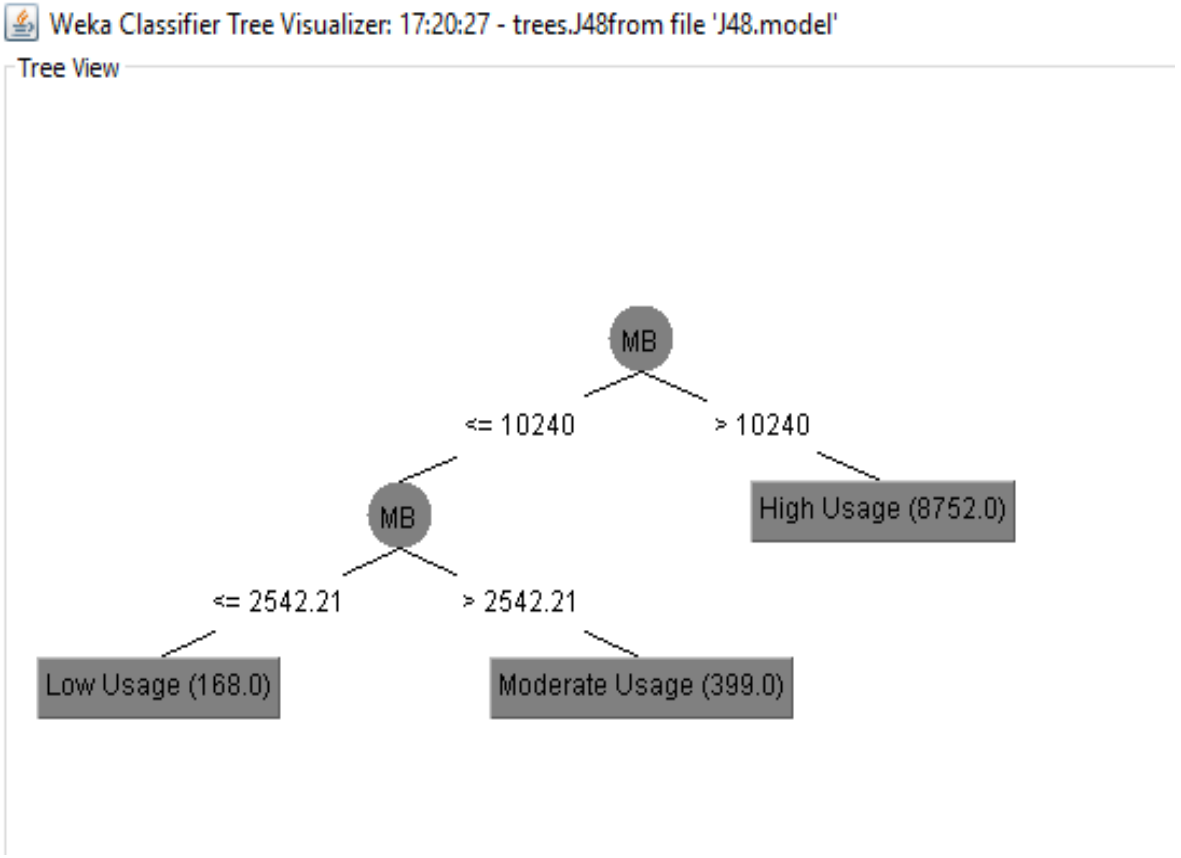

Figure 20. Tree visualizer for J48 algorithm on DATA dataset

Figure 20 shows how the *J48* algorithm classified the data from the dataset provided. From all the attributes, the *J48* algorithm has used the MB attribute to classify the subscribers because that is the attribute that determines which category a subscriber falls into.

After obtaining satisfactory results on all three datasets as shown in Table 12, a test was also done on the data on three other different algorithms and the results were captured as follows.

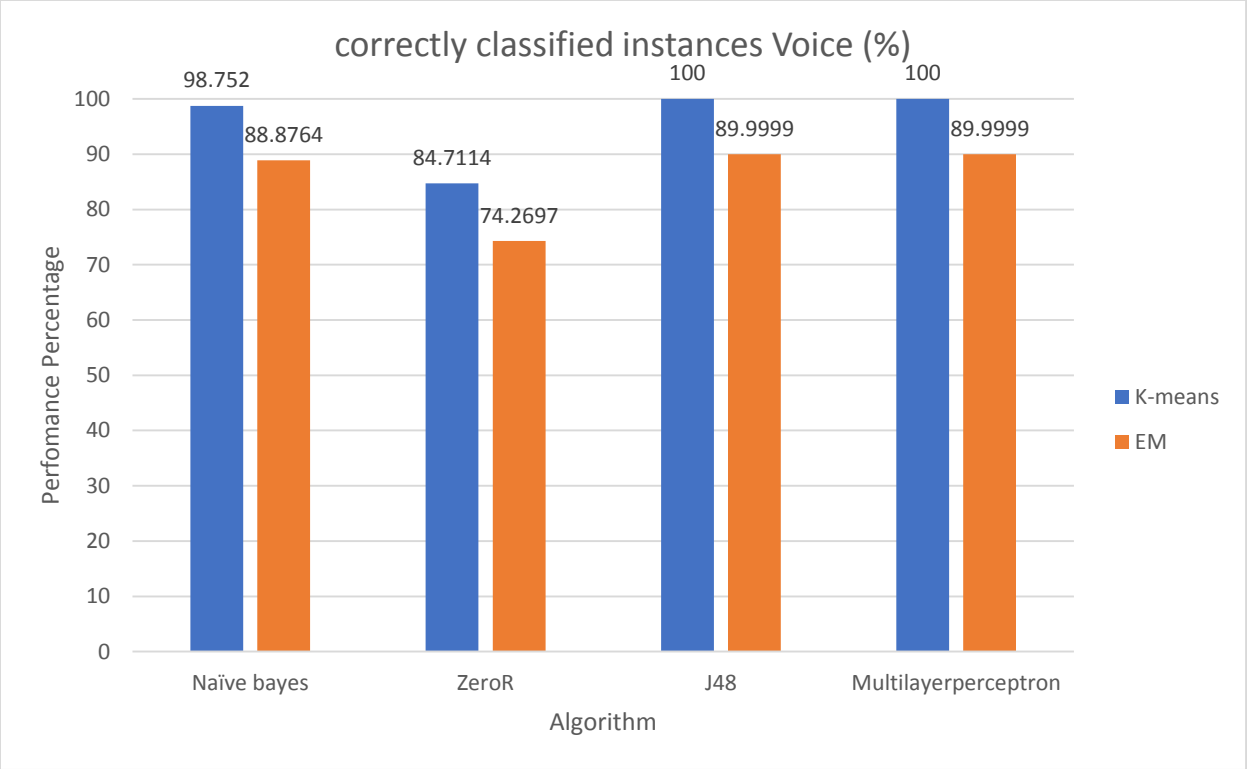

Figure 21. Results for correctly classifying the voice instances

To make the recommendations the datasets needed to be classified. It is of outmost importance that the subscribers are correctly classified therefore the above algorithms were tested on all datasets after they have been clustered on *k-means* and *EM* and subscribers classified according to their usage groups. Figure 21 shows the results for correctly classifying the voice subscribers. It is visible on figure 21 that *k-means* together with *J48* outperformed all other algorithms.

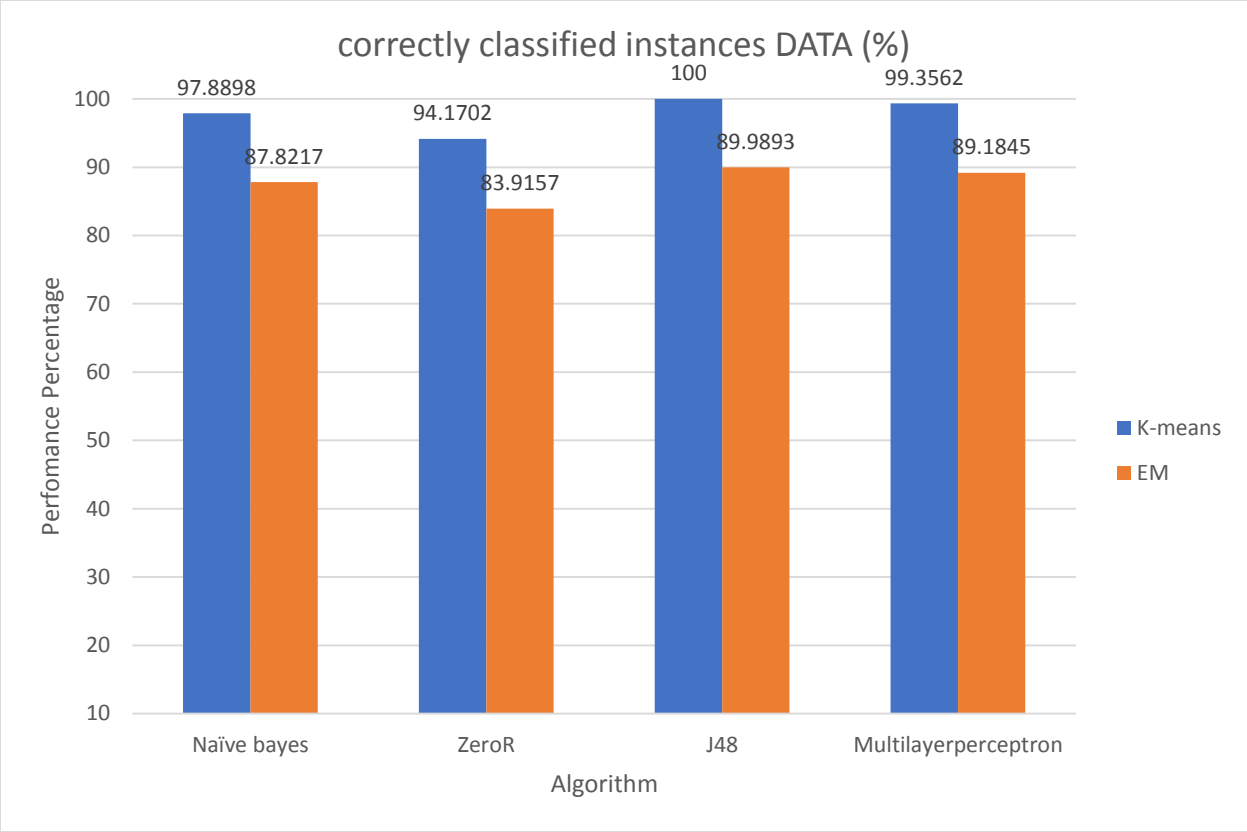

Figure 22. Results for correctly classified DATA instances

By correctly classifying the instances for the DATA dataset, the results in Figure 22 show that the *J48* on *k-means* clustered dataset outperformed the other algorithms by yielding 100%.

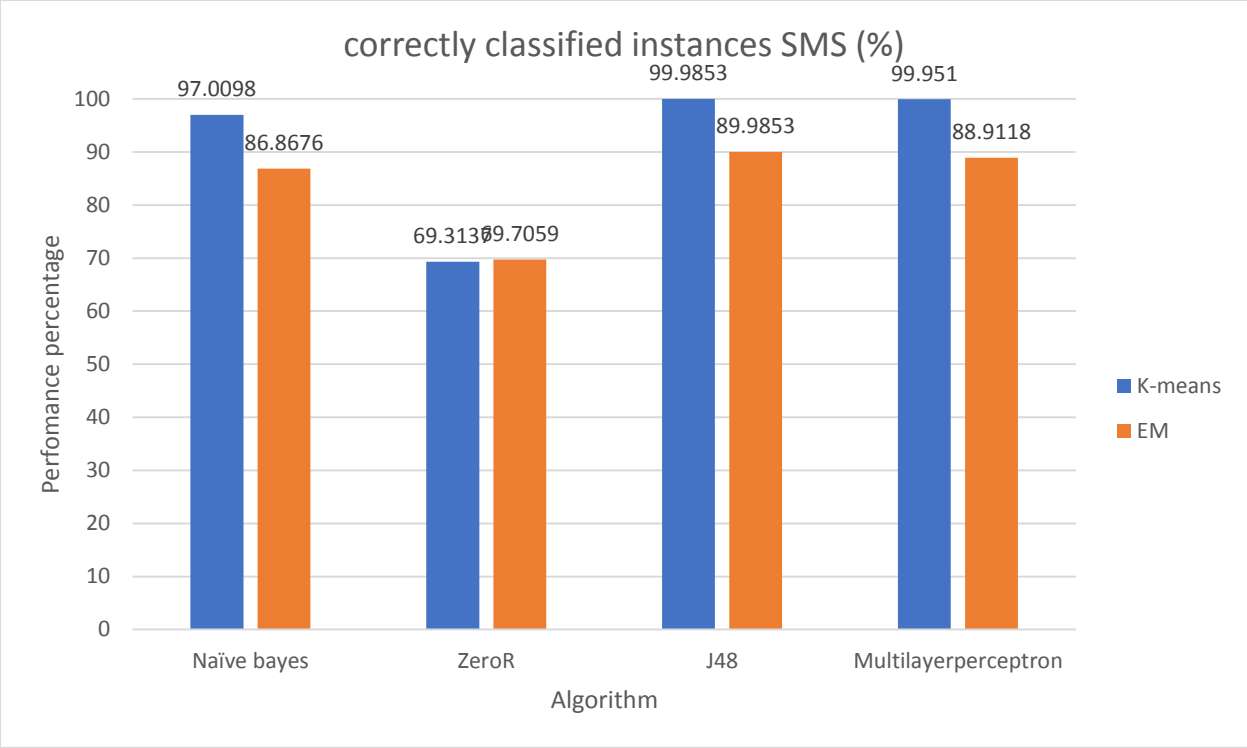

Figure 23. Results for correctly classified SMS instances

By correctly classifying the instances for the SMS dataset, figure 23 show that the *J48* on *k-means* clustered dataset outperformed the other algorithms.

COMPARISON OF ALGORITHM RUN TIME

Table 13. Comparison of EM and K-means clustering run time

Number of clusters	Algorithm run time	
	EM	K-means
1	0.15 seconds	0.2 seconds
2	0.29 seconds	0.3 seconds
3	0.36 seconds	0.3 seconds
4	0.34 seconds	0.2 seconds

Table 13 shows details of how the two algorithms performed in terms of the runtime on the full dataset. In our dataset, *EM* runtime was relatively faster than *k-means* when the number of clusters was between one (1) and two (2). However, when the number of clusters were between three (3) and four (4) *k-means* ran faster than *EM*. With an increase in number of subscribers the number of clusters can be increased and based on the results in Table 13. EM clustering would impact performance of the system with an increase in number of clusters.

5.2 RESULTS CLUSTERING USING K-MEANS

Figure 24 shows the results that were obtained using k-means clustering algorithm in weka. The number of clusters have been set to four so that the interpretation of the results can be clearer, and the dataset can be fully represented. The experiments started with two clusters that were congested, then gradually increased the number of clusters till a point of purity i.e. cluster four (4) was reached. Cluster 0 contains 48% (1621) of subscribers), cluster 1 contains 42 % (14233) of subscribers, cluster 2 contains 8% (272) of subscribers) and lastly cluster 3 contains 2% (74) of the population. This study used up to four clusters and the results on the Figure 24 can be interpreted as follows.

```
Number of iterations: 2
Within cluster sum of squared errors: 300.4084800514328
Missing values globally replaced with mean/mode

Cluster centroids:
                         Cluster#
Attribute    Full Data          0          1          2          3
                (6599)     (3109)     (2759)      (604)      (127)
=============================================================================
Voice            0.8871          1     0.9949          0          0
Sms              0.5819          1          0          1          1
GPRS             0.9326     0.8993     0.9982          1          0



Time taken to build model (percentage split) : 0.02 seconds

Clustered Instances

0      1621 ( 48%)
1      1433 ( 42%)
2       272 (  8%)
3        74 (  2%)
```

Figure 24 simple k-means clustering algorithm results

Sixty six percent (66%) of the subscribers has been used for this experiment and the number of iterations has been set to two. The algorithm is set to replace the missing values with mean or mode even though there were no missing values.

5.2.1 FULL DATASET

The full data set contained six thousand five-hundred and ninety-nine (6599) subscribers and these subscribers used similar data with some using different products.

5.2.2 CLUSTER 0

In this cluster there were three thousand, one hundred and nine (3109) subscribers; these subscribers are similar in terms of usage patterns not usage amounts. In cluster 0, one hundred percent (100%) of subscribers used voice and SMS but only eighty nine percent (89%) used GPRS.

5.2.3 CLUSTER 1

Cluster 1 contained two thousand, seven hundred and fifty-nine (2759) subscribers. In this cluster 99% of them used voice and GPRS while 0% used SMS.

Cluster 2

This cluster had six hundred and four (604) subscribers. While 0% of the subscribers used voice 100%) of them use SMS and GPRS. The conclusion is that no subscriber used voice while there was high usage of SMS and GPRS.

5.2.4 CLUSTER 3

In this cluster there were one hundred and twenty-seven (127) subscribers with the following usage patterns: 0% uses voice, 100% used SMS and none (0%) used GPRS. This is a pure SMS cluster.

5.3 THE SUBSCRIBER DATA

After analyzing the dataset provided by the OPCO, the following results were obtained. Table 13 illustrates the different combinations that were found on the subscriber data. From Table 13, it can be concluded that 93% of all the subscribers used DATA and only 7% of the subscribers did not use any data for the term. A further 89% used VOICE while 11% of the dataset did not use VOICE. While 64% used SMS a large population of the subscribers up to 36% did not use SMS for the period of the data provided.

Table 14. Dataset results analysis

DATA	VOICE	SMS	VOICE&SMS hybrid	DATA&VOICE Hybrid	DATA&SMS Hybrid
93%	89%	64%	59%	26%	2%

Table 14 shows that these usage streams were further classified into different hybrid systems and different combinations are revealed within the data. Since not all subscribers are using the same products, further classifications and combinations are possible. Therefore, the combinations have been broken down into different hybrids. It was discovered that the VOICE&SMS hybrid has 59% usage, while DATA&VOICE has a twenty-six percent (26%) usage. Only 2% used DATA&SMS hybrid. A graphical representation of Table 14 can be found in Figure 25.

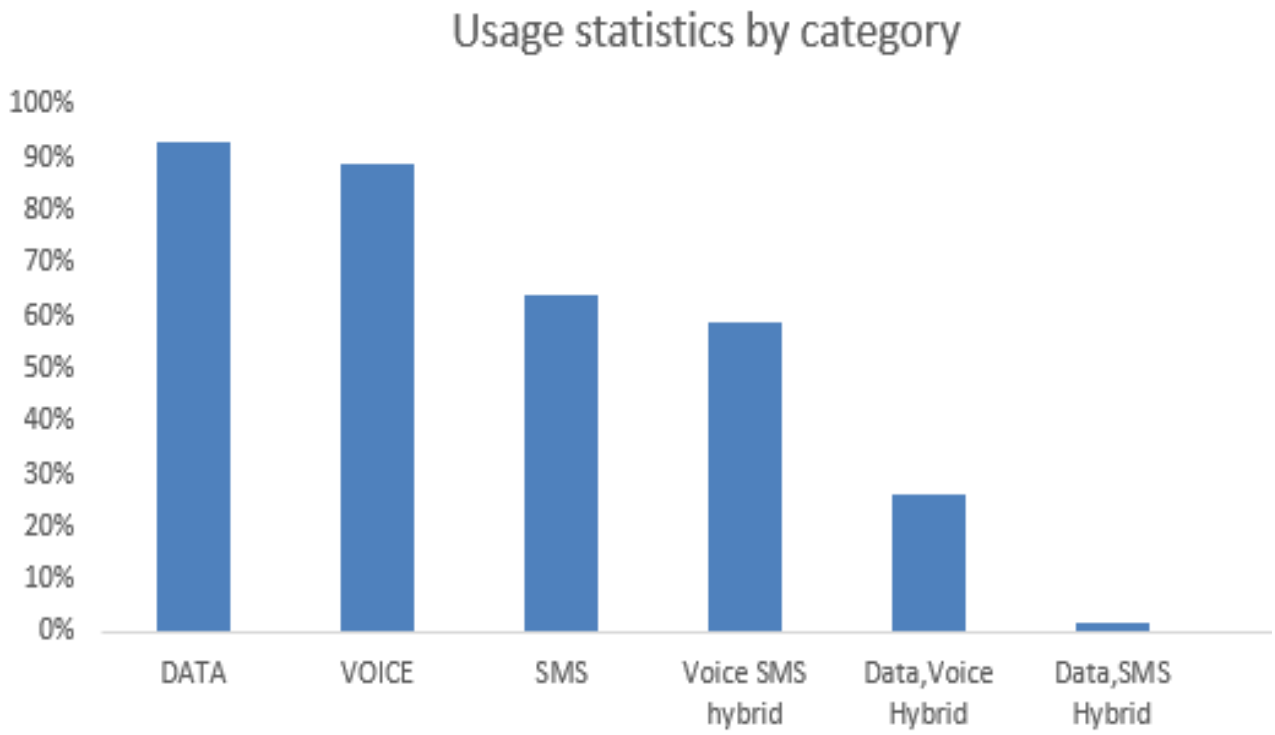

Figure 25. Results for usage statistics and hybrids.

5.4 RESULTS FOR SUBSCRIBER TIME SERIES

Figure 26 is the average usage time series for subscriber 5 per day. The results show how the current data has been averaged and used to predict the future usage. This is then used to check the similarity in usage patterns for the subscribers. The time series in Figure 27 shows that subscriber 5 does not use much data between 23h00 and 05:00AM but the usage increases between 15h00 and 23h00. This suggest that subscriber 5 is not on a contract that allows free night internet surfing or that the subscriber does not prefer to use the mobile device after midnight. The usage pattern remains the same over the weekend (Saturday and Sunday). The time series also illustrates how the usage may change in future, and to achieve this, the 4-period moving average was used to smooth the time series.

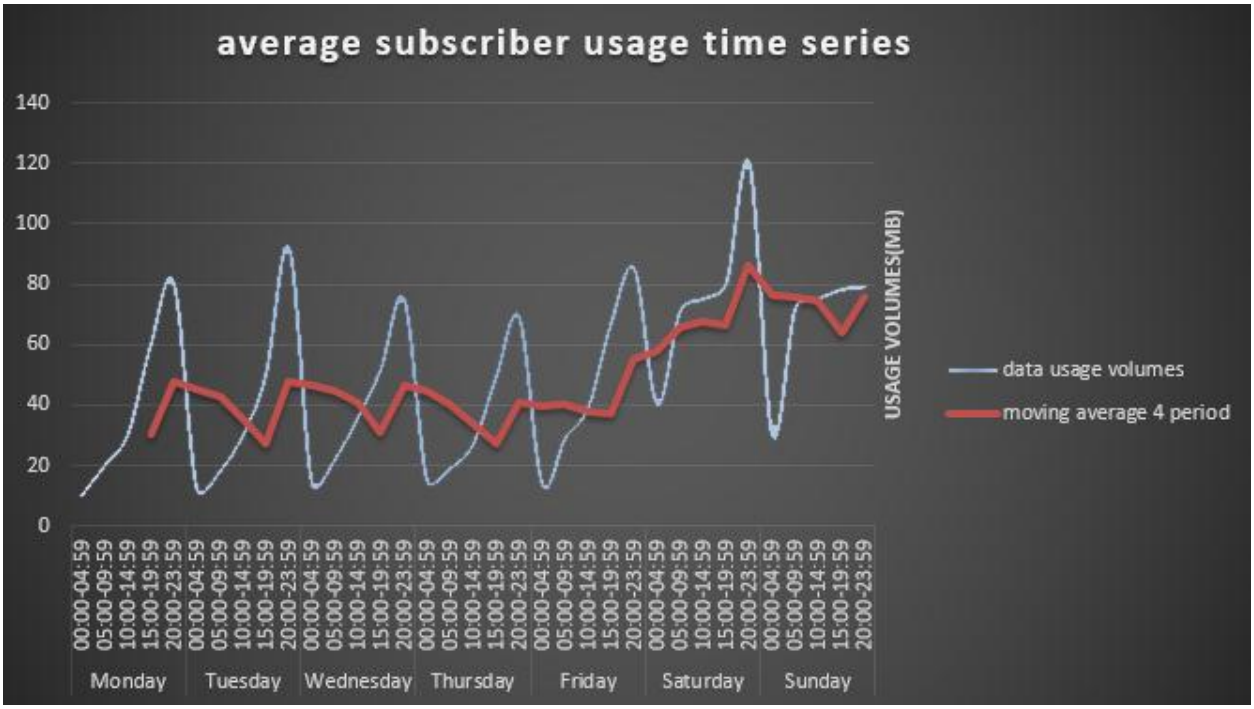

Figure 26. Results for time series for a subscriber averaged per day

CLIENT-SIDE RESULTS

5.5 RESULTS FOR THE RETRIEVAL PART OF THE RECOMMENDER SYSTEM

Query based recommendation

Hybridization allows for the retrieval part of the system to display recommendations based on collaborative filtering and content-based approaches and therefore yields recommendations that may be desirable to the mobile subscriber. Figure 27 represents the recommendations upon a subscriber query. The database is loaded with 100 products and precision and recall were evaluated.

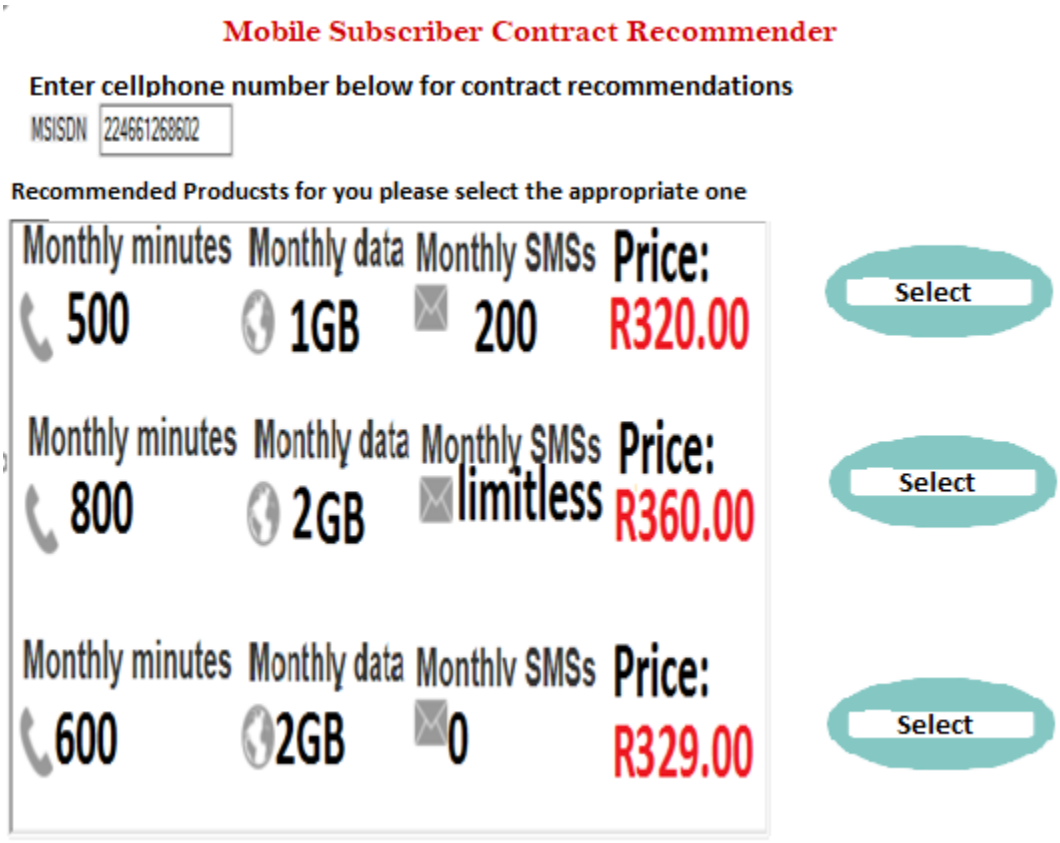

Figure 27. Results for recommendations based on query results

The products are displayed in the order of importance and a binary precision is calculated. A subscriber may have more than three items that may fit in based on the combinations, but they are only using one product. The first product on the list is the endeavor to be precise and the second and third products are within the rank. The precision-recall graph is shown in Figure 28.

5.6 SYSTEM EVALUATION

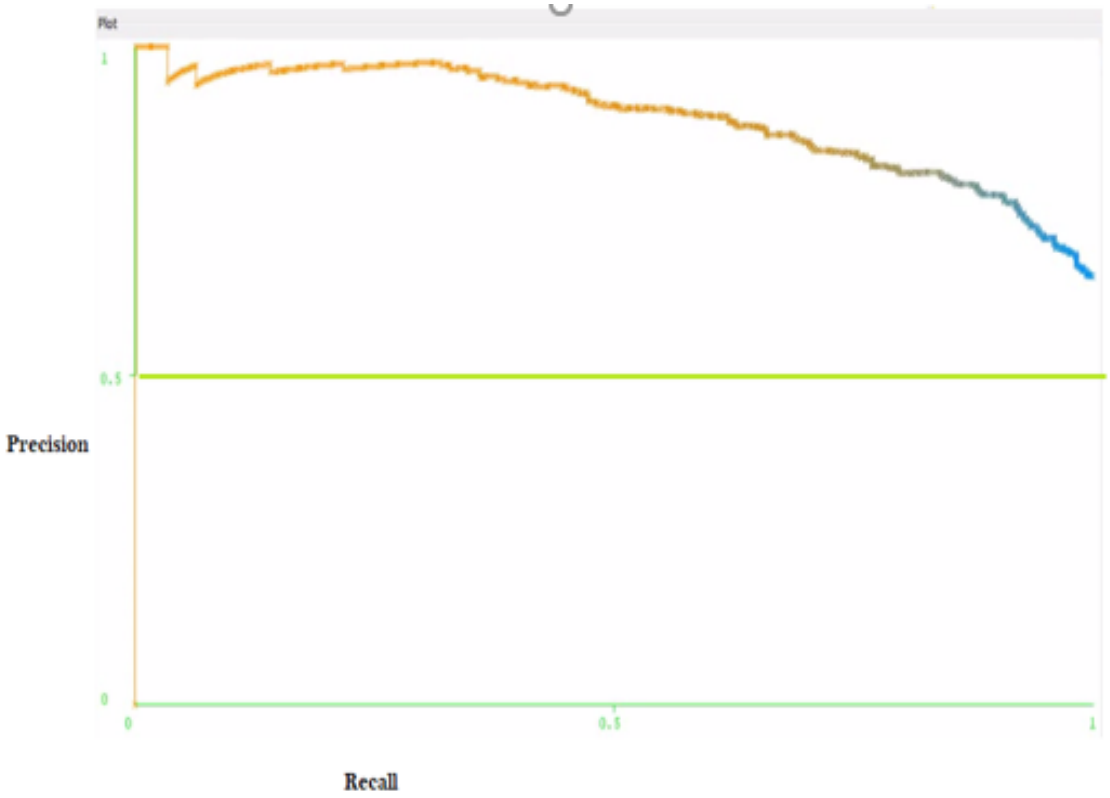

Figure 28. Precision and recall curve of a binary classifier for correctly predicting the products.

The nature of the precision and recall in this study is for a binary classifier whereby the product recommended can either be correct or not correct. If the recommended product matches the usage, then it is classified as correct and when the product does not match the usage it is classified as incorrect. The precision-recall performance in Figure 28 is exceptional, as the precision increases the recall decreases.

(Ricci, 2011) in their study found the precision of their system to be performing around 93%. (Baltrunas et al., 2012a) in their study the system was performing at 75% precision. Therefore, the precision-recall curve represented in figure 28 show good performance of the system with precision of 98%.

5.7 OVERALL RESULTS ANALYSIS

The proposed *J48* algorithm has shown excellent classification statistics with an average of 99.9% classification accuracy. The *J48* algorithm has been extensively studied and thoroughly evaluated

simple *k-means* clustering has also been studied and cluster assignments have been analyzed. It has revealed that it fulfils the objective. Both the *J48* and *k-means* algorithms have been compared with other algorithms and the hybrid has proven to be the best to fulfil the objectives of this study. From the results obtained the inaccuracy of the recommendation cannot be linked to the classifier nor the clustering algorithm. The system was evaluated on how well it can predict and recommend the correct products, and the issues of cold start were resolved by incorporating a hybrid system of the *J48* and *k-means* algorithms. The precision-recall curve has shown a good performance of the recommender system. The recommender system was able to recommend the products to the subscribers. The recommender system in this study performed better than other mobile recommender system that were considered for comparison

CHAPTER 6

6 CONCLUSION AND FUTURE WORK

This chapter provides the conclusion, contribution and future recommendations of this research. In chapter one the main objective of the research was stated as follows "To explore how current recommender system algorithms can be optimized using content and collaborative filtering for mobile subscribers". In this chapter an evaluation of whether that goal was met, will be done and discuss future work.

6.1 CONCLUSION

In this research, the use of CDRs for mobile subscribers as the source of recommendation was the main interest. An effective and efficient recommender system for mobile subscribers involves having a well classified dataset. The fundamentals of the mobile subscriber recommender system for mobile subscribers' entail having a well-tested methodology for developing the system. In the endeavor to fulfil the objective of this research the literature was reviewed and the construction and evaluation of the recommender systems was done. The section that follows is aa summary of the research to fulfil the primary objective of developing a mobile subscriber recommender System using the data that is stored by the OPCOs.

Collaborative filtering, content-based methods and similarity methods were reviewed. The importance of the assessments and revisions are to understand the difficulties and issues involved in these techniques. An in-depth understanding of issues, advantages and disadvantages of these techniques were highlighted so that they could be applied with knowledge of what short comings they may present as well as how to deal with those short comings. In this research the mobile subscriber dataset was used as provided by the OPCO.

The primary research question was detailed as follows: How can a mobile subscriber recommender system be developed in such a way that it takes into consideration usage patterns of the mobile subscriber to make a recommendation? To answer this question, extensive literature exploration was done in chapter 2. A comparison between collaborative filtering and content-based approaches was done as shown in Table 1. After the extensive literature exploration, it was discovered that a

mobile subscriber recommender system can be built using a hybrid. Different hybrid approaches were also discussed in this study.

Another important question that was immediately addressed was how to measure the effectiveness and efficiency of the developed recommender system? The developed recommendation system was then evaluated using precision and recall. The nature of precision and recall used in this study is that of a binary classifier that uses Boolean values (true and false). If the recommender system predicted the correct products the value is true, otherwise false. The precision-recall curve can be found in chapter 5 and it shows that the system performed well.

This study successfully highlighted the challenges in recommender systems, and were able to implement a hybrid system that was able to recommend and use both content based and collaborative filtering approaches. The mobile subscriber recommender system can recommend the correct products. The performance measurements reveal that the applied methods yielded satisfactory results. The classification algorithm used also outperformed other classification algorithms on mobile subscriber dataset. The incorporation of a time series for subscriber similarity computation and future usage prediction using a four-period moving average was done.

6.2 FUTURE WORK

Future research on this topic involves making the recommender system a location-based by using subscriber location that can be retrieved from the CDR and including more components to the system. These components can be the mobile dealers as well as different handset models nearest the dealer. Further research can also include how the recommender system can be made proactive instead of query based or reactive. In a proactive recommendation system scenario, the recommender will proactively push the recommendations to the subscriber.

REFERENCES

ADOMAVICIUS, G. & TUZHILIN, A. 2005. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE transactions on knowledge and data engineering,* 17**,** 734-749.

AVESANI, P., MASSA, P. & TIELLA, R. A trust-enhanced recommender system application: Moleskiing. Proceedings of the 2005 ACM symposium on Applied computing, 2005. ACM, 1589-1593.

BALTRUNAS, L., LUDWIG, B., PEER, S. & RICCI, F. 2012. Context Relevance Assessment and Exploitation in Mobile Recommender Systems. *Personal and Ubiquitous Computing,* 16.

BAMSHAD, M., BURKE, R., BHAUMIK, R. & WILLIAMS, C. 2007. Towards Trustworthy Recommender Systems. *ACM Transactions on Internet Technology,* 7**,** 1-0.

BANERJEE, A., DHILLON, I., GHOSH, J., MERUGU, S. & MODHA, D. 2007. A generalized maximum entropy approach to bregman co-clustering and matrix approximation. *Journal of Machine Learning Research* 8**,** 509–514.

BHAIDANI, S. 2008. *Recommender System Algorithms.* B. Sc thesis, Department of Mechanical and Industrial Engineering, University of Toronto, Canada.

BILENKO, M., BASIL, S. & SAHAMI, M. Adaptive product normalization: Using online learning for record linkage in comparison shopping. Data Mining, Fifth IEEE International Conference on, 2005. IEEE, 8 pp.

BILENKO, M. & MOONEY, R. J. Adaptive duplicate detection using learnable string similarity measures. Proceedings of the ninth ACM SIGKDD international conference on Knowledge discovery and data mining, 2003. ACM, 39-48.

BRIDGE, D., GÖKER, M. H., MCGINTY, L. & SMYTH, B. 2005. Case-based recommender systems. *The Knowledge Engineering Review,* 20**,** 315-320.

BURKE, R. 2002. Hybrid recommender systems: Survey and experiments. *User modeling and user-adapted interaction,* 12**,** 331-370.

BURKE, R., MOBASHER, B., WILLIAMS, C. & BHAUMIK, R. Detecting profile injection attacks in collaborative recommender systems. E-Commerce Technology, 2006. The 8th IEEE International Conference on and Enterprise Computing, E-Commerce, and E-Services, The 3rd IEEE International Conference on, 2006. IEEE, 23-23.

CASTRO SOTOS, A. E., VANHOOF, S., VAN DEN NOORTGATE, W. & ONGHENA, P. 2009. THE TRANSITIVITY MISCONCEPTION OF PEARSON'S CORRELATION COEFFICIENT. *Statistics Education Research Journal,* 8.

CHAKRABORTY, P. & KARFORMA, S. 2013. Detection of Profile-injection attacks in Recommender Systems using Outlier Analysis *Procedia Technology,* 10**,** 963-969.

CHANDRAMOULI, B., LEVANDOSKI, J. J., ELDAWY, A. & MOKBEL, M. F. 2011. StreamRec: a real-time recommender system. *Proceedings of the 2011 ACM SIGMOD International Conference on Management of data.* Athens, Greece: ACM.

CHEE, S. H. S., HAN, J. & WANG, K. 2001. RecTree: An Efficient Collaborative Filtering Method. in Proceedings of the 3rd International Conference on DataWarehousing and Knowledg**,** 141–151.

CHEN, Z., KALASHNIKOV, D. V. & MEHROTRA, S. Exploiting context analysis for combining multiple entity resolution systems. Proceedings of the 2009 ACM SIGMOD International Conference on Management of data, 2009. ACM, 207-218.

CHENG, Y. & CHURCH, G. M. 2000. Biclustering of expression data. *Proceedings of the 8th International Conference on Intelligent Systems for Molecular Biology (ISMB)***,** 93–103.

CHO, Y. H., KIM, J. K. & KIM, S. H. 2002. A personalized recommender system based on web usage mining and decision tree induction. 329–342.

CHURCH, K. & SMYTH, B. Who, what, where & when: a new approach to mobile search. Proceedings of the 13th international conference on Intelligent user interfaces, 2008. ACM, 309-312.

COHEN, W. W. & RICHMAN, J. Learning to match and cluster large high-dimensional data sets for data integration. Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining, 2002. ACM, 475-480.

DAVIS, J. V., KULIS, B., JAIN, P., SRA, S. & DHILLON, I. S. Information-theoretic metric learning. Proceedings of the 24th international conference on Machine learning, 2007. ACM, 209-216.

DCODE. 2017. *Permutations with Repetition* [Online]. Available: https://www.dcode.fr/permutations-with-repetitions [Accessed 8-February-2018 2018].

DESHPANDE, A., PUERTA, M., MELGUIZO, M. C. & BOVES, L. A proactive recommendation system for writing: helping without disrupting. 2007 New York. ACM Press.

DUNHAM, M. H. 2006. *Data mining: Introductory and advanced topics*, Pearson Education India.

DUNLOP, M., MORRISON, A., MCCALLUM, S., PTASKINSKI, P., RISBEY, C. & STEWART, F. 2004. Focussed Palmtop Information Access through Starfield Displays and Profile Matching. *Proceedings of Workshop on Mobile and Ubiquitous Information Access***,** 79-89.

FLEDER , D. & HOSANAGAR, K. 2009. The Impact of Recommender Systems on Sales Diversity. *Blockbuster Culture's Next Rise or Fall,* 55**,** 697-712.

FUNAKOSHI, K. & OHGURO, T. A content-based collaborative recommender system with detailed use of evaluations. Knowledge-Based Intelligent Engineering Systems and Allied Technologies, 2000. Proceedings. Fourth International Conference on, 2000. IEEE, 253-256.

GAO, J. & SHRINKAGE, D. B. H. J.-S. 2009. to Improve K-meansCluster Analysis‖ University of South Carolina. *Department of Statistics November*, 30.

GAVALAS, D. & ECONOMOU, D. 2011. Development platforms for mobile applications status and trends. *EEE Software,* 28**,** 77-89.

GHAZANFAR, M. A. & PRUGEL-BENNETT, A. A scalable, accurate hybrid recommender system. Knowledge Discovery and Data Mining, 2010. WKDD'10. Third International Conference on, 2010. IEEE, 94-98.

GODFREY, A. L. 2007. *A product segmentation approach and its relationship to customer segmentation approaches and recommendation system approaches*, The University of Texas at Austin.

GOSH, J. 2003. Scalable clustering. *In:* YE, I. N. (ed.) *The Handbook of Data Mining.* Lawrence Erlbaum Assoc.

GREENE, D., DOYLE, D. & CUNNINGHAM, P. Tracking the evolution of communities in dynamic social networks. Advances in social networks analysis and mining (ASONAM), 2010 international conference on, 2010. IEEE, 176-183.

GUO, G., ZHANG, J., THALMANN, D., BASU, A. & YORKE-SMITH, N. From ratings to trust: an empirical study of implicit trust in recommender systems. Proceedings of the 29th Annual ACM Symposium on Applied Computing, 2014. ACM, 248-253.

GUO, G., ZHANG, J. & YORKE-SMITH, N. A Novel Bayesian Similarity Measure for Recommender Systems. IJCAI, 2013. 2619-2625.

HALL, M., FRANK, E., HOLMES, G., PFAHRINGER, B., REUTEMANN, P. & WITTEN, I. H. 2009. The WEKA data mining software: an update. *ACM SIGKDD explorations newsletter,* 11**,** 10-18.

HASTIE, T., TIBSHIRANI, R. & FRIEDMAN, J. 2009. Unsupervised learning. *The elements of statistical learning.* Springer.

HERLOCKER, J. L., KONSTAN, J. A., TERVEEN, L. G. & RIEDL, J. T. 2004. Evaluating collaborative filtering recommender systems. *ACM Transactions on Information Systems (TOIS),* 22**,** 5-53.

HWANG, C.-S. & CHEN, Y.-P. Using trust in collaborative filtering recommendation. International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, 2007. Springer, 1052-1060.

ISHAK, I. S. & ALIAS, R. A. 2005. *Designing a strategic information system planning methodology For Malaysian institutes of higher learning (ISP-IPTA)*, Universiti Teknologi Malaysia.

JAMALI, M. & ESTER, M. Trustwalker: a random walk model for combining trust-based and item-based recommendation. Proceedings of the 15th ACM SIGKDD international conference on Knowledge discovery and data mining, 2009. ACM, 397-406.

JANNACH, D., ZANKER, M., FELFERNIG, A. & FRIEDRICH, G. 2010. *Recommender systems: an introduction*, Cambridge University Press.

JONES, M. & MARSDEN, G. 2005. *Mobile Interaction Design*, Wiley.

JØSANG, A. 2001. A logic for uncertain probabilities. *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems,* 9**,** 279-311.

JUNG, Y. G., KANG, M. S. & HEO, J. 2014. Clustering performance comparison using K-means and expectation maximization algorithms. *Biotechnology & Biotechnological Equipment,* 28**,** S44-S48.

KAUR, G. & CHHABRA, A. 2014. Improved J48 classification algorithm for the prediction of diabetes. *International Journal of Computer Applications,* 98.

KAVZOGLU, T. & MATHER, P. M. 2003. The use of backpropagating artificial neural networks in land cover classification. *International journal of remote sensing,* 24**,** 4907-4938.

KOREN, Y., BELL, R. M. & VOLINSKY, C. 2009. Matrix factorization techniques for recommender systems. *IEEE Computer***,** 30–37.

KORTING, T. S. 2006. C4. 5 algorithm and multivariate decision trees. *Image Processing Division, National Institute for Space Research–INPE Sao Jose dos Campos–SP, Brazil*.

KUMAR, B. & SHARMA, N. 2016. Approaches, Issues and Challenges in Recommender Systems: A Systematic Review. *Indian Journal of Science and Technology,* 9.

LATHIA, N., HAILES, S. & CAPRA, L. Trust-based collaborative filtering. IFIP International Conference on Trust Management, 2008. Springer, 119-134.

LIN, W., ALVAREZ, S. & RUIZ, C. 2001. Efficient Adaptive{Support Association Rule Mining for Recommender Systems.

LIU, J., JIANG, Y., LI, Z., ZHANG, X. & LU, H. 2016. Domain-sensitive recommendation with user-item subgroup analysis. *IEEE Transactions on Knowledge and Data Engineering,* 28**,** 939-950.

MAHMUD, M. S., RAHMAN, M. M. & AKHTAR, M. N. Improvement of K-means clustering algorithm with better initial centroids based on weighted average. Electrical & Computer Engineering (ICECE), 2012 7th International Conference on, 2012. IEEE, 647-650.

MANNING, C. D., RAGHAVAN, P. & SCHÜTZE, H. others. 2008. Introduction to information retrieval. Vol. 1. Cambridge university press Cambridge.

MASSA, P. & AVESANI, P. Trust-aware recommender systems. Proceedings of the 2007 ACM conference on Recommender systems, 2007. ACM, 17-24.

MASSA, P. & AVESANI, P. 2009. Trust metrics in recommender systems. *Computing with social trust***,** 259-285.

MIN, S. & HAN, I. 2005. Detection of the Customer Time-Variant Pattern for Improving Recommender Systems. *Expert Syst.Appl,* 28**,** 188-199.

MYRBERG, J. 2016. A Recommender System for an Online Auction.

NELSON, R., 2013. Probability, stochastic processes, and queueing theory: the mathematics of computer performance modeling. Springer Science & Business Media.

O'DONOVAN, J. & SMYTH, B. Trust in recommender systems. Proceedings of the 10th international conference on Intelligent user interfaces, 2005. ACM, 167-174.

PAPAGELIS, M., PLEXOUSAKIS, D. & KUTSURAS, T. 2005. Alleviating the sparsity problem of collaborative filtering using trust inferences. *Trust management***,** 125-140.

PATEL, V. R. & MEHTA, R. G. Performance analysis of MK-means clustering algorithm with normalization approach. Information and Communication Technologies (WICT), 2011 World Congress on, 2011. IEEE, 974-979.

PITSILIS, G. & MARSHALL, L. F. 2004a. *A model of trust derivation from evidence for use in recommendation systems*, University of Newcastle upon Tyne, Computing Science.

PITSILIS, G. & MARSHALL, L. F. 2004b. A model of trust derivation from evidence for use in recommendation systems. University of Newcastle upon Tyne. *Computing Science*.

POLATIDIS, N. & GEORGIADIS, C. 2013. Mobile Recommender Systems: An Overview of Technologies and Challenges. *In Informatics and Applications (ICIA)*.

POWERS, D. M. 2011. Evaluation: from precision, recall and F-measure to ROC, informedness, markedness and correlation.

RAJPUT, A., AHARWAL, R. P., DUBEY, M., SAXENA, S. & RAGHUVANSHI, M. 2011. J48 and JRIP rules for e-governance data. *International Journal of Computer Science and Security (IJCSS),* 5**,** 201.

RICCI, F. 2011. Mobile recommender systems. *International Journal of Information Technology and Tourism* 12.

RICCI, F., CAVADA, D., MIRZADEH, N. & VENTURINI, A. 2006. Case-based travel recommendations. 50-55.

RICCI, F. & NGUYEN, Q. N. 2005. Critique-based mobile recommender systems. *OEGAI Journal,* 24**,** 1-7.

RICCI, F. & NGUYEN, Q. N. 2007. Acquiring and revising preferences in a critique-based mobile recommender system. *IEEE Intelligent Systems* 22–29.

RIEDL, K. & RIEDL, J. 2013. Recommender systems for fast paced users. *A Peaceful Passing*.

SARWAR, B., KARYPIS, G., KONSTAN, J. & RIEDL, J. 2001a. Item-Based Collaborative Filtering Recommendation Algorithms. *ACM,* 1.

SARWAR, B., KARYPIS, G., KONSTAN, J. & RIEDL, J. Item-based collaborative filtering recommendation algorithms. Proceedings of the 10th international conference on World Wide Web, 2001b. ACM, 285-295.

SCHAFER, J. B., FRANKOWSKI, D., HERLOCKER, J. & SEN, S. 2007a. Collaborative Filtering Recommender Systems. *The Adaptive Web*.

SCHAFER, J. B., FRANKOWSKI, D., HERLOCKER, J. & SEN, S. 2007b. Collaborative filtering recommender systems. *The Adaptive Web.* Springer.

SCHÜTZE, H., MANNING, C. D. & RAGHAVAN, P. 2008. *Introduction to information retrieval*, Cambridge University Press.

SETTEN, M. V., POKRAEV, S. & KOOLWAAIJ, J. 2004. Context-Aware Recommendations in the Mobile Tourist Application COMPASS. *n Proceedings of the 4th International Conference on Adaptive Hypermedia and Adaptive Web-Based Systems***,** 235-244.

SHAMBOUR, Q. & LU, J. 2012. A trust-semantic fusion-based recommendation approach for e-business applications. *Decision Support Systems,* 54**,** 768-780.

SHAMS, B. & HARATIZADEH, S. 2016. Graph-based Collaborative Ranking.

SHARMA, N., BAJPAI, A. & LITORIYA, M. R. 2012. Comparison the various clustering algorithms of weka tools. *facilities,* 4.

SHARMA, N. & SHARMA, A. CLASSIFICATION OF CONTINUOUS DATA-A SURVEY.

SHEPPERD, M. & KADODA, G. 2001. Comparing software prediction techniques using simulation. *IEEE Transactions on Software Engineering,* 27**,** 1014-1022.

SHISHEHCHI, S., BANIHASHEM, S. Y., ZIN, N. A. M. & NOAH, S. A. M. Review of personalized recommendation techniques for learners in e-learning systems. Semantic

Technology and Information Retrieval (STAIR), 2011 International Conference on, 2011. IEEE, 277-281.

SHUKLA, S. K., RUNGTA, S. & SHARMA, L. K. 2012. Self-organizing map based clustering approach for trajectory data. *International Journal of Computer Trends and Technology (IJCTT),* 3**,** 321-324.

SINGH, M., KAUR, K. & SINGH, B. 2008. Cluster algorithm for genetic diversity. *World Acad Sci Eng Technol,* 2**,** 432-436.

SMITH, J. E. & BROWN, A. M. 2005. Building a culture of learning design: Reconsidering the place of online learning in the tertiary curriculum.

TAKAMURA, H. & MATSUMOTO, Y. 2003. Co-clustering for text categorization. *nformation Processing Society of Japan Journal*.

TERVEEN, L. & HILL, W. 2001. Beyond recommender systems: Helping people help each other. *HCI in the New Millennium,* 1**,** 487-509.

TREWIN, S. 2000. Knowledge-based recommender systems. *Encyclopedia of library and information science,* 69**,** 180.

TUNG, W. & SOO, W. 2004. A Personalized Restaurant Recommender Agent for Mobile E-Service.

VIRMANI, D., TANEJA, S. & MALHOTRA, G. 2015. Normalization based K means Clustering Algorithm. *arXiv preprint arXiv:1503.00900*.

VOZALIS, E. & MARGARITIS, K. G. Analysis of recommender systems algorithms. The 6th Hellenic European Conference on Computer Mathematics & its Applications, 2003. 732-745.

WANG, C.-Y., WU, Y.-H. & CHOU, S.-C. T. 2010. Toward a ubiquitous personalized daily-life activity recommendation service with contextual information: a services science perspective. *Information Systems and E-Business Management,* 8**,** 13.

WOERNDL, W., SCHUELLER, C. & WOJTECH, R. 2007. A Hybrid Recommender System for Context-aware Recommendations of Mobile Applications.

XIA, Y. & XI, B. Conceptual clustering categorical data with uncertainty. Tools with Artificial Intelligence, 2007. ICTAI 2007. 19th IEEE International Conference on, 2007. IEEE, 329-336.

XING, E. P., JORDAN, M. I., RUSSELL, S. J. & NG, A. Y. Distance metric learning with application to clustering with side-information. Advances in neural information processing systems, 2003. 521-528.

YANG, W.-S., CHENG, H.-C. & DIA, J.-B. 2008. A location-aware recommender system for mobile shopping environments. *Expert Systems with Applications,* 34.

YEUNG, K. F., YANG, Y. & NDZI, D. 2012. A proactive personalised mobile recommendation system using analytic hierarchy process and Bayesian network. *Journal of Internet Services and Applications,* 3**,** 195-214.

YUAN, W., SHU, L., CHAO, H.-C., GUAN, D., LEE, Y.-K. & LEE, S. 2010. ITARS: trust-aware recommender system using implicit trust networks. *IET communications,* 4**,** 1709-1721.

ZHOU, B. & YAO, Y. 2010. Evaluating information retrieval system performance based on user preference. *Journal of Intelligent Information Systems,* 34**,** 227-248.

ZIEGLER, C.-N., MCNEE, S. M., KONSTAN, J. A. & LAUSEN, G. Improving recommendation lists through topic diversification. Proceedings of the 14th international conference on World Wide Web, 2005. ACM, 22-32.

ZUVA, T. 2012. Image content in shopping recommender system for mobile users.