

A Dissertation
On

**" Analysis of Algorithms to counter Long Tail Issue in
Recommender System "**

Submitted in partial fulfillment of the requirement
for the award of degree of

MASTER OF TECHNOLOGY
Software Engineering
Delhi Technological University, Delhi

SUBMITTED BY

Nitin Sodera
2K15/SWE/11

Under the Guidance of

DR. AKSHI KUMAR

Assistant Professor
Department of Computer Science & Engineering
Delhi Technological University



DEPARTMENT OF

COMPUTER SCIENCE

& ENGINEERING
DELHI TECHNOLOGICAL UNIVERSITY
2017

CERTIFICATE

This is to certify that the dissertation entitled “**Analysis of Algorithms to counter Long Tail Issue in Recommender System**” has been submitted by **Nitin Sodera (Roll Number: 2K15/SWE/11)**, in partial fulfillment of the requirements for the award of Master of Technology degree in Software Engineering at **DELHI TECHNOLOGICAL UNIVERSITY**. This work is carried out by him under my supervision and has not been submitted earlier for the award of any degree or diploma in any university to the best of my knowledge.

(DR. AKSHI KUMAR)

Project Guide

Assistant Professor

Department of Computer Science & Engineering

Delhi Technological University

ACKNOWLEDGEMENT

First of all I would like to thank the Almighty, who has always guided me to work on the right path of the life. My greatest thanks are to my parents who bestowed ability and strength in me to complete this work.

I owe a profound gratitude to my project guide **Dr. Akshi Kumar** who has been a constant source of inspiration to me throughout the period of this project. It was her competent guidance, constant encouragement and critical evaluation that helped me to develop a new insight into my project. Her calm, collected and professionally impeccable style of handling situations not only steered me through every problem, but also helped me to grow as a matured person.

I am also thankful to her for trusting my capabilities to develop this project under her guidance.

I would also like to express gratitude to Mrs. Arunima Jaiswal (Research Scholar, Delhi Technological University) for providing me continuous support and guidance during this project.

I would also like to express my gratitude to the university for providing us with the laboratories, infrastructure, testing facilities and environment which allowed us to work without any obstructions.

Nitin Sodera
2K15/SWE/11

DECLARATION

I hereby declare that the work entitled “**Analysis of Algorithms to counter Long Tail Issue in Recommender System**” which is being submitted to Delhi Technological University, in partial fulfilment of requirements for the award of degree of Master of Technology (Software Engineering) is a bonafide report of thesis carried out by me. The material contained in the report has not been submitted to any university or institution for the award of any degree.

Nitin Sodera
2K15/SWE/11

ABSTRACT

Today's world is all about information, with most of it online which enables anytime, anywhere, easy and unlimited access; participation & publishing of information has consequently escalated the suffering of 'Information Glut'. Such increase in data leads to information overload, thus creating a high level of stress and chaos. So, as to save the person from this misperception and in order to make the surfing practice better, RS was introduced. Assisting users' informational searches with reduced reading or surfing time by extracting and evaluating accurate, authentic & relevant information are the primary concerns in the present milieu. The recommendation system is defined as the software technology/tools that make relevant suggestions to a user. Nowadays, the most prominent problem while making a recommender system is a cold start and long tail problem. Where we deal with new and rare data items thus creating sparsity in the dataset. Which in turns leads to suggesting same items to the user again and again. Thus, there's a great need in dealing and evaluating various algorithms to leverage long tail recommender system.

The long Tail problem happens when we deal with relatively rare item set. It is a persistent version of cold start problem. To leverage it, four different algorithms to compare and contrast long tail issue in Recommender system have been studied and implemented.

List of Figures & Tables

Figure 1. Different Approaches on recommender System.	7
Figure 2. Challenges Faced	14
Figure 3. Impact of R factor	29
Figure 4. R factor in Pearson Correlation	29
Figure 5. Example of Pearson Correlation	30
Figure 6. Decision tree Set	33
Figure 7. Entropy Value	33
Figure 8. Entropy form Single attribute	34
Figure 9. Entropy form Multiple attribute	34
Figure 10. Decision Tree Data Set	36
Figure 11. Decision Tree Data Set	36
Figure 12. Decision Tree Data Set	36
Figure 13. Mapping Decision tree to Decision Rules	37
Figure 14. ABC workflow	38
Figure 15. Recommendations based on ABC	40
Figure 16. Relevance Ven Diagram	45
Figure 17. Comparitive Analysis between different Algorithms	48
Table 1. Recommendation approaches and techniques	13
Table 2. Long Tail Survey	21
Table 3. Kmeans data set	23
Table 4. Kmeans data set	24
Table 5. Kmeans data set	24
Table 6. Kmeans data set	25
Table 7. Kmeans data set	25
Table 8. Pearson Correlation set	27
Table 9. Pearson Correlation set	27
Table 10. Pearson Correlation set	28
Table 11. Decision Tree set	35

Table 12. Set 1	46
Table 13. Set 2	46
Table 14. Set 3	46
Table 15. Set 4	46
Table 16. Mapping between Recommendation approaches and there corresponding challenges	49

Contents

Chapter 1. Introduction	1
1.1 Overview	1
1.2 Motivation	2
1.3 Scope	2
1.4 Algorithm Used	3
1.5 Research Objectives	5
1.6 Organization of Report	5
1.7 Chapter Summary	6
Chapter 2. Literature Review	7
2.1 Approaches of Recommender System	7
2.2 User's Aspect based Recommender System	7
2.3 Approaches used based Recommender System	8
2.4 Knowledge Based Recommender System	11
2.5 Domain Specific Context-Based/ Time-Specific/ Location Based Recommender System	12
2.6 Issues	13
2.7 Long Tail	17
2.8 Related Work	17
Chapter 3. Proposed Work	22
3.1 K-means	22
3.2 Pearson Correaltion	26
3.3 Decision Tree	30
3.4 Artificial Bee Colony	37
Chapter 4. Implementation	41
4.1 Data Set Collection	41
4.2 Algorithm	42
4.3 Environment & Experimental Settings	42
4.4 Programming Tools Used	43
Chapter 5. Result & Analysis	44
5.1 Results	44
5.2 Analysis	47
5.3 Mapping	48

Chapter 6. Conclusion	50
6.1 Research Summary	50
6.2 Future Scope	51
References	52
Appendix A	56
Appendix B	69
Appendix C	72

CHAPTER 1

INTRODUCTION

This chapter briefly introduces the research work proposed in the thesis. Section 1.1 gives an overview of the research undertaken. Section 1.2 discusses motivation behind the thesis and the scope of various algorithms in Recommender System is briefly described in section 1.3. Section 1.4 explains the algorithm used in this thesis, the Research objectives of the method is discussed in section 1.5. Section 1.6 tells about Organisation of Report. Finally, Section 1.7 gives the summary of the chapter.

1.1. OVERVIEW

Recommender systems(RS) has been a crucial research subject after the inclusion of the various demonstration on filtering systems. Despite the fact that research on recommender systems has extended extensively over the last decades, there's still a requirement in the complete evaluation of the research made till date and classification made on the RS. Therefore, in this thesis RS has been classified based on algorithms and approaches, also on the basis of the challenges faced by the recommender system. Also, four major algorithms to cater to the problem of Long Tail issue in Recommender System has been studied and implemented in this thesis. The long Tail problem happens when we deal with relatively rare item set. It is a persistent version of cold start problem. To leverage it, four different algorithms to compare and contrast long tail issue in Recommender system have been studied and implemented. The four algorithms: Decision tree, Pearson Correlation, K-Mean, Artificial Bee Colony are then implemented and compared so as to give the clear view of the effectiveness of algorithms in dealing with Long Tail issue. The RS hence is categorized into four techniques as per the evaluation, i.e.; collaborative filtering (CF), content based filtering hybrid filtering, and Demographic. These studies give statistical analysis and recent trends in RS, giving researchers and practitioners, a perception on the recommender system. Also, various ways to leverage long tail problem have been discussed and computed efficiently. Hopefully, this thesis enables everyone interested in recommender systems research with an insight about the trend being followed.

1.2 Motivation

With the great improvement in technology (IT), there's exponential increase in the inflow of products, in every domain via E-market. Although it is easier for a consumer to choose from a small set of items when the item set increases exponentially, it is cumbersome and difficult for a user to consider various properties of the alternative product. Following these conditions, the user wants recommendations from the known users who already have information regarding the product. We currently live in an era where there is overloaded info. There's a plethora of information from the articles, blogs, and talks on several social networking websites. Users who use the internet have been through 40% hike ever since 1995 and touched a net count of 3.2 billion in such a short span of time. Such increase in data leads to information overload, thus creating a high level of stress and chaos. So, as to save the person from this misperception and in order to make the surfing practice better, RS was introduced. The recommendation system is defined as the software technology/tools that make relevant suggestions to a user. Nowadays, the most prominent problem while making a recommender system is a cold start and long tail problem. Where we deal with new and rare data items thus creating sparsity in the dataset. Which in turns leads to suggesting same items to the user again and again. Thus, there's a great need in dealing and evaluating various algorithms to leverage long tail recommender system. Hence it provides perfect motivation, to work towards resolving this issue and provides researchers interested in Recommender system with an all in one thesis focusing on all types of Recommender system and challenges faced. Therefore, it required detailed study and implementation of different algorithms to resolve the long tail issue. Also, Mapping between different types of Recommender systems and the challenges have never been covered before, thereby adding more credibility to the thesis. The long tail issue in Recommender system is one of the biggest issues, therefore we covered and researched the work done in Long Tail issue of Recommender system which is summarized here. More than 32% of users rate a product online, over 33% writes the positive reviews and nearly 88% users trust online reviews. Usually, RS suggests products that a user might find valuable, thus helping both the recipient and the seller.

1.3 Scope

With increasing information accessible electronically, the need for effective information filtering tools and information retrieval have become essential. RS are software techniques and tools predicting products and/or facilities to be of liking to a particular consumer. They are

achieving enormous triumph in e-commerce applications during the past few years. This thesis deals with summarizing and detailing Recommender systems and focuses on Long Tail issue of Recommender System. Therefore, in this thesis, various techniques and algorithms to leverage long tail Recommender systems have been discussed. The four major techniques are hereby evaluated and compared based on accuracy, precision, recall, and f-score. So as to give the clear idea of superiority of one algorithm over another and thus helping in getting a better understanding of procedure to leverage long tail issue in the Recommender system. This thesis gives a categorical analysis in the field of RS describing the state-of-the-art survey of RS classified into four broad categories: collaborative filtering (CF), content based filtering, hybrid filtering, and Demographic. Various pros and cons of the all the categories along with the trustworthiness of the RS have been discussed here. This work also put forward the categorical reviews on the long tail crisis of the recommender system and how it has been dealt till now. In this thesis, we are going to discuss the types of the recommender system in detail. We also discuss the challenges faced and finally a state-of-the-art analysis of the work done to handle the long tail problem in Recommender System. Also, algorithms to leverage Long Tail Problem are efficiently and effectively analysed, which are then executed and evaluated. So, that this thesis can serve all the aspiring Researchers in the field of Recommender System to get each and every detail. This thesis is allowing the experts to analyse the algorithms to leverage the long tail issue in the Recommender system.

1.4 Algorithms used

In order to leverage long tail issue in Recommender Systems, many different algorithms and techniques have been studied and detailed in this thesis. Out of which, the thesis discusses in depth about these four algorithms and thus measuring the impact on the RS. The Algorithms that we analysed and executed are:

Decision tree

K-Mean

Pearson Correlation

Artificial Bee Colony Optimisation

Decision tree: Decision Tree forms a predictive model that links input data to the predicted value depending on input attributes. Each data node in the interior of the tree resembles an

attribute and every arch from the parent to a child node suggests the likely data value or the set of the data attribute. The development procedure begins from the root node with the help of the dataset given by the client. The root node is allotted a characteristic and for each set of values, arches and sub-nodes are produced. Information groups are categorized by the qualities in order to ensure that every child node receives only that part of the input set that corresponds to the attribute value as given by the child node's arc. The procedure at that point reiterates recursively for every child until the split is no longer attainable. Either a solitary grouping (anticipated value) can be connected to every component in the partitioned set, or some other limit is taken into consideration.

K-Mean: The Main logic behind K-Mean is to outline k centroids, each corresponding to a specific cluster. Then we associate each point of the data set to its nearest centroid. The preliminary step is performed and an advance grouping is done in the case of no remaining new sets, now reiterate k new centroids as new centres of cluster based on past iteration. Now a new bonding is done between the new centres and the centroid. The result of this is the generation of a loop. Due to this loop, we observe that all the k centroids differ their known location using a systematic process until no further changes can be achieved.

Pearson Correlation: This technique is used in analysing the relationship amongst 2 quantitative, continuous variables. For eg. blood pressure and age. The measure of the association between the two variables in terms of strength is called Pearson's correlation coefficient (r).

Artificial Bee Colony: The Artificial Bee Colony algorithm is similar to the foraging behaviours of bees, and duplicate the features of swarm intelligence solving the optimization issue. In general, the community of bees in a colony consists of 3 types of bees, which consists of queen functioning as the kernel, only a hand full drones are utilized for reproduction purposes, but lastly a vast quantity of workers that search for pollen and to monitor the larvae in the colony. This algorithm depends on deployed bees, which is further classified into a scout, employed, and onlooker bees. The job of the Scout bees is to look for the new food sources and make note of the quantity of nectar at each location. After the collecting of data is done by scout bees, the employed bees go to the adjacent areas of the food sources in order to find a new source and the amount of nectar they are storing. The gathering of the nectar is done by the onlooker bees who process the data about the food sources from the employed bees.

1.5 Research Objectives

The main research objectives of the work done in this thesis are:

Research objective 1 – To study the various approaches/techniques for Recommender system

Research objective 2 - To study the various challenges/issues for Recommender system.

Research objective 3 – To map the approaches/techniques to challenges/issues for Recommender system.

Research objective 4 - To leverage Long Tail Recommender System. Thereby implementing major algorithm to cater Long Tail issue in Recommender System.

Research objective 5 – Statistical Comparison between different algorithms to leverage Long Tail Recommender System.

The objective of this thesis is to find an algorithm which can be a hybrid approach to extract keywords of the articles with an improve accuracy.

1.6 Organization of Report

This thesis is structured into 6 Chapters followed by references and three appendix.

Chapter 1 presents the overview, research objectives scope and motivation of the project.

Chapter 2 Complete Literature survey on RS is done in this chapter and long tail issue in Recommender System is also described in Detail. Also, the list of issues while dealing with RS is Detailed.

Chapter 3 Different algorithms have been detailed. Their working and usage are explained along with research methodology. Also detailing on their pro's and con's.

Chapter 4 Discussion on Experimentation i.e. discussion about the infrastructure required to be set to do the further experimentation with minimum required resources. Also detailed discussion the data set used and how it is calibrated for the algorithms.

Chapter 5 The analysis and calculation of different algorithms helping in comparison based on 4 different features. Thereby determining the superiority of one algorithm over another.

Chapter 6 presents future scope and conclusions based on the contribution made by this thesis.

Appendix A contains the code snippets and **Appendix B** contains the snapshots of the system and **Appendix C** contains the list of publications.

1.7 Chapter Summary

This chapter presents the idea used in this thesis. It discusses research problem, objectives, goals and motivation for the research. Justification for the research problem is outlined, together with an explanation of the research methodology used. The next chapter describes the literature survey and relevant background work done till date in context of this thesis.

CHAPTER 2

LITERATURE REVIEW

In this Chapter classification and categorisation of Recommender system is done, also various challenges faced while dealing with Recommender system are discussed.

2.1. Approaches of Recommender System

RS are mainly utilized for the class of individuals that lack necessary experience/Information resulting in poor evaluation of a high number of alternative items provided by the seller/websites.

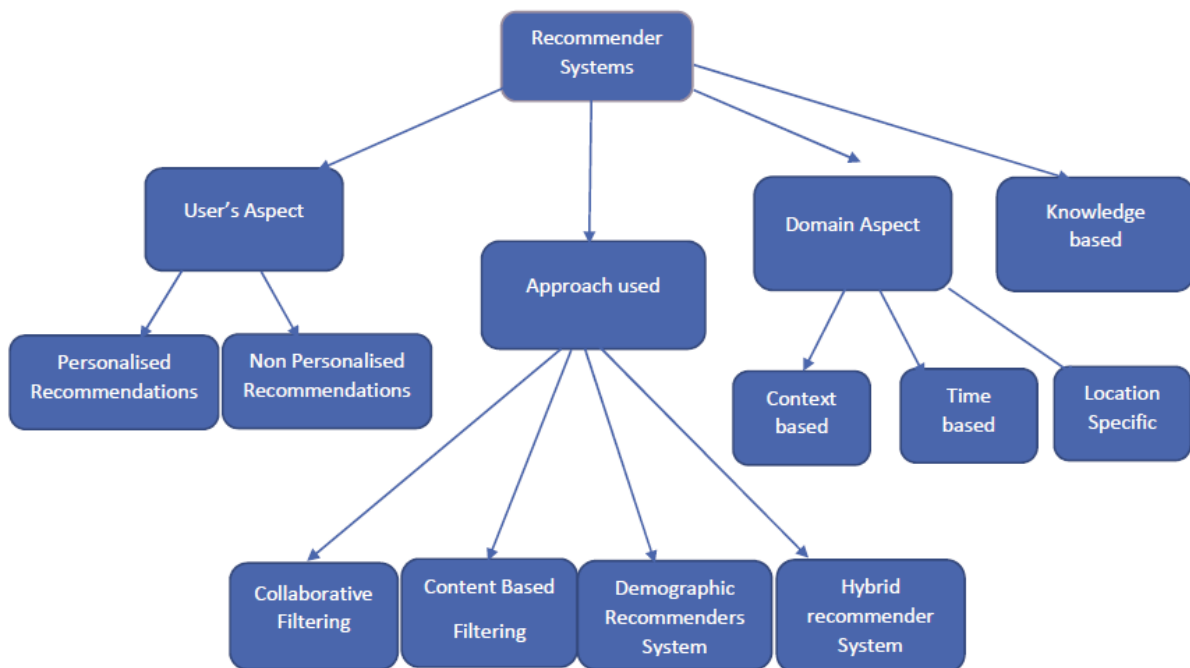


Figure 1: Different Approaches on recommender System.

2.2. User's Aspect based Recommender System

- A. **Personalised Recommendations** : These types of recommendations are given as lists of ranked products. During this task, the system tries to recommend/suggest the best matching items/services, depending on recipient's choices. The products can be

suggested depending on the ranking from a website, past analysis of user's behaviour or demographics of a user as a suggestion for next recommendations for the customer

B. Non- Personalised Recommendations : These types of RS are easier to generate in comparison to the Personalised Recommendations and are usually used in newspapers and general. These kinds of suggestions are independent of the user, therefore everyone tends to get same suggestions. They are automatically generated, on the grounds that they require little client effort to create the suggestions and are transient. These proposals are totally independent of the specific client focused by the RS or newspapers/Magazines.

2.3. Approaches used based Recommender System

A. Collaborative Recommender System : Collaborative filtering (CF) uses the numerical reviews given by the consumers and is based mainly on the historical data of the user available to the system. It makes suggestions to all dynamic consumer with data about a group of consumers and their connection with the itemset. Both the consumer profile and the item profile are used to make a recommendation system. It is considered as one of the most basic and the easiest method to give suggestions and make predictions regarding a product depending on consumer's previous behaviour and the consumer's behaviour of other like-minded consumers.

It can be further categorized as:

User-based CF: Correlation is computed between one user and others user. Also for every data-item, we calculate the ratings of the consumers that are heavily related to each consumer.

Problem: data sparsity, bad correlation, ease of getting attacked.

Item-based CF: Correlation is computed between one item and every itemset. Also for every consumer, we calculate the consumer's ratings of products that are heavily related with each data-item. Therefore in data-items, there is less sparsity, It uses cosine and Pearson correlation similarity approach. Collaborative filtering can be categorized into three main algorithms:

Memory based: Users and dataset with similar interest are combined

Model based: different techniques involving data mining and machine learning are used to determine complex patterns.

Hybrid CF: Different CF techniques and other RS' techniques are combined.

Advantages: These kinds of filtering approaches don't require representation of data in the dataset but are only based on the precision of active consumer's group. Database's scalability is large as it doesn't require manual involvement.

Disadvantages: The products can't be suggested to any consumer until the data/item is either ranked/rated by any another consumer/users or correlated with data-items of the same kind from the itemset. Usually, the persistent consumer's rate very less number of items even though, products database is very large thus leading to very sparse results. Because of changes in opinions, many users find this approach expensively costly and it also requires a lot of time. One other issue that is predominant with collaborative filtering is sparsity issue and the measures taken to resolve it which includes: implicit rating, dimensionality reduction, and content description.

Various Techniques: Unified relevance model, Hybrid CF model, Fuzzy Association Rules and Multilevel Similarity (FARMS), Flexible mixture model (FMM), Maximum entropy approach

Application: Collaborative filtering application is used to recommend befitting information as judged by the community. Collaborative filtering is usually used to work with very large data sets. It is also used in solving the nearest neighbour problem.

B. Content Based Recommender System : It focuses on the features of the items and the goal is creating a user profile depending on the previous reviews of the users and also a profile of the item in accordance with the features it provides and the reviews that have been received for that particular item from the itemset. It uses information from the itemset and knowledge from the dynamic consumers. This technique is made from the structural information of properties/content of product/item instead of a description of consumer's ratings of the particular dataset. The probability It compares the content

of items of user's interest with the content in the item list. It helps overcome sparsity problem that is faced in collaborative filtering based recommendation system.

Various Techniques: Content-Boosted Collaborative Filtering, FAB Technique, Bayesian hierarchical model(BHM).

Advantages: This approach recommend items from the dataset to the consumer and thereby don't require data of other users and also it don't face first rater problem the i.e., It recommend new items/products and rare items for each and every consumer. i.e helpful in both long tail as well as cold start problem.

Disadvantages: In these types of techniques products, knowledge is restricted to the initial descriptions/features i.e. explicit specifications of the product is done. Therefore it depends on the information provided explicitly and that too manually.

Application: Used in situations to deal with cold start problem as it is capable of recommending rare data from the itemset. Used in confidential places like banking etc.

C. Demographic Filtering System : This type of RS uses previous information of demographic knowledge regarding consumers and their views for items that were recommended as a criteria basis for suggestions, a classifier based on demographic data can be obtained by Machine learning techniques. The display of such info in a consumer's model varies to a large extent.

Advantages: It doesn't require knowledge of ratings given by consumer, which was required by the other main techniques (Content based and collaborative). The demographic approach is fast, simple and straight forward for making depending on few datasets.

Disadvantages: Compilation of full consumer info is required to get good recommendations which can't be possible. As this types of RS is dependent on consumer's field of interest, it generally results in, recommending the usual data to

consumers having the same field of demographic interest, thus resulting in too general recommendations. There are security and privacy issues.

Application: This technique is very well utilized in formulating the recommender system such as trip adviser or a party planner where the demographic information is taken into consideration and so a new user can also be recommended as it does not require the user's previous information.

D. Hybrid Recommender System: Hybrid RS is a type of RS, efficiently overcomes the limitations of other recommendations approaches. This type of techniques uses the good features of two or more approaches to gain stable and robust system and to have an efficient recommender system. Content-based and Collaborative filtering is the most common hybrid approaches. Mostly, this type of approaches uses both ratings of all users and items as attributes. Usually, such RS adapts Heuristic mixture of Content based and Collaborative filtering methods.

Various Techniques: Weighted, Switching, Mixed, Feature combination, Cascade, Feature augmentation.

Application: As it takes into consideration the best aspect of multiple Recommender systems that should practically be used to implement any type of Recommender system eg. Movie data, cab, travel advisor, Website Recommender system etc.

2.4. Knowledge Based Recommender System

Afore mentioned challenges can be tackled using this approach. The benefit of such knowledge based recommender system can be viewed as no cold start/ramp up issue persists, as no rating information is required. Recommendations are computed exclusively for every consumer's ratings: either on the basis of explicit recommendation rules or based on the common expects between user needs and product. This type of RS can be split into 2 different categories: constraint and case-based systems. The way in which they use the knowledge provided is the main difference between the two: case-based RS depends on gathering similar items using different similarity measure, while constraint-based RS rely on protocol.

Advantages: One of the biggest benefits of such a Recommender system is that cold-start (ramp-up) problems don't exist in it. The main setback is that there are potential information extraction bottlenecks, initiated by the necessity of defining information of suggestions in an explicit way. Deterministic recommendations can be extracted from knowledge based recommender system as we have assured quality. Also, it can resemble sales dialogue

Limitations: The cost of knowledge acquisition is very high from domain experts/consumers and from web resources. Knowledge engineering effort to bootstrap is quite high. This approach is basically static and it does not react to short-term trends. Independence assumption can be challenged as preferences are not always independent from each other

Application: This technique can be used to deal with long tail data set such as Recommending exotic villas to users, Poker Recommendation system.

2.5. Domain Specific Context-Based/ Time-Specific/ Location Based Recommender System

Contextual information present in a RS helps to get a clear view of the situation of any object, place or person which is of relevance to the system for suggestions and anything that can be incorporated. In this kind of Recommender system, the contextual knowledge of consumers is also taken into consideration while designing a recommender system. Context refers to the location, time, area and environment of the Consumer which define a user's state. RS requires situational information of the user and context based RS accesses the information directly using various techniques (such as GPS). The user's locational data, social data, current time, weather data are also taken into consideration as the contextual data. Contextual factors are of two types: Dynamic and static, depending on whether they change with time or not.

Various Techniques: Hidden Markov Model, Multidimensional approach, Fuzzy Bayesian Networks, Human memory model, Matrix-factorization Predictive Context Based Model

Application: Used for recommending the cab or the hotel to the user based on its current location.

Recommendation Approach	Recommendation Technique	
	Heuristic-based	Model-based
Content-based	Commonly used techniques: <ul style="list-style-type: none"> • TF-IDF (information retrieval) • Clustering Representative research examples: <ul style="list-style-type: none"> • Lang 1995 • Balabanovic & Shoham 1997 • Pazzani & Billsus 1997 	Commonly used techniques: <ul style="list-style-type: none"> • Bayesian classifiers • Clustering • Decision trees • Artificial neural networks Representative research examples: <ul style="list-style-type: none"> • Pazzani & Billsus 1997 • Mooney et al. 1998 • Mooney & Roy 1999 • Billsus & Pazzani 1999, 2000 • Zhang et al. 2002
Collaborative	Commonly used techniques: <ul style="list-style-type: none"> • Nearest neighbor (cosine, correlation) • Clustering • Graph theory Representative research examples: <ul style="list-style-type: none"> • Resnick et al. 1994 • Hill et al. 1995 • Shardanand & Maes 1995 • Breese et al. 1998 • Nakamura & Abe 1998 • Aggarwal et al. 1999 • Delgado & Ishii 1999 • Pennock & Horwitz 1999 • Sarwar et al. 2001 	Commonly used techniques: <ul style="list-style-type: none"> • Bayesian networks • Clustering • Artificial neural networks • Linear regression • Probabilistic models Representative research examples: <ul style="list-style-type: none"> • Billsus & Pazzani 1998 • Breese et al. 1998 • Ungar & Foster 1998 • Chien & George 1999 • Getoor & Sahami 1999 • Pennock & Horwitz 1999 • Goldberg et al. 2001 • Kumar et al. 2001 • Pavlov & Pennock 2002 • Shani et al. 2002 • Yu et al. 2002, 2004 • Hofmann 2003, 2004 • Marlin 2003 • Si & Jin 2003
Hybrid	Combining content-based and collaborative components using: <ul style="list-style-type: none"> • Linear combination of predicted ratings • Various voting schemes • Incorporating one component as a part of the heuristic for the other Representative research examples: <ul style="list-style-type: none"> • Balabanovic & Shoham 1997 • Claypool et al. 1999 • Good et al. 1999 • Pazzani 1999 • Billsus & Pazzani 2000 • Tran & Cohen 2000 • Melville et al. 2002 	Combining content-based and collaborative components by: <ul style="list-style-type: none"> • Incorporating one component as a part of the model for the other • Building one unifying model Representative research examples: <ul style="list-style-type: none"> • Basu et al. 1998 • Condliff et al. 1999 • Soboroff & Nicholas 1999 • Ansari et al. 2000 • Popescul et al. 2001 • Schein et al. 2002

Table 1: Recommendation approaches and techniques

2.6. Issues

Perhaps the biggest issue in having a good recommender system can be stated as they require a large amount of itemset to efficiently give suggestions. As a result, the companies with a lot of consumer data have excellent and accurate recommendations: Google, Amazon, Netflix, Last.fm. Firstly, an accurate recommender system requires itemset (from a catalogue or by any other way), then it ought to incorporate and refer to user dataset (behavioural events), following

which the algorithm is implemented on the analyzed dataset. The larger item and user dataset to work on, the more are the chances of having good and accurate recommendations/suggestions. Although it may also lead to chicken and egg problem I.e in order to have good suggestions, we require a lot of users, so as to achieve a large quantity of information for the recommendations there's the requirement of large number of consumers which in turn requires a good and accurate recommender system so as to attract and extract large numbers of users.

The Issues which need to be addressed in recommendation systems are:

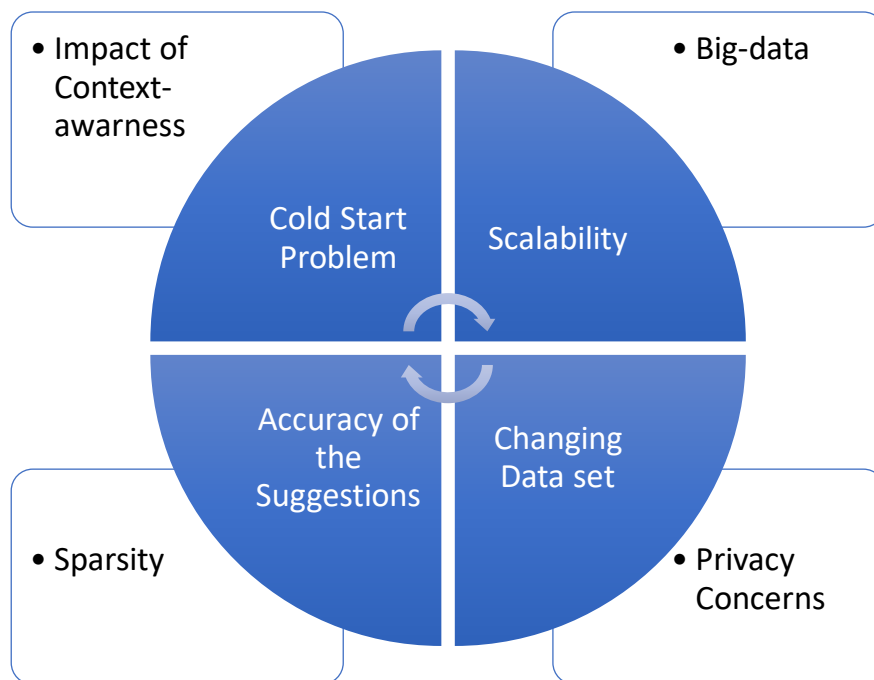


Figure 2: Challenges Faced

A. Cold Start Problem

This problem mostly telecasts at an early stage of a recommender system's life cycle/ if a rare item/product is incorporated into the dataset. If there is less knowledge available on a particular item or dataset, ontologies can be used as a tool for information extension and extraction. This problem affects every recommender system: the content-based filtering will behave poorly, provided there is less knowledge regarding the itemset. The collaborative filtering also leads to the exactly same result. If the recommender system has no previous knowledge of using the content-based methods, and also there are none of the user's behaviour history in the database/itemset, the hybrid approach is ought to produce almost random recommendations.

B. Long Tail:

Approx. \$1M prize given by Netflix to the 3rd party to give away CF algorithm improving there RS by 10%, noticeably there was an issue with eccentric movies.

C. Scalability and big data:

This is an equally important issue in RSs. As databases related to rating increases, the performance declines exponentially. A system that can handle vast dataset and give correct suggestions hastily are required. The exchange between performance and the likelihood accurateness is very common. For e.g., clustering technique increases performance, although it results in a decrease of the correctness. Matrix factorization methods are not at all applicable on online recommendations comprising of large itemsets. This algorithm runs on the Netflix Prize competition dataset takes 8 hours to complete the process. Algorithm "Gellyfish", having parallelization technique decreased time by 3min for Netflix Prize competition dataset. Algorithms' parallelization is a way to solve this problem.

D. Accuracy of the Suggestions

Along with various other specs., the user is responsible for false negatives (incorrect recommendations, which are not appreciated by the user) which lead to low accuracy in the recommender system. Following such circumstances, users lose trust in the RS so the quality and accuracy should be kept at its best.

E. Changing Data Set

With an increase in a number of items and dataset day by day, there is a constant change in the structure of the itemset by the constant inclusion of new data in the previously defined itemset. An algorithmic routine will see it as difficult but not unattainable to carry on with the changing dataset. Most users that are not active faces a great issue. They rely on trusted user and groups to recommend and suggest them the new items from the given dataset.

This issue can be stated as, biased towards the old and difficult to incorporate new.

F. Changing User Preferences (Context Awareness)

A user being the intaker needs to get details about different types of data from the single contiguous dataset. An authentic example is that on any given day a user will be using Amazon for electronics, but the next day the same user will be on Amazon searching for a gym kit.

G. Loss of neighbor transitivity:

Situations with the Transitive nature are usually not taken into consideration in the Recommender system. Let us assume that user 1 is highly related to user 2, which in turn is highly correlated with user 3. Also, user 3 can, in turn, be highly related to user 1. Such relationships cannot be seen by the recommender systems, although it can be done with information from consumer such as ontology. Like, user aggregating 75-100 are correlated as intelligent ones, whereas users aggregating 35-50 as an average one.

H. Sparsity:

It's very usual that user commonly buy or rate relatively fewer items in context to the total itemset which in turn forms a sparse users-items represented with matrix and, thereby making it difficult to locate neighbours or derive common behaviour patterns resulting in low accuracy system. Latent factor models algorithms can be used to address this issue, which makes use of dimensionality reduction of various users or items leading to pattern detection in a dense place.

I. Privacy:

Personal data collected by RS should be kept safe and piracy must be neglected and should be uninfluenced and unmodified. There are three aspects to be taken care of :-

- Value and risk of personal information
- Shilling
- Distributed Recommender System

J. Value and risk of personal information

we need to determine when to stop collecting the info to balance the privacy and to intelligently choose which info is to be discarded and which to keep. Aforementioned privacy protection may make shilling easier for distributed RS, security Techniques such as Cryptosystem and zero knowledge are to be used to counter different security and privacy attacks.

2.7 Long Tail

Long-tail phenomena are ubiquitous in real world applications, challenging the task of information trustworthiness estimation. Sources with very few suggestions and items in the dataset are common in applications. Such low number of items(rare items) and suggestions exhibited by the user typically exhibits long-tail phenomenon, i.e, most of the users only provide data about one or two items, and there are only a few users that make lots of suggestions. For example, There are numerous sites containing info/knowledge about one or many celebrities, there are few sites which, like Amazon, Wikipedia, provide extensive coverage for thousands of celebrities. On average, participants show suggestions to few items whereas very few users cover most of the items. Zied Zaier et al. introduce a long tail theory along with its effects on RS. They provide a review on the various itemsets considered for examining and evaluating collaborative filtering recommender systems algorithms alongside its techniques. Further note that the performance of different recommendation techniques based on different itemset was also compared. The study analyses only one criterion for the itemsets rating which is similar to all of the present collaborative filtering recommender systems.

2.7 Related Work

The following table summarise all the major Works done in the field of Recommender System

<u>S.N</u> <u>Q</u>	<u>Year</u>	<u>TITLE</u>	<u>AUTHOR</u>	<u>METHOD</u>	<u>DATA SET</u>
1	Apr,2016	Categorizing Long Tail SEO spam on cloud web hosting services	Elaine shi, Seerang Hao, Raheem Beyah	BlackhatSEO TECHniques measures & analyse SEO Spam	15,774 Cloud directories over 10 cloud platforms
2	Feb,2016	Long Tail Vocabulary Dictionary Extension	zheChan, Michael Carferella,H.v Jagadish	LYRETAIL'S software Architecture:Web page Fetching, Dictionary aggregation	LYRETAIL'S data set Clue Web 09 crwaler

3	Nov,20 14	Confidence Awareness approach for truth discovery on longtail	QiLi, YarianLi, Jing Gao, LuSu, Bo Zhao	Chi-Squared distribution, esti mation based on confidence interval of sore reliability	City Population dataset(wiki's edit history of city's population Biography Dataset(wikip edia)
4	Aug,20 14	Discovering unique Interest with latent variable model in large scale Social Ecommerce	Diane Hu, Rob Hall, Josh Attenberg	LDA (Latent dirichlet Allocatioon)and latent variable method	Log from amazon
5	Oct,20 13	Trading off among accuracy, Longt ail	Lie Shi	1 st order amrkovian graph with transition problem between user-item pairs OSO-NMTF modelling, User-Item biclustering	Movie Lens and Last.fm
6	Dec,20 12	Double Ranking Stratergy(Long tail for top N recommendatio ns)	Mizhang, neol Hurley<wei Lei, XiangYong Xue	Compared: 1. Cosine 2. PearsonCorrela tion and scaled cosine using jacaard Similarities(SR algo set)	Movie lens and yahoo movies dataset

7	Apr,20 12	Challenging the long tail recommender system	Hongui Yin,Jingli,chenchen ,juigie Yao	1.Hitting time algo based on user item graph 2.entropy cost model 3.LDA (Latent dirichlet Allocation)based model	Custom data set
8	Oct,20 11	Finding images of difficult entities	Bilyama Taneva, Mouna Kacirem, Gwrhurd Weikum	Comparison between KL Divergence based model and NDGC &MRR based model	Wikipedia seed page
9	Oct,20 11	My Head is your tail	Kibe on Lee and kyogun lee	Collaborative filtering Methods:link analysis method	Online music store(custom web crawler from last.fm)
10	Mar,20 11	Recommender system for long tail through Term Query graph	Franseco Bonchi,Raf Faele Perego,Fabrizio Silverstone	Term Modeland query model	Query log from yahoo
11	June,20 10	Long Tailed recommendation in grid complex network	Lovro Ilijasic,Lorenzo Saitta	Modelling Grid as Complex network	EGEE jobs
12	Feb,20 10	Anatomy of Long Tail	Sharad Goel,Andrie Broder,Evgening,Bo Pang	Null hypothesis model, Customer Behaviour:- Independent	Movies:- Movie Rental Service,Netflix

				model,sticky model,shared inventory model	Music:- Yahoo music Search:- Yahoo Search Browsing:We b Browsing by Nelson company
13	Dec,20 08	Long Tail Recommendati on Using Info Duffusion theory	Mansauki Ishikava,Peter Geezy,Naliwki Izumi	Analytical result of Info diffusion Charactersticcs	AmZon's ,Netflix's long taol datasat
14	Oct,20 08	Long tail recommenders ystem and how to Levarage it	Yoon-Too Park, Alexendra Tuzilin	Exception Maximinization and clustering:base version Total clustering (TL) Clustering Tail(CT) Each Item(EI):Used mainly	Movie lens and book Crossing
15	Aug,20 08	Info extraction from Wikipedia	FeiWu,Raphael Hoffman,Dawel S,weld	1.Shrinking training data set using KOG ontology 2.Retaining for Improving Training Data	Wikipedia

				usingText Runner 3.Extracting from Border web	
16	Nov,20 02	Fitting of long tail data set into Phase type Distribution	Alma Riska, Vesseline Dev, Evegenice Smirni	Divide and conquer and then using exception maximization	Sizes of req. by 1998 world Soccer Web

Table 2: Long Tail Survey

CHAPTER 3

PROPOSED WORK

This chapter briefly introduces the research work proposed in the thesis. In this chapter, there is an in depth discussion about all four algorithms. Their work flow is discussed along with an example for each Algorithm.

3.1 K-means

This algorithm [1967, MacQueen] is the easiest unsupervised learning algorithm which resolves the famous problems of clustering. This method is based on a straightforward and simple approach to group a provided collection of data using a specified number of clusters (accept k groups/clusters) as encountered from the previous. The fundamental idea is to categorize k centroids, one per cluster. These centroids should be ideally set in respect to numerous areas causing the diverse outcome. Therefore, the ideal way is to set them distinctively distant from one another. The previous step is to take every point and place them to nearest declared point of data set and also allocate it to the nearest centroid. When no point is remaining, the starting step is completed and an advance groupage is done. Then we try to re-figure k new centroids as barycentre's from the group containing the previous sets. When we obtain the K-new centroids, perform coupling between similar focuses of dataset and approximate the new centroid. An infinite loop would be created. Therefore, in this loop we notice that the k centroids deflect their area until nothing can be changed more. As such centroids become quite stable. Lastly, we calculate for minimizing the objective function, in present situation a squared error function. The objective function,

$$J = \sum_{j=1}^k \sum_{i=1}^n \|x_i^{(j)} - c_j\|^2, \text{ where } \|x_i^{(j)} - c_j\|^2 \text{ gives the distance from a data point } x_i^{(j)} \text{ to}$$

the cluster center whereas, is a measure of the path coverage between n data points and their respective cluster centers.

Following are the steps to implement an algorithm:

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.

It can be shown using proof that the procedure will definitely stop, but the k-means algorithm may not always give the most optimal configuration, in response to the global objective function minimum. The algorithm is also largely dependent to the initial randomly selected cluster centers. This effect can be greatly diminished by running various simulations of the k-means algorithm.

The applications of K-means has been adapted to incorporate various problem domains. As we can see, for an extension it is a good candidate with fuzzy feature vectors. k-Means: Example explaining the whole process step-by-step. Starting with an easy example of a k-means algorithm, we will consider the after mentioned data set containing the score of 2 variables on all the 7 individuals:

Subject	A	B
1	1.0	1.0
2	1.5	2.0
3	3.0	4.0
4	5.0	7.0
5	3.5	5.0
6	4.5	5.0
7	3.5	4.5

Table 3: Kmeans data set

The before mentioned data cluster is now organized into two sets. For the 1st step, we find a suitable starting partition, suppose A & B are the values of the 2 individuals most distant from each other (making use of the Euclidean distance measure), and therefore gives the definition of the initial cluster means, making:

	Individual	Mean Vector (centroid)
Group 1	1	(1.0, 1.0)
Group 2	4	(5.0, 7.0)

Table 4: Kmeans data set

The individuals which are left are then examined in order and assigned to the nearest cluster/group, by calculating the Euclidean distance from the mean of the cluster. Every time an unknown item object is included in the data set, the mean is recalculated. This gives way to the following sequence of steps:

	Cluster 1		Cluster 2	
Step	Individual	Mean Vector (centroid)	Individual	Mean Vector (centroid)
1	1	(1.0, 1.0)	4	(5.0, 7.0)
2	1, 2	(1.2, 1.5)	4	(5.0, 7.0)
3	1, 2, 3	(1.8, 2.3)	4	(5.0, 7.0)
4	1, 2, 3	(1.8, 2.3)	4, 5	(4.2, 6.0)
5	1, 2, 3	(1.8, 2.3)	4, 5, 6	(4.3, 5.7)
6	1, 2, 3	(1.8, 2.3)	4, 5, 6, 7	(4.1, 5.4)

Table 4: K-means data set

As we have now changed the initial partition, the characteristics of the 2 clusters are as follows:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2, 3	(1.8, 2.3)
Cluster 2	4, 5, 6, 7	(4.1, 5.4)

Table 5: K-means data set

But we have no way of checking that every individual was appointed to the correspondingly perfect cluster. Thereby we do the comparison between the mean of each cluster and every other cluster. Our observations are:

Individual	Distance to mean (centroid) of Cluster 1	Distance to mean (centroid) of Cluster 2
1	1.5	5.4
2	0.4	4.3
3	2.1	1.8
4	5.7	1.8
5	3.2	0.7
6	3.8	0.6
7	2.8	1.1

Table 6: K-means data set

Only item value "3" is closest to the average of the group 'Cluster Two', when compared to it's own 'Cluster 1'. Thereby, every itemset's distance from its own group average has to be less than the path to the opposite's group's average. , Thereby, every single itemset 3 is now allocated to Cluster/group 2. Thus resulting in a new partition:

	Individual	Mean Vector (centroid)
Cluster 1	1, 2	(1.3, 1.5)
Cluster 2	3, 4, 5, 6, 7	(3.9, 5.1)

Table 7: K-means data set

In the example explained above, every single item is currently closer to its own group mean rather than the mean of the opposite cluster and therefore iteration terminates, thereby selecting the last partition as the result.

As explained earlier, there is a possibility that the k-means algorithm won't be able to pinpoint any final solution. In these type of cases, we could follow the strategy of choosing to stop the algorithm when a pre-chosen maximum of iterations has been reached.

3.2 Pearson Correlation

The Pearson correlation coefficient is quite a useful statistical formula which gives the measure of strength between variables and relationships. This formula is also known to as the Pearson R test, in the field of statistics. While holding a statistical test between two variables, the best way is to conduct a Pearson correlation coefficient value to find out how good the given association is between those 2 variables.

It is a way of checking matching and contrasting the association between two quantities or continuous variables, for eg. , time duration and bp. Pearson's correlation coefficient denoted by r gives a way to calculate the potency of the relationship between the two variables.

Correlation amongst the set of data is an indicator of how closely are they related. The significant prevalent indicator of the correlation in figures is none other than the Pearson Correlation. The abbreviation PPMC stands for Pearson Product Moment Correlation. It gives the straight line relationship amongst two clusters of data. The Pearson correlation can be represented by using the two letters: The letter "r" for a sample and the Greek letter rho (ρ) for a population.

$$r = \frac{n(\sum xy) - (\sum x)(\sum y)}{\sqrt{[n\sum x^2 - (\sum x)^2][n\sum y^2 - (\sum y)^2]}}$$

....(1)

Also the constant can be evaluated by a graphing calculator or by hand [TI-89]

If we want to determine the relationship between two variables in the form of strength, a formula should be used to give what is called as the coefficient value. The coefficient value may fall from 1.00 to -1.00. If it falls in non-positive range, then this signifies that the association amongst them is negatively correlated, which means that the values are inversely

proportional. If it falls in the positive category, then this signifies that the association amongst them is positively correlated, which means that the values are directly proportional. Now focusing on the formula to calculate the Pearson correlation coefficient value.

Step one: Construct a graph with the information for two factors, namely factor (x) and (y), and include three extra sections marked (x^2) , (y^2) and (xy) . A basic information graph may resemble this:

	Person	Age (x)	Score (y)	(xy)	(x^2)	(y^2)
1						
2						
3						

Table 8: Pearson Correlation set

More information would be required, however just three specimens are appeared for motivations behind case.

Step two: Complete the diagram utilizing fundamental multiplication of the variables.

	Person	Age (x)	Score (y)	(xy)	(x^2)	(y^2)
1		20	30	600	400	900
2		24	20	480	576	400
3		17	27	459	289	729

Table 9: Pearson Correlation set

Step three: After multiplying every value to finish the chart, take sum of all the columns starting from bottom to top.

Person	Age (x)	Score (y)	(xy)	(x ²)	(y ²)
1	20	30	600	400	900
2	24	20	480	576	400
3	17	27	459	289	729
Total	61	77	1539	1265	2029

Table 10: Pearson Correlation set

Step four: Utilize the below mentioned formula to calculate Pearson correlation coefficient value

$$r = \frac{N\sum xy - (\sum x)(\sum y)}{\sqrt{[N\sum x^2 - (\sum x)^2][N\sum y^2 - (\sum y)^2]}}$$

Where:

- N = number of pairs of scores
- $\sum xy$ = sum of the products of paired scores
- $\sum x$ = sum of x scores
- $\sum y$ = sum of y scores
- $\sum x^2$ = sum of squared x scores
- $\sum y^2$ = sum of squared y scores

....Eq. (2)

Step five: After completing the formula for r by using in all the accurate values, the end product is the coefficient value. On the off chance that the value is a non-positive number, at that point there would be the negative correlation of association potency, and if the value is a positive number, at that point there is a positive correlation of association potency. Take note of: The cases just shown above contain information regarding three individuals, however, the perfect specimen configuration to compute a Pearson connection coefficient ought to be greater in size than ten individuals.

The outcomes range from - 1 to 1. You will seldom observe -1, 0 or 1. The number would be some place in the middle of those qualities. The nearer the estimation of r gets the opportunity to 0, the more prominent the variety of the information focuses would be near to the line of best fit.

High correlation : 0.5-1.0||-0.5-1.0.

Medium correlation : .3 to .5 or -0.3 to .5.

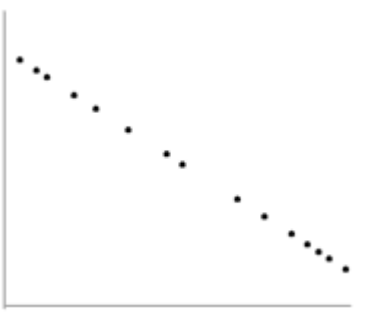

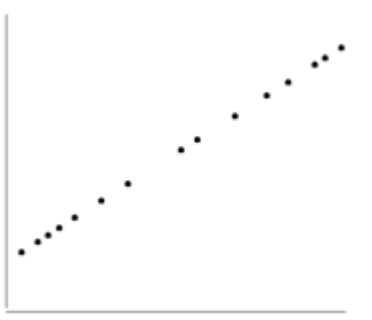
$r = -1$		Info is present on singular line marking decline slant.
$r = 0$		no direct connection between the factors.
$r = +1$		Info is present on singular line marking decline slant.

Figure 3: Impact of R factor

Low correlation : .1 to .3 or -0.1 to -0.3.

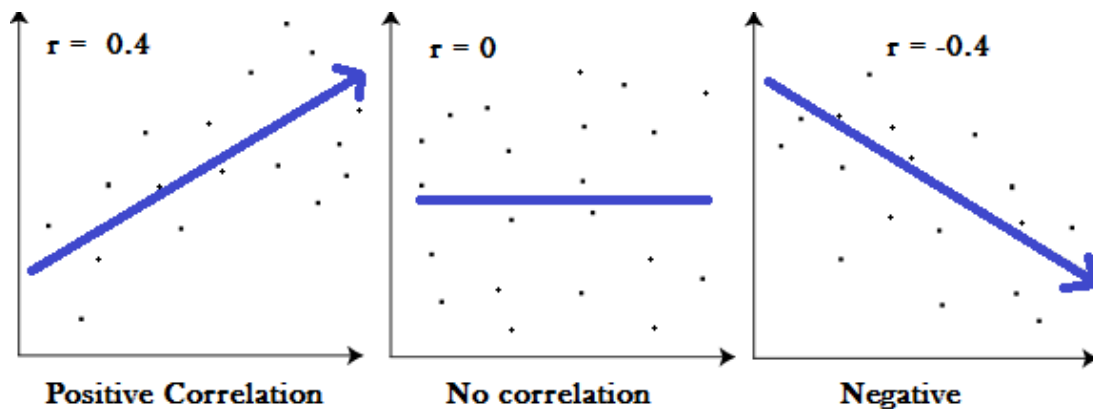


Figure 4: R factor in Pearson Correlation

Potential issues with Pearson connection:

The PPMC is not ready to differentiate amongst needy and autonomous factors. For instance, an event that you are attempting to discover the relationship between an unhealthy eating regimen and diabetes, you may locate a high connection of .8. In any case, you could likewise work out the connection coefficient equation with the factors exchanged around. At the end of the day, you could state that diabetes causes a fatty eating regimen. That clearly has neither rhyme nor reason. In this way, as a specialist, you must know about the information you are connecting to. Also, the PPMC won't give you any data about the incline of the line; It just lets you know whether there is a relationship.

Real Life Example :

Pearson relationship is utilized as a part of thousands of after-effects circumstances. For instance, researchers in China needed to know whether there was a connection between how weedy rice populaces are distinctive hereditarily. The objective was to discover the developmental capability of the rice. Pearson's connection between's the two gatherings were dissected. It demonstrated a positive Pearson Product Moment connection in range if 0.783 and 0.895 for weedy rice populaces. This figure is very high, which proposed a genuinely solid relationship.

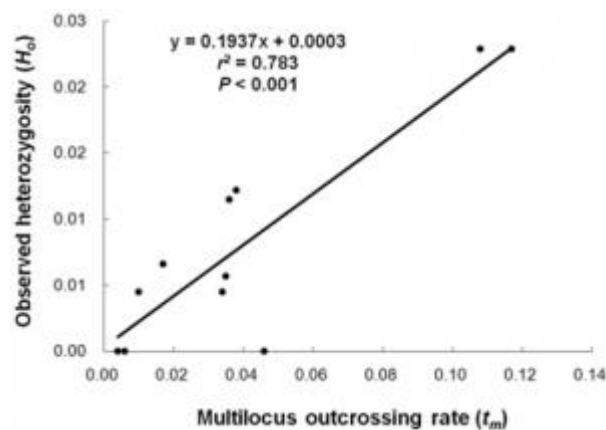


Figure 5: Example of Pearson Correlation

3.3 Decision Tree

Already it was utilized as the model-based approach for RS. Decision trees give numerous benefits for RS, for example, proficiency, adaptability, and interpretability while controlling a collection of information datasets (like demographic, ratings, contextual, and so on.). Each data node in the interior of the tree relates to an attribute and every arch from the parent to a child node, which inturns suggest the likely data value or the set of data values of that variable.

Development of the tree starts from the root node along with the dataset given by a client. Root node is allotted a characteristic and for each set of values arches, sub-nodes are produced. The information set is now part according to the qualities so that every child node receives only a particular part of the input set which satisfies the variable value as given by the arc to the child node. The procedure at that point rehashes itself recursively for every child until the part is no longer attainable. Either a solitary grouping (anticipated value) can be connected to every component in the partitioned set, or some other limit is taken into consideration.

A noteworthy shortcoming in utilizing decision trees as a forecast display in RS is provided by the necessity to construct countless trees(either for every thing or for every client). Also, the model can just figure the normal rating of a solitary thing at any given moment. To give suggestions to the client, we should cross the tree(s) starting from the root node to leaf node once for everything so as to register its anticipated rating. Simply in the wake of registering the predicted rating of all items, RS can, therefore, give the proposals (most elevated anticipated rating things). Therefore decision trees in RS don't scale well concerning the quantity of things. The development of a decision tree is done by a self-recurring procedure. The procedure begins from the root node according to the provided input set (training set). The objects quality is picked as the part characteristic. For every conceivable attribute (or set of qualities) groups are made, also the parent's set is divided into child groups in order for every child group to get info set of every thing corresponding to the proper value(s). Selecting the split-quality is done heuristically since we can't confirm the split which will deliver the ideal tree (Tree which creates the ideal outcomes for future contribution), for instance, Well known C4.5 calculation ([Qui93]) utilizes a heuristic which selects the split which delivers the biggest data increase from every single conceivable split. One of the properties is pre-characterized as the objective trait. The recursive procedure proceeds until every one of the attributes in that node's set offer a similar certified quality attribute or else the quantity of things achieves a specific limit. Each leaf node is allocated a mark (characterizing its arrangement of things), this name is the mutual target characteristic attribute or the most widely recognized as an incentive in the event that the possible number values exceed one.

Decision trees may be utilized for various recommender frameworks approaches:

- Collaborative Filtering - Breese et al. [BHK98] built the collaborative filtering system using the decision tree. Every instance of the training set the delegate to an individual client. The evaluation given by the customer for every item from the system is referred from the training set attributes. Therefore a dedicated decision tree is

constructed for every item. Therefore the decision which is required for the prediction is taken from the evaluation given for the specific item (for example dislike/like) has to be taken into account, whereas the input attributes (decision nodes) is the feedback given for all another itemset.

- Content-Based Approach - Li and Yamda [LY04] and Bouza et al. [BRBG08] decision tree are built using content features. For each user, a unique decision tree is created which is then used as a user profile. Every item's feature is required to make a model which demonstrates the user's requirements. Information gain is taken as the splitting criterion. Although the approach is good from a theoretical perspective, the precision is bad than that of most recommending the mean rating.
- Hybrid Approach - In this approach construction of only one tree takes place. It is quite alike to the collaborative approach, where it considers the user's attributes to the make the division "for eg. His/her liking/disliking" but then the attributes it uses are broad-spectrum attributes which inturn represents the user's preference for the common base, dependent on the item's content. The characteristics are built on the basis of consumer's previous ratings along with the content of the data. Such as, a consumer which rates an undesirable for all movies of a given genre say adventure is given a short value in a "degree of liking adventure movies" characteristic. Likewise for the collaborative approach, the tree built is hereby valid for every consumer. However, now it is valid for all items as the latest features denotes the consumer's likings for every items in the dataset and not only a particular provided item.

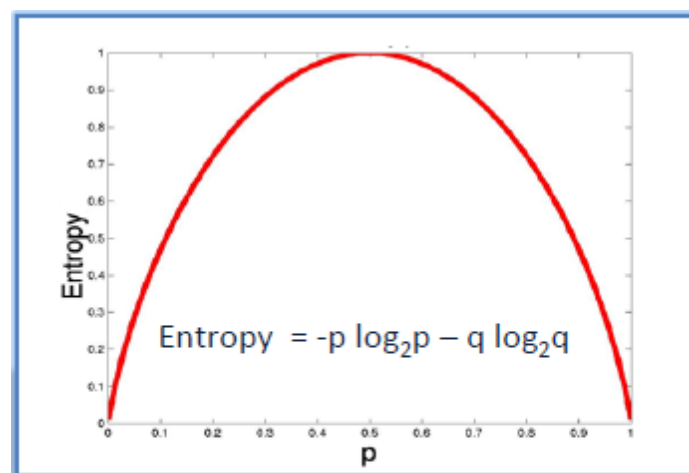
Decision tree constructs classification models in way amongst the tree framework. Each and every dataset is then split into many subsets while increasingly formulating an associated decision tree node by node in a horizontal manner. The final point is symbolized by a tree defining decision nodes and leaf nodes. The former contains at least 2 branches/children nodes. Leaf node defines a decision or classification. The root node is the uppermost decision node in a tree which is linked to the ideal analyst. Decision trees can deal with both numerical as well as categorical data.



Figure 6: Decision tree Set

Algorithm : The main algorithm for constructing decision trees is known as ID3 by J. R. Quinlan which utilizes a greedy which is a top down inquiry from the space of conceivable branches but it has not any backtracking options. Manufacturing of Decision Tree utilizes the Entropy and Information Gain.

Entropy: The decision tree is constructed top-down using a root node and includes dividing the information in the form of subsets which includes examples using alike standards (homogenous). ID3 calculation utilizes entropy to ascertain the uniformity of a specimen. In the event which includes the specimen being entirely uniform the entropy is given to be zero but in case the example is equally divided the entropy is 1.



$$\text{Entropy} = -0.5 \log_2 0.5 - 0.5 \log_2 0.5 = 1$$

Figure 7: Entropy Value

We use the two categories of entropy via frequency tables to build a decision tree as described below:

- a) Entropy utilizing recurrence data from a single attribute;

$$E(S) = \sum_{i=1}^c -p_i \log_2 p_i$$

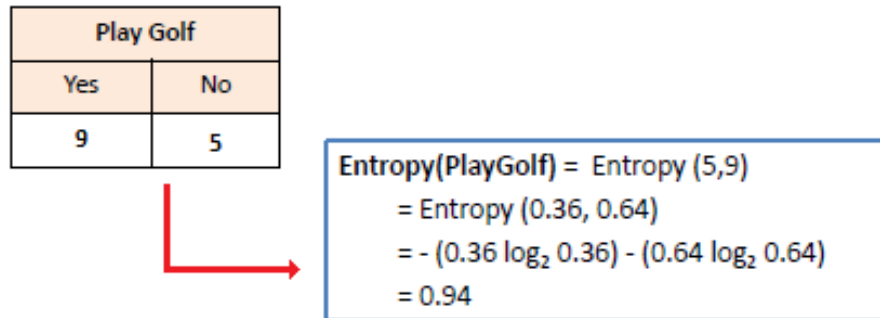


Figure 8: Entropy form Single attribute

- b) Entropy utilizing recurrence data from a mixture of two attributes:

$$E(T, X) = \sum_{c \in X} P(c)E(c)$$

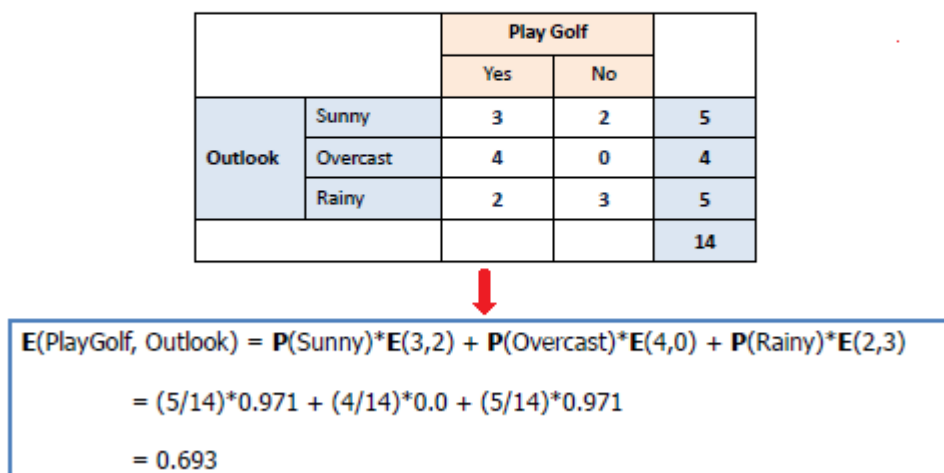


Figure 9: Entropy form Multiple attribute

Information Gain : It depends on the diminishing in entropy after a dataset is part of a property. Mostly finding attribute that gives the maximum information gain after effects for constructing a decision tree (i.e., the branches with the maximum uniformity).

Step 1: Find out entropy of the target

$$\begin{aligned}
 \text{Entropy(PlayGolf)} &= \text{Entropy}(5,9) \\
 &= \text{Entropy}(0.36, 0.64) \\
 &= -(0.36 \log_2 0.36) - (0.64 \log_2 0.64) \\
 &= 0.94
 \end{aligned}$$

- The splitting of the dataset is done on the basis of distinctive qualities. The entropy for each branch is figured. At that point it is included relatively, to get add up to entropy for the partition. Before the partition, subsequent entropy is deducted from the entropy. The outcome is the Information Gain, or reduction of entropy.

		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
		Gain = 0.247	

		Play Golf	
		Yes	No
Temp.	Hot	2	2
	Mild	4	2
	Cool	3	1
		Gain = 0.029	

		Play Golf	
		Yes	No
Humidity	High	3	4
	Normal	6	1
		Gain = 0.152	

		Play Golf	
		Yes	No
Windy	False	6	2
	True	3	3
		Gain = 0.048	

Table 11: Decision Tree set

$$Gain(T, X) = Entropy(T) - Entropy(T, X)$$

$$\begin{aligned}
 G(\text{PlayGolf, Outlook}) &= E(\text{PlayGolf}) - E(\text{PlayGolf, Outlook}) \\
 &= 0.940 - 0.693 = 0.247
 \end{aligned}$$

....(3)

- Trait possessing the highest information gain is taken as the decision node, distributing the dataset from all the outlets and thereby reiterating the identical method for each single division.

★		Play Golf	
		Yes	No
Outlook	Sunny	3	2
	Overcast	4	0
	Rainy	2	3
Gain = 0.247			

Figure 10: Decision Tree Data Set

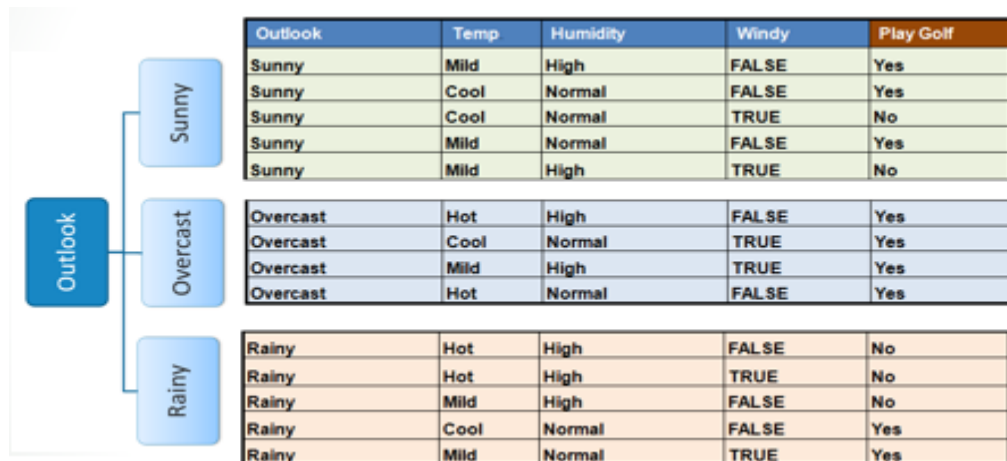


Figure 11: Decision Tree Data Set

- If the entropy of the branch is 0 then it is a leaf node.

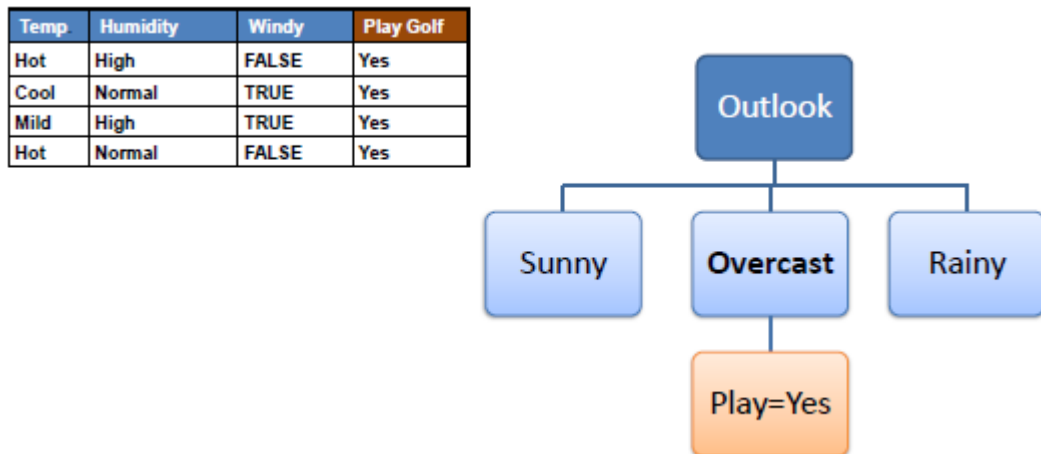


Figure 12: Decision Tree Data Set

- The ID3 calculation is recursively executed on middle branches till each and every information is grouped.

Decision Tree to Decision Rules : Mapping between root and leaf node helps in transforming the decision tree into the set of rules.

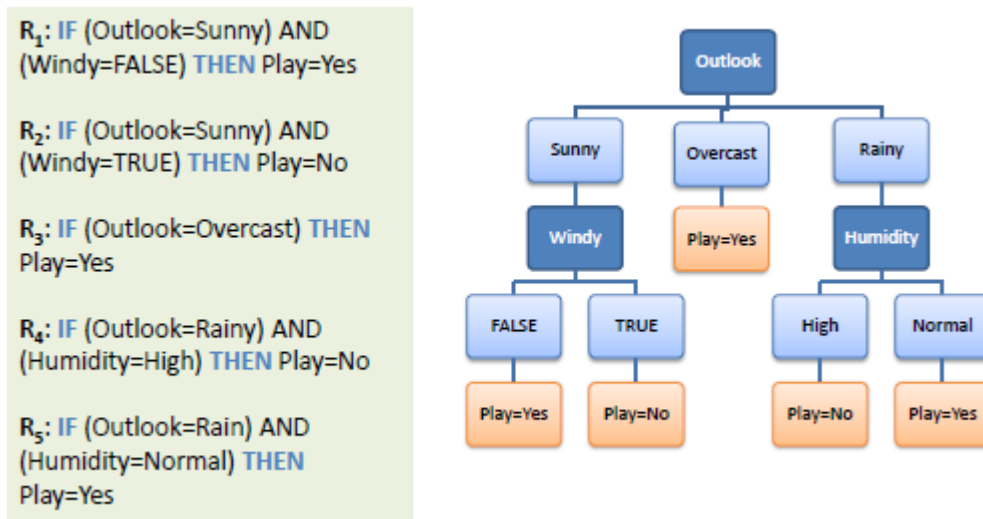


Figure 13: Mapping Decision tree to Decision Rules

Challenges faced from Decision tree

- Working with consistent characteristics (binning)
- Overfitting can't be avoided easily.
- There are many attributes with more than one value (Super attributes).
- Missing values pose a great issue.

3.4 Artificial Bee Colony

The ABC algorithm was first planned by Karagoga to look for best optimal results. The ABC calculation depends on the scrounging practices of bees and impersonates the communications that happen in swarm acquaintance to take care of enhancement issues. Swarm insight, similar to PSO, has been effectively connected in different fields, for example, remote sensor systems, flow shop issues, e-learning, clustering, and scheduling issues. In the genuine condition, a social group of bees in a state is made out of three fundamental sorts of bees, which are a ruler of the kernel, a couple bees for reproduction purpose, and an extensive number of specialists that search for pollen and deal with hatchlings in the province. The ABC algorithm depends on the behaviour of working bees, which it isolates into deployed, passer-by, and scout bees. Scout

bees are in charge of hunting down new food sources and announcing the measure of nectar at every area. After the scout bees have assembled the data, the deployed bees travel to the area of the nourishment sources to scan for the new source and discover how much nectar they have. The passerby bees sit tight for data about the nourishment sources from the deployed bees, and afterward utilize this to go out and assemble nectar.

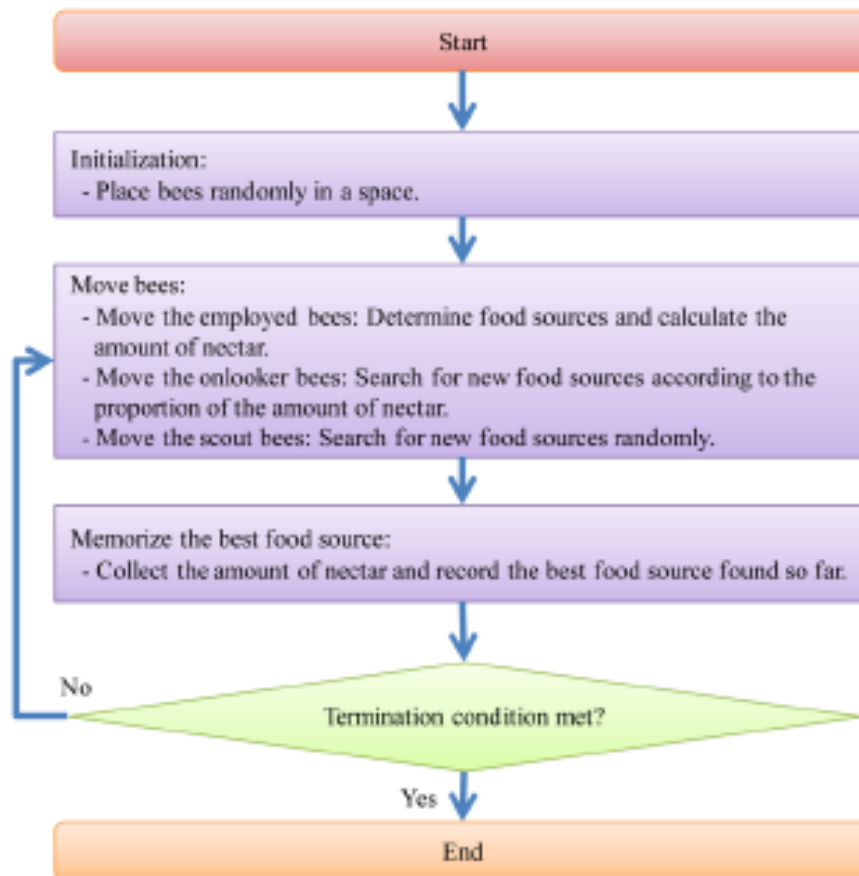


Figure 14: ABC workflow

Firstly, accept that bees are produced at the first step. Half of the N bees are chosen to arbitrarily spread out to look for the food source in the arrangement space. Every bee chooses a position and holds the measure of nectar there in its memory, with the aftereffects of this being utilized to create the primary f -value (fitness value). Next, every deployed bee flies to the chosen nourishment source and picks another position close to the first ones. In the wake of looking at the measure of nectar in its memory, the properties of this being utilized to create the main f -value. Next, each utilized honey bee flies to the chosen sustenance source and picks another position close to the first ones. In the wake of contrasting the measure of nectar at both

nourishment sources, the utilized honey bee chooses the one with the most nectar as the new sustenance source. Third, the onlooker bees remain at the hive to sit tight for data about the measure of nectar at the chose sustenance sources. The onlooker bees at that point select a sustenance source and the likelihood that a specific nourishment source will be chosen increments alongside the measure of nectar that it has.

Probability of food source is calculated as:

$$P_i = \frac{F(\theta_i)}{\sum_{k=1}^E F(\theta_k)} \quad \dots(4)$$

Here θ_i is location of food source i , $F(\theta_i)$, The amount of nectar at their i th source is denoted by $F(\theta_i)$ similarly for $F(\theta_k)$, E represents the no of bees deployed. In the wake of moving to the chose nourishment source, every onlooker bee chooses a position close to the first sustenance source utilizing the accompanying condition, and acquires the measure of nectar:

$$x_{id}(t+1) = \theta_{id} + \phi(\theta_{id}(t) - \theta_{kd}(t)) \quad \dots(5)$$

Here t denotes the number of turns, i is no of bees deployed, d is the dimension of the position, k denotes the choice of randomly chosen bee deployed at an area, θ_{id} means the of the i th onlooker bee, θ_{kd} denotes position of the randomly chosen bee deployed, and $\phi()$ is randomized to $[-1, 1]$, where $1 \leq d \leq D$, $1 \leq i \leq E$ and $1 \leq k \leq E$. Fourth, scout bees are utilized to arbitrarily scan for new sustenance sources. In the ABC calculation, every nourishment source chosen by the utilized honey bee has a parameter to record the quantity of the sustenance source chosen. Once the parameter esteem surpasses the foreordained number of iterations, known as the threshold, the sustenance source is relinquished. The deployed bee turns into a scout bee and looks for another nourishment source to supplant the first one. The operation of the scout bee is as per the following:

$$\theta_{id} = \theta_{id_{\min}} + \eta \times (\theta_{id_{\max}} - \theta_{id_{\min}}) \dots(6)$$

where η is a random value in the range [0, 1]. The scout bee at that point turns into the deployed bee once more. At long last, after the bees finish an inquiry procedure, the highest fitness value is gained through a correlation of all the wellness esteems. Toward the finish of each round, the calculation checks whether the bees should keep on searching for new sustenance sources or not. In the event that the end condition is fulfilled, the calculation stops the inquiry procedure and afterward yields the related aftereffects of the fitness value and the nourishment sources.

$$x_{id}(t + 1) = \theta_{id} + \phi(\theta_{id}(t) - \theta_{kd}(t)) \dots(7)$$

The deployed bee whose nourishment source has been deserted by the bees turns into a scout. The fundamental strides of the calculation can be portrayed as takes after.

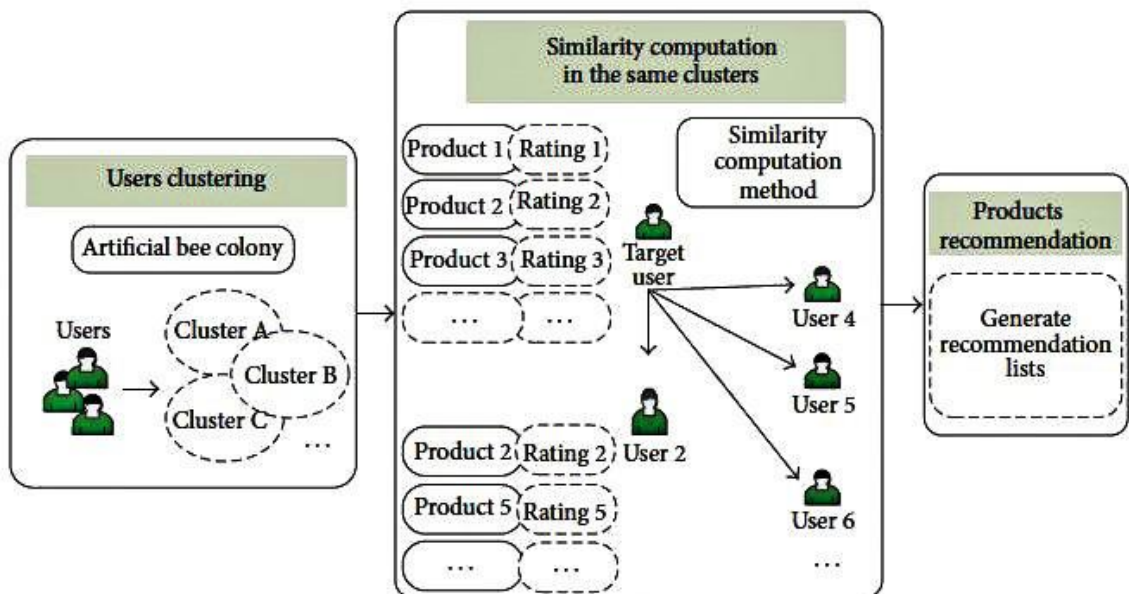


Figure 15: Recommendations based on ABC

CHAPTER 4

IMPLEMENTATION

In this chapter we will discuss the experimental setup of the research work done. First section will discuss the data set followed by tools used for programming. In the next section, evaluation procedure is discussed. In the last section summary of the chapter is given.

4.1. Data Set Collection

The data set collected here in this thesis is the dataset of rare elements picked from the vast variety of tv series, out of which anime series dataset was chosen. So as to cater to longtail issue in the recommender system. Also, we utilized the numerically represented dataset so as to ease the usage of information utilized in the dataset.

Data set for animated Series:

Name	Collection	Episodes	Genre	Rating
Durarara!!	Album	24	Act/Adv	8.38
ElfenLied	Single	13	Act/Adv	7.86
FairyTail	Album	175	Act/Adv	8.23
SoulEater	Single	51	Act/Adv	8.08
Psycho-Pass	Single	22	Act/Adv	8.51
KilllaKill	Single	24	Act/Adv	8.23
AkamegaKill!	Single	24	Act/Adv	7.84
GuiltyCrown	Single	22	Act/Adv	7.82
Kuroshitsuji	Album	24	Act/Adv	8.06
Baccano!	Single	13	Act/Adv	8.54
K	Album	13	Act/Adv	9.04
Berserk	Single	25	Act/Adv	8.40
FairyTail	Album	175	act/adv/com	8.98
SoulEater	Album	51	act/adv/com	8.08
TengenToppaGurrenLagann	Single	27	act/adv/com	8.78
OnePunchMan	Single	13	act/adv/com	8.83
CowboyBebop	Single	26	act/adv/com	8.83
SamuraiChamploo	Single	26	act/adv/com	8.50
D.Gray-man	Album	103	act/adv/com	7.91
InuYasha	Album	167	act/adv/com	7.89
FullMetalPanic!	Single	24	act/adv/com	7.81
TokyoGhoul	Album	12	horror	8.90
Another	Single	12	horror	7.89
DeadmanWonderland	Single	12	horror	7.49
Hellsing	Album	13	horror	7.64
Gantz	Single	13	horror	7.27
Blood-C	Single	12	horror	6.73
PerfectBlue	Single	1	horror	8.23
Mayoiga	Single	12	horror	5.81

HatarakuMaou-sama!, Single, 13, romcom, 8.04
HighSchoolDxD, Single, 12, romcom, 7.70
KaichouwaMaid-sama!, Single, 26, romcom, 8.26
SakurasounoPetnaKanojo, Single, 24, romcom, 8.40
YahariOrenoSeishunLoveComedywaMachigatteiru, Album, 13, romcom, 8.12
Nisekoi, Single, 20, romcom, 7.91
BakatoTesttoShoukanjuu, Album, 13, romcom, 8.45
NHKniYoukoso!, Single, 24, romcom, 8.41
GekkanShoujoNozaki-kun, Single, 12, romcom, 8.24
KaminomizoShiruSekai, Single, 12, romcom, 7.95
KokoroConnect, Single, 13, romcom, 8.01
GoldenTime, Single, 24, romcom, 7.91
Bakuman, Single, 25, romcom, 8.35
HighschooloftheDead, Single, 12, ecchi, 7.47
HighSchoolDxD, Single, 12, ecchi, 7.70
Zeronotsukaima, Single, 13, ecchi, 7.62
ShokugekinoSouma, Album, 24, ecchi, 8.62
NanatsunoTaizai, Album, 24, ecchi, 8.43
Chobits, Single, 26, ecchi, 7.57
ToLOVE-Ru, Album, 26, ecchi, 7.34
StriketheBlood, Album, 24, ecchi, 7.44
Sankarea, Single, 12, ecchi, 7.53
PrisonSchool, Single, 12, ecchi, 8.04
Sekirei, Single, 12, ecchi, 7.40
GoldenBoy, Single, 6, ecchi, 8.05

4.2. Algorithm

Algorithm proposed in chapter 3 was used. Each of the four techniques were first implemented separately, then their output is given as the input to the algorithm to extract more effective keywords, for determining the central idea of the document. The algorithms used are Decision tree, Artificial Bee Colony, Pearson Correlation and k- means algorithm.

4.3. Environment & Experimental Setting

For these algorithms anime dataset corpus used in "kaggle" is explored and utilized. To legitimately assess our calculation, and contrast the existing calculations we have endeavored tests on 150 records approximately of this dataset. In this assignment, all synopsis frameworks give a short synopsis for each of the 200 anime series. Every summary is naturally assessed against 4 display outlines mined from "kaggle". These numerical values are taken from "kaggle" which is in turn compatible with the final data used for analysis. In the accompanying analyses for independent event based synopsis, we explore the viability of the approach. Furthermore, we endeavour to test the significance of logical data in scoring event terms. The number and kind of neighbouring renamed substances are considered to set the weights of

occasion terms. Synopsis assessment is a hard errand; execution of a calculation can change significantly in various synopsis settings. It is unrealistic to state that solitary technique is best for each corpus. There are frameworks that fuse distinctive elements also, provide solitary calculation and choose which one to utilize depending upon the corpus with machine learning calculations.

4.4. Programming Tools Used

SOFTWARE REQUIREMENTS

Programming Language :	Python 2.7 .
Database :	anime custom dataset
Tool :	manual

HARDWARE REQUIREMENTS

Processor :	1GHz or over
RAM :	1GB or more
HDD :	80GB
OS :	Windows 8 or above

CHAPTER 5

RESULTS & ANALYSIS

In Chapter 5, we discuss and analyse the result and related studies to fulfil our research objectives mentioned in chapter 1.

5.1 Results

Recall and precision is calculated to get the value for the f-score and the value of the accuracy. which inturn will help us in comparing and contrasting the four algorithms F-score is a composite measure of precision and recall. F-score (iii) is nothing but harmonic average of precision (ii) and recall (i):

$$Recall = \frac{RelevantSentences \cap RetrievedSentences}{Relevant Sentences} \quad (i)$$

$$Precision = \frac{RelevantSentences \cap RetrievedSentences}{Retrieved Sentences} \quad (ii)$$

$$F - score = \frac{2 * Precision * Recall}{Precision + Recall} \quad (iii)$$

Below is a more complex formula for measuring the F-score:

$$F - score = \frac{(\beta^2 + 1) \cdot P \cdot R}{\beta^2 \cdot P + R} \quad (iv)$$

where β factor that favors precision when $\beta > 1$ and favors recall when $\beta < 1$. For n distinct tasks, the precision for a category is the number of true positives (i.e. No. of items rightly abled

as recommended)divided by the total no of items categorized as recommended (rightfully and wrongly) Recall in the same way is defined as the no of true positives upon the total no items categorized as not recommended(rightfully and wrongly)

$$\text{Precision} = \frac{tp}{tp + fp} \quad (v)$$

$$\text{Recall} = \frac{tp}{tp + fn} \quad (vi)$$

$$\text{Accuracy} = \frac{tp + tn}{tp + tn + fp + fn} \quad (vii)$$

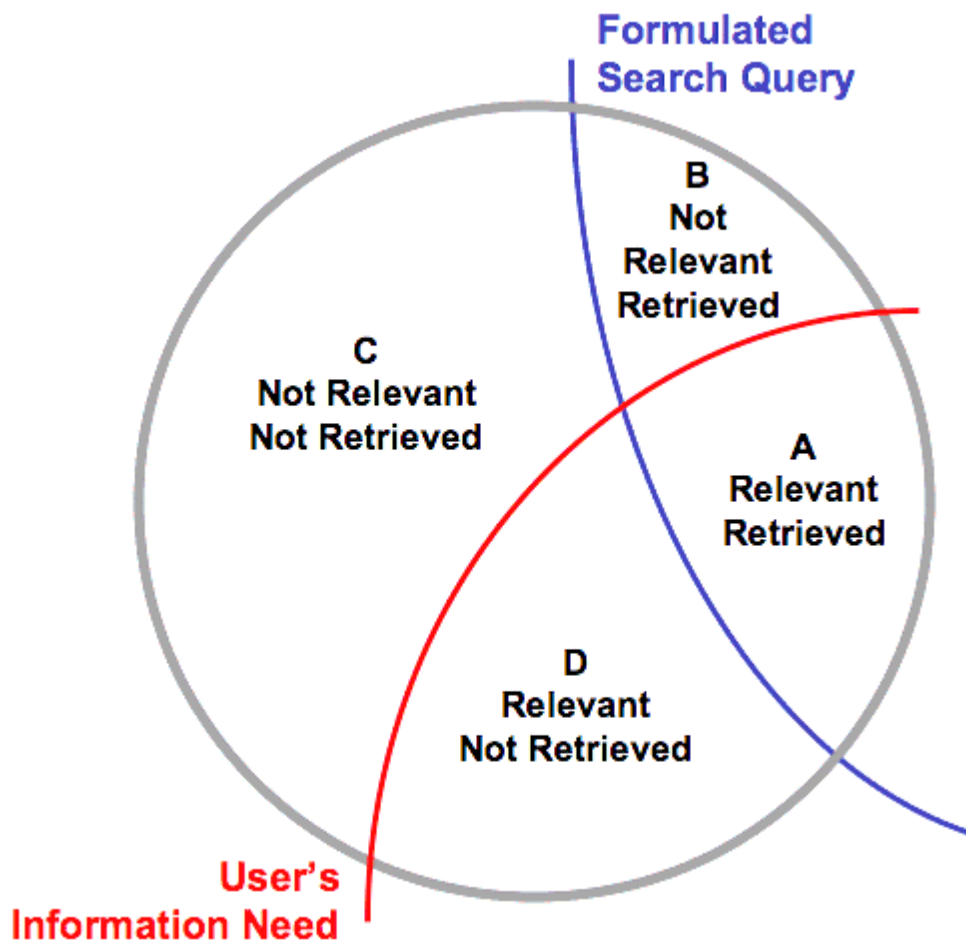


Figure 16: Relevance Ven Diagram

Since the computation of these measures is troublesome and tedious, additionally the work is completely in light of machine learning algorithms, thus mechanized programming method or, on the other hand calculation is utilized for this evaluation.

SET 1	k-mean	Pearson	ABC	decision tree
Precision	70.58	58.16	73.85	64.25
Recall	96.64	97.15	98.88	94.51
f-score	81.27	83.82	84.54	76.49
Accuracy	69.56	77.52	86.41	64.14

Table 12: SET 1

SET 2	k-mean	Pearson	ABC	decision tree
Precision	68	54.87	61.98	53.12
Recall	98.55	97.12	96.27	92.37
f-score	80.47	72.29	75.4	67.44
Accuracy	67.32	83.82	85.36	52.45

Table 13: SET 2

SET 3	k-mean	Pearson	ABC	decision tree
Precision	66.66	54.58	72.83	54.18
Recall	99.31	94.55	100	97.33
f-score	79.79	69.13	84.27	67.33
Accuracy	65.45	65.57	84.47	50.76

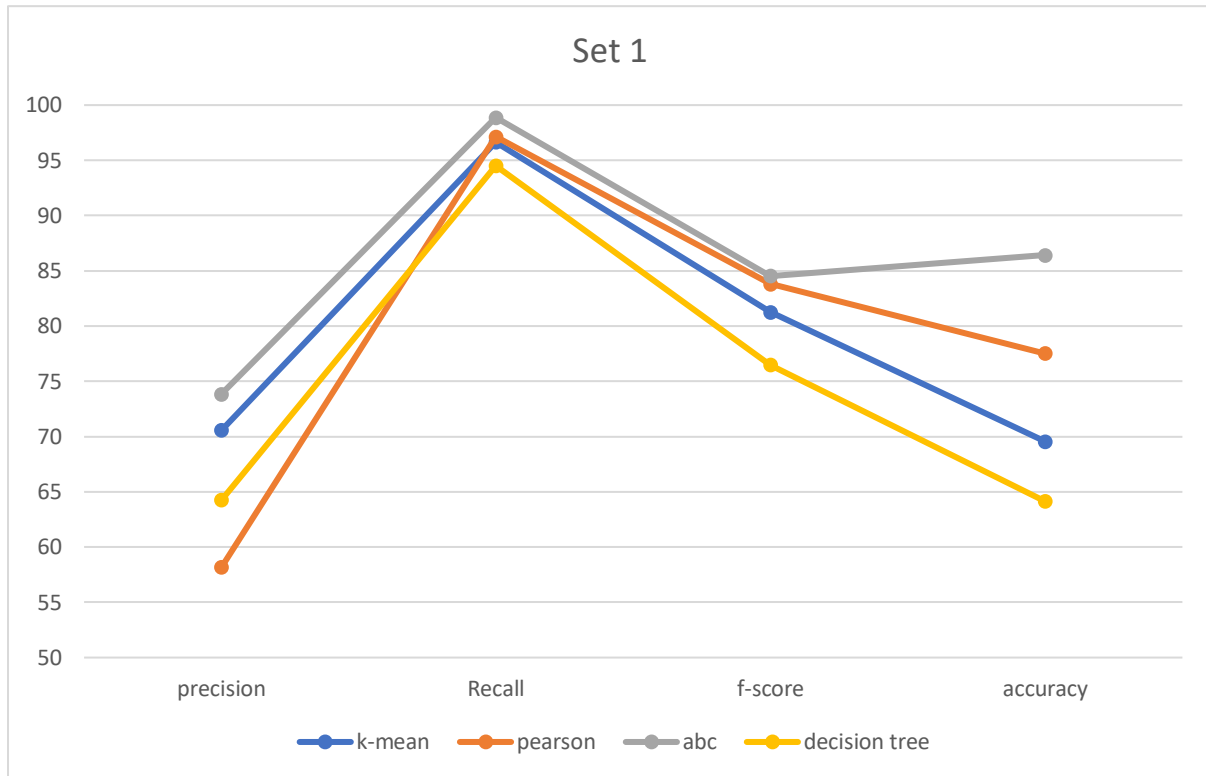
Table 14: SET 3

SET 4	k-mean	Pearson	ABC	decision tree
Precision	45.12	52	70.56	41.02
Recall	100	98.58	100	94.52
f-score	81.27	67.21	82.73	57.2
Accuracy	42.5	63.1	85.78	40.49

Table 15: SET 4

5.2 Analysis

In this section all four algorithms have been compared and analysed based on the following factors: Precision, Recall, f-score and accuracy. Thus helping us to determine the effectiveness of the algorithms and their impact on Recommender System.



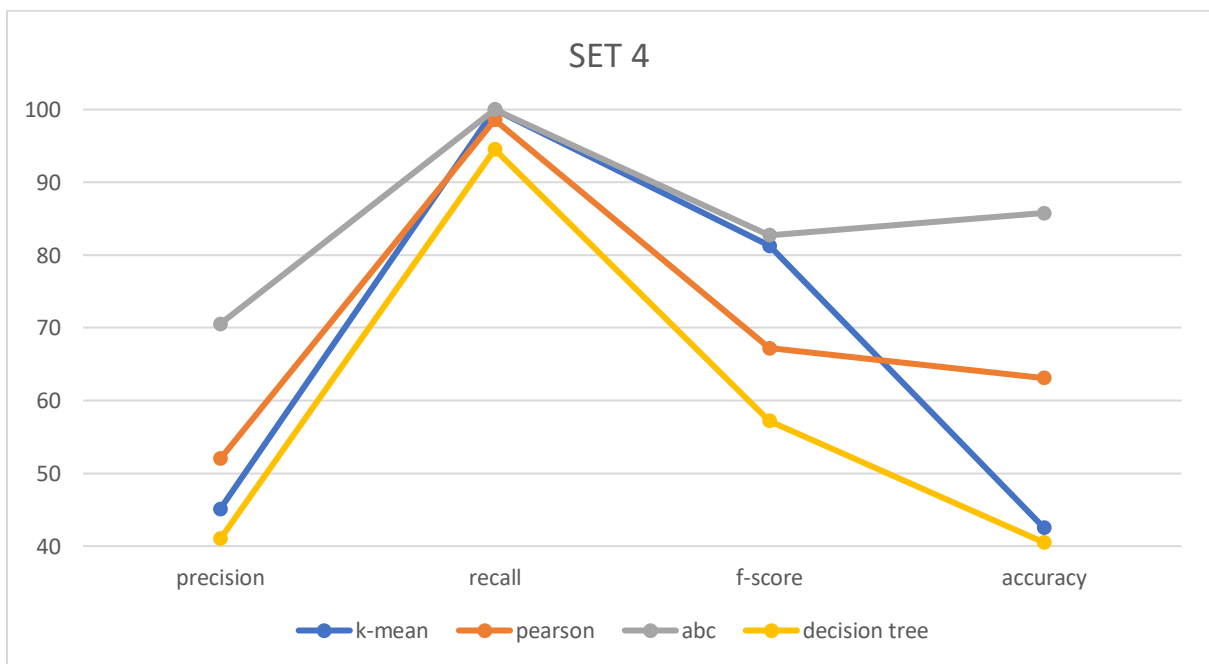
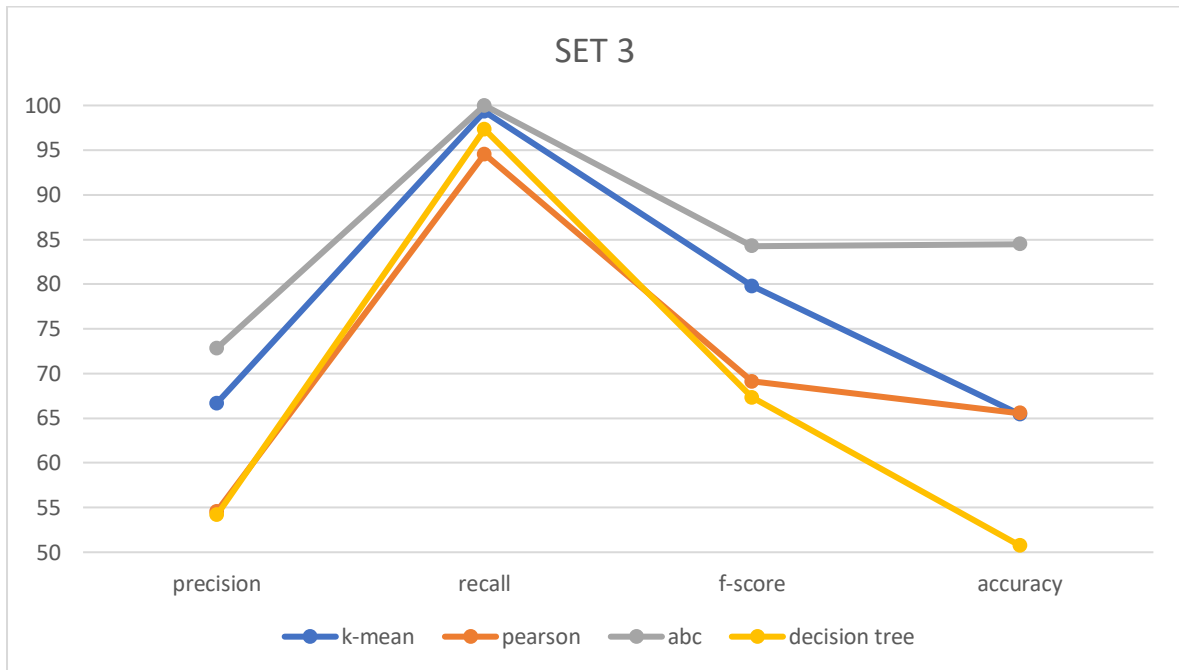


Figure 17: Comparative Analysis between different Algorithms

5.3 Mapping

In this section we have done mapping between Different types of Recommender System's and the Challenges faced while dealing with them. Which gives user, the complete preview of all the challenges he/she might face while dealing with a particular type of Recommender System

Types of Recommender Systems						
Challenges	Collaborative RS	Content based RS	Demographic RS	Hybrid RS	Domain Based	Knowledge based
Cold start problem	?	✓	✓	✓	?	✓
Scalability of the approach.	✓	✓	✓	?	X	X
Big-data	✓	✓	✓	?	X	X
Privacy concerns.	✓	X	✓	✓	✓	✓
Sparsity	X	✓	X	✓	?	?
Recommending the items in the Long tail	?	✓	✓	✓	?	✓
Accuracy of the Suggestions	?	✓	X	?	X	X
Changing data set	✓	✓	X	✓	✓	?
Impact of context-awareness	?	✓	?	✓	✓	?

Table 16: Mapping between Recommendation approaches and there corresponding challenges

CHAPTER 6

CONCLUSION

This chapter concludes the contributions made by this thesis. Also figure out the limitation of the work done and briefly discuss the future scope of the research.

6.1 Research Summary

This thesis investigated the diversity in Recommender Systems discussing the scope and practical use of each algorithm. The existent challenges within the research domain of recommender system are ascertained from pertinent literature and finally, a mapping of the open problems in the type of recommendation system is given. The idea was to probe issues that were valid for the specific type of RS and which spanned across different types. The findings clearly suggest the challenges as opportunities within the research area. Also, the detailed survey of the long tail issue and all the articles related to the long tails have been elaborated and duly stated. Thus giving the clear overview to the newbie on how to cater and work on the long tail issue in the recommender system. Also the detailed survey of the four major algorithms and thereby providing the overview on how to fabricate the algorithm. According to the statistical categorization of all four blogs.

Hereby we can state the order of accuracy and preference of the four algorithms:

Artificial Bee Colony

Pearson Correlation

K-Means

Decision tree

6.2 Future Scope

In future, I'll try and examine more algorithms and compare them on the basis of the main criteria's so as to get the better understanding of the diversity in a recommender system. Also helping the newbies to pick the best algorithm that suits the need. Also, I'll try and focus on the different issue in recommender system thus inturn helping to cater new issues and thus providing the better understanding of the environment. In future I will be working on the hybrid approach so as to get the accuracy of ABC algorithm and the fast execution of Decision tree, encapsulating the best features of the algorithms, enabling us to make a better Recommender system.

REFERENCES

1. Singh G., Boparai R.S., "A survey on recommendation system", (January 20, 2017) IOSR Journal of Computer Engineering (IOSR-JCE), e-ISSN: 2278-0661,p-ISSN: 2278-8727, Volume 17, Issue 6, Ver. V (Nov – Dec. 2015), PP 46-51, : 10.9790/0661-17654651.
2. Lakshmi S S., Dr. Lakshmi T.A., "Recommendation Systems:Issues and challenges"(January 2015), (IJCSIT) International Journal of Computer Science and Information Technologies, Vol. 5 (4) , 2014, 5771-5772,ISSN :0975-9646,11.
3. Prasad R. and Kumari V V., "A CATEGORICAL REVIEW OF RECOMMENDER SYSTEMS"(5 September,2015), International Journal of Distributed and Parallel Systems (IJDPS) Vol.3, No.5, September 2012: 10.5121/ijdps.2012.3507.
4. Shinde S., Mrs. Potey M. A., "Survey on Evaluation of Recommender Systems"(2 February 2015), International Journal Of Engineering And Computer Science ISSN:2319-7242 Volume 4 Issue 2 February 2015, Page No. 10351-10355, e-ISSN: 2319-7242.
5. Jannach D., Zanker M., Felfernig A., G.Friedrich," Recommender Systems – An Introduction"(21 ,januaray 2014), <https://pdfs.semanticscholar.org/5d1d/d378962c7601526f65f69e408f8800a0d3c4.pdf>, .
6. Macmanus R.,"5 problems on Recommender systems ,a web article" (January 28, 2009), http://readwrite.com/2009/01/28/5_problems_of_recommender_systems/ ,

7. Zhao Z.D., Shang M.S., "User-Based Collaborative-Filtering Recommendation Algorithms on Hadoop" (10 Jan. 2010), Scholar of Computer Science & Eng, University of Electron Science & Technology of China.
8. Sarwar B., Karypis G., Konstan J., and Riedl J., "Item- Based Collaborative Filtering Recommendation Algorithms" (May 1-5, 2001), GroupLens Research Group/Army HPC Research Center Department of Computer Science and Engineering University of Minnesota, Minneapolis, ACM 1-58113-348-0/01/0005.
9. Lops P., Gemmis M.D., Semeraro G., " Content-based Recommender Systems: State of the Art and Trends (05 October 2010), pp 73-105, 10.1007/978-0-387-85820-3_3.
10. Yin H., Sun Y., Cui B., Hu Z., Chen L., "a location- content-aware recommender system" (August 14, 2013), KDD '13 Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining Pages 221-229, 10.1145/2487575.2487608 .
11. Noia T.D., Mirizzi R., Ostuni V.C., Romito D., Zanker M., "Linked open data to support content-based recommender systems" (September 06, 2012), I-SEMANTICS '12 Proceedings of the 8th International Conference on Semantic Systems, Pages 1-8, 10.1145/2362499.2362501.
12. Cantador I., Bellogín A., Vallet D., "Content-based recommendation in social tagging systems" (September 30, 2010), RecSys '10 Proceedings of the fourth ACM conference on Recommender systems, Pages 237 -240, 10.1145/1864708.1864756.
13. Beel J., Langer S., Nürnberger A., Genzmehr M., " The Impact of Demographics (Age and Gender) and Other User- Characteristics on Evaluating Recommender Systems" (September 26, 2013), Research and Advanced Technology for Digital Libraries Volume 8092 of the series Lecture Notes in Computer Science pp 396-400, 10.1007/978-3- 642-40501-3_45.

14. Ngai G., Chan S.C., Wang Y., "Applicability of Demographic Recommender System to Tourist Attractions: A Case Study on Trip Advisor" (December 07, 2012), WI-IAT '12 Proceedings of the The 2012 IEEE/WIC/ACM International Joint Conferences on Web Intelligence and Intelligent Agent Technology - Volume 03, Pages 97-101 ,10.1109/WI-IAT.2012.133.
15. Ghazanfar M.A., Bennett A.P., " A Scalable, Accurate Hybrid Recommender System"(10 Jan. 2010), 2010 Third International Conference on Knowledge Discovery and Data Mining, Phuket, 2010, pp. 94-98.: 10.1109/WKDD.2010.117.
16. Campos L.M.D., Fernández-Luna J.M., Huete J.F., Rueda-Morales M.A., " Combining content-based and collaborative recommendations: A hybrid approach based on Bayesian networks "(11 April 2010), International Journal of Approximate Reasoning Volume 51, Issue 7, September 2010, Pages 785-799,;10.1016/j.ijar.2010.04.001.
17. Porcel, Viedma P.E., " Dealing with incomplete information in a fuzzy linguistic recommender system to disseminate information in university digital libraries"(5 August 2009), Published on: Knowledge-Based Systems Volume 23, Issue 1, February 2010, Pages 32–39 Special Issue on Intelligent Decision Support and Warning Systems, 10.1016/j.knosys.2009.07.007.
18. Carrer-Neto W., Hernández-Alcaraz M.L., Valencia- García R., García-Sánchez F., " Social knowledge-based recommender system. Application to the movies domain"(10 March 2012), Expert Systems with Applications, Volume 39, Issue 12, 15 September 2012, Pages 10990–11000.
19. Adomavicius G., Tuzhilin A., " Context-Aware Recommender Systems"(05 October 2010), pp 217-253, 10.1007/978-0-387-85820-3_7.

20. Lika B., Kolomvatsos K., Hadjiefthymiades S.," Facing the cold start problem in recommender systems"(16 September 2013), Expert Systems with Applications Volume 41, Issue 4, Part 2, March 2014, Pages 2065–2073, 10.1016/j.eswa.2013.09.005.
21. Bobadilla J., Ortega F., Hernando A., Bernal J.," A collaborative filtering approach to mitigate the new user cold start problem"(30 August 2011), Knowledge-Based Systems Volume 26, February 2012, Pages 225–238, 10.1016/j.knosys.2011.07.021.
22. Dhillon, Hsieh S.S.C.C., Hsiang F.Y.," Scalable Coordinate Descent Approaches to Parallel Matrix Factorization for Recommender Systems"(13 Dec. 2012), Data Mining (ICDM), 2012 IEEE 12th International Conference on, 10.1109/ICDM.2012.168.
23. Pearl P., Chen L., Rong H.," Evaluating recommender systems from the user's perspective: survey of the state of the art"(10 March 2012), User Modeling and User-Adapted Interaction, The Journal of Personalization Research, Journal no. 11257, : 10.1007/s11257-011-9115-7.
24. Adomavicius G., Young O k.," Improving Aggregate Recommendation Diversity Using Ranking-Based Techniques"(06 January 2011), IEEE Transactions on Knowledge and Data Engineering (Volume: 24, Issue: 5, May 2012),: 10.1109/TKDE.2011.15.
25. Katrien V., Manouselis N., Ochoa X.," Context-Aware Recommender Systems for Learning: A Survey and Future Challenges"(24 April 2012), IEEE Transactions on Learning Technologies (Volume: 5, Issue: 4, Oct.-Dec. 2012),: 10.1109/TLT.2012.11.
26. Zhou X., Yue X., Yuefeng L., Audun J., Clive Cox," The state- of-the-art in personalized recommender systems for social networking"(12 May 2011), Artificial Intelligence Review February 2012, Volume 37, Issue 2, pp 119–132, : 10.1007/s10462-011-9222-1.

27. Georgios P., Svein J. K., " Social Trust as a solution to address sparsity-inherent problems of Recommender systems"(5 Aug 2012), ACM RecSys 2009, Workshop on Recommender Systems & The Social Web, Oct. 2009, ISSN:1613-0073, New York, USA, Cite as:arXiv:1208.1004 [cs.SI].
28. Athanasios N. Nikolakopoulos, Marianna A. Kouneli, John D. Garofalakis, " sparsity in ranking-based recommendation"(24 February 2015), : 10.1016/j.neucom.2014.09.082.
29. Eran T.,Yang W.,Lorrie F.C.," Personalization and privacy: a survey of privacy risks and remedies in personalization-based systems"(10 March 2012), Cited as: Toch, E., Wang, Y. & Cranor, L.F. User Model User- Adap Inter (2012) 22: 203,: 10.1007/s11257-011-9110-z.
30. Hunt N., Carlos A. Gomez-U.,"The Netflix Recommender System: Algorithms, Business Value, and Innovation"(4, January 2016), journal, ACM Transactions on Management Information Systems (TMIS), Volume 6 Issue 4, January 2016, Article No. 13, 10.1145/2843948 .
31. Kumar, A. & Sharma, A. (2012). Alleviating Sparsity & Scalability Issues in Collaborative filtering based Recommender System .Proceedings of International Conference on Frontiers of Intelligent Computing: Theory and applications (FICTA), Springer AISC, pp. 103-112

CODE SNIPPETS

K-MEAN

```
import math
def euclideanDistance(instance1, instance2, length):
    distance = 0
    for x in range(length):
        distance += pow((instance1[x] - instance2[x]), 2)
    return math.sqrt(distance)
```

```
import numpy as np
import pandas as pd
UserData = pd.read_csv(r'Anime.data')
UserDataArray = np.array(UserData)
array = []
c1 = []
c2 = []
center1=[4.6,3.1,1.5,0.2]
center2=[5.0,3.6,1.4,0.2]
#center1 = np.random.randint(0,UserDataArray.shape[0],1)
#center2 = np.random.randint(0,UserDataArray.shape[0],1)
print(center1)
print(center2)
for row in range(len(UserDataArray)):
    array = UserDataArray[row]
    distance1 = euclideanDistance(center1, array, 4)
    distance2 = euclideanDistance(center2, array, 4)
    if distance1 < distance2 :
        c1.append(array)
    else:
        c2.append(array)
print(c1)
print('\n')
print(c2)
print('\n')
for k in range(5):
    a1=np.array(c1)
    a1=a1.mean(axis=0)
    print(a1)
    a2=np.array(c2)
    a2=a2.mean(axis=0)
    print(a2)
    print('\n')
min=1000.0
```

```

l=len(c1)
newcenter1=[]
for row in range (l):
    array = c1[row]
    temp=euclideanDistance(a1, array, 4 )
    if(temp<min):
        min=temp
        newcenter1=array
min=1000.0
l=len(c2)
newcenter2=[]
for row in range (l):
    array = c2[row]
    temp=euclideanDistance(a2, array, 4 )
    if(temp<min):
        min=temp
        newcenter2=array
if (newcenter1==c1).all() and (newcenter2==c2).all():
    break
else:
    center1 = newcenter1
    center2= newcenter2
    for row in range(len(UserDataArray)):
        array = UserDataArray[row]
        distance1 = euclideanDistance(center1, array, 4)
        distance2 = euclideanDistance(center2, array, 4)
        if distance1 < distance2 :
            c1.append(array)
        else:
            c2.append(array)
print(center1)
print(center2)
print('\n')
testArray = [7.3,2.9,6.3,1.8]
center1
center2

```

ABC

```
import numpy as np
```

```

import random
import operator

class Centroid():
    def __init__(self, cl, acc):
        self.cl = cl
        self.acc = acc
        self.count = 1

    def append(self, data):
        for i, val in enumerate(self.acc):
            self.acc[i] += data[i]
            self.count += 1

    def getCentroid(self):
        return self.acc / self.count
#Reads and normalize the database, returns the data and classes apart
def readDatabase(filename, has_id, class_position):

    with open(filename) as f:
        # Getting only the lines without missing attribute
        lines = (line for line in f if '?' not in line)
        dataset = np.loadtxt(lines, delimiter = ',')

    # Shuffling the dataset, once sometimes the data are grouped by class
    np.random.shuffle(dataset)

    # Considering the last column being the class column
    if class_position == 'first':
        classes = dataset[:, 0]
        dataset = np.delete(dataset, 0, axis = 1)
    else:
        classes = dataset[:, -1]
        dataset = np.delete(dataset, -1, axis = 1)

    if has_id:
        # Remove the first column (ID)
        dataset = np.delete(dataset, 0, axis = 1)

    # Normalizing the data in the [0 1] interval
    arr_max = np.max(dataset, axis = 0) # gets the max of each column
    arr_min = np.min(dataset, axis = 0) # gets the min of each column

    rows, cols = np.shape(dataset)
    for i in range(rows):
        for j in range(cols):
            dataset[i][j] = (dataset[i][j] - arr_min[j]) / (arr_max[j] - arr_min[j])

    return dataset, classes
# Determine the classes centroids as the mean values of the data

```

```

# in each class
def determineCentroids(dataset, classes):
    rows, cols = np.shape(dataset)

    stats = {}

    for i, row in enumerate(dataset):
        class_id = str(classes[i])
        if class_id in stats:
            stats[class_id].append(row)
        else:
            stats[class_id] = Centroid(classes[i], row)

    centroids = {}
    for key in stats:
        centroids[key] = stats[key].getCentroid()

    return stats, centroids
# Simple Euclidian distance between two arrays
def euclidianDistance(a, b):
    diff_sqrt = [(x - y)**2 for x, y in zip(a, b)]

    return np.sqrt(np.sum(diff_sqrt))
# The sum of the distances between a data point and its class centroid
# in the training set
def costFunction(dataset, classes, cl, centroid):
    # 'cl' will be the string representation of the class already
    distances_sum = 0
    count = 0
    for i, d in enumerate(dataset):
        if str(classes[i]) == cl: # limiting the search only in the specific class
            distances_sum += euclidianDistance(d, centroid[cl])
            count += 1

    return distances_sum / count
def fitnessFunction(costs):
    fitness = costs.copy()
    for key in fitness:
        fitness[key] = 1/(1 + costs[key])

    return fitness
def rouletteWheelFunction(P):
    p_sorted_asc = sorted(P.items(), key = operator.itemgetter(1))
    p_sorted_desc = dict(reversed(p_sorted_asc))

    pick = np.random.uniform(0, 1)
    current = 0
    for key in p_sorted_desc:
        current += p_sorted_desc[key]
        if current > pick:

```

```

    return key
def ABC(dataset, classes, centroids, a_limit, max_iter):
    n_data, n_attr = np.shape(dataset) # Number of cases and number of attributes in each case
    n_bees = len(centroids) # Number of bees in the problem
    var_min = 0 # Minimum possible for each variable
    var_max = 1 # Maximum possible for each variable

    keys = [key for key in centroids] # centroid keys

    # Initialize the counter of rejections array
    C = centroids.copy()
    for key in C:
        C[key] = 0

    # Initilize the cost array
    costs = centroids.copy()
    for cl in costs:
        costs[cl] = costFunction(dataset, classes, cl, centroids[cl])

    best_solution = 999999999
    best_solutions = np.zeros(max_iter)

    for it in range(max_iter):
        # Employed bees phase
        for cl in centroids:
            _keys = keys.copy() # copying to maintain the original dict
            index = _keys.index(cl)
            del _keys[index]
            k = random.choice(_keys) # getting a index k different from i

            # Define phi coefficient to generate a new solution
            phi = np.random.uniform(-1, 1, n_attr)

            # Generating new solution
            # centroids: numpy array
            # phi: numpy array
            # (centroids[cl] - centroids[k]): numpy array
            # The operation will be element by element given that all the operands
            # are numpy arrays
            # TODO: ceil and floor of the new solution
            new_solution = centroids[cl] + phi * (centroids[cl] - centroids[k])

            # Calculate the cost of the dataset with the new centroid
            new_solution_cost = costFunction(dataset, classes, cl, new_solution)

            # Greedy selection: comparing the new solution to the old one
            if new_solution_cost <= costs[cl]:
                centroids[cl] = new_solution
                costs[cl] = new_solution_cost
                C[cl] = 0

```

```

else:
    # Increment the counter for discarded new solutions
    C[cl] += 1

F = fitnessFunction(costs) # calculate fitness of each class
f_sum_arr = [F[key] for key in F]
f_sum = np.sum(f_sum_arr)
P = {} # probabilities of each class
for key in F:
    P[key] = F[key]/f_sum

# Onlooker bees phase
for cl_o in centroids:
    selected_key = rouletteWheelFunction(P)

    _keys = keys.copy() # copying to maintain the original dict
    index = _keys.index(selected_key)
    del _keys[index]
    k = random.choice(_keys) # getting a index k different from i

    # Define phi coefficient to generate a new solution
    phi = np.random.uniform(-1, 1, n_attr)

    # Generating new solution
    # centroids: numpy array
    # phi: numpy array
    # (centroids[selected_key] - centroids[k]): numpy array
    # The operation will be element by element given that all the operands
    # are numpy arrays
    # TODO: ceil and floor of the new solution
    new_solution = centroids[selected_key] + phi * (centroids[selected_key] -
centroids[k])

    # Calculate the cost of the dataset with the new centroid
    new_solution_cost = costFunction(dataset, classes, selected_key, new_solution)

    # Greedy selection: comparing the new solution to the old one
    if new_solution_cost <= costs[selected_key]:
        centroids[selected_key] = new_solution
        costs[selected_key] = new_solution_cost
        C[selected_key] = 0
    else:
        # Increment the counter for discarded new solutions
        C[selected_key] += 1

# Scout bees phase
for cl_s in centroids:
    if C[cl_s] > a_limit:
        random_solution = np.random.uniform(0, 1, n_attr)
        random_solution_cost = costFunction(dataset, classes, cl_s, random_solution)

```

```

        centroids[cl_s] = new_solution
        costs[cl_s] = random_solution_cost
        C[cl_s] = 0

    # Update best solution for this iteration
    best_solution = 9999999999
    for cl in centroids:
        if costs[cl] < best_solution:
            best_solution = costs[cl]

    best_solutions[it] = best_solution

    #print('Iteration: {it}; Best cost: {best_solution}'.format(it = "%03d" % it, best_solution
    = best_solution))

    return best_solutions, centroids
def nearestCentroidClassifier(data, centroids):
    distances = centroids.copy()
    for key in centroids:
        distances[key] = euclidianDistance(data, centroids[key])

    distances_sorted = sorted(distances.items(), key = operator.itemgetter(1))
    nearest_class, nearest_centroid = distances_sorted[0]

    return nearest_class
def getSets(dataset, classes):
    size = len(dataset)

    training_set = dataset[:round(size * 0.75), :]
    training_set_classes = classes[:round(size * 0.75)]

    test_set = dataset[round(size * 0.75):, :]
    test_set_classes = classes[round(size * 0.75):]

    return training_set, test_set, training_set_classes, test_set_classes
databases = [{ 'filename': 'Anime.data', 'has_id': True, 'class_position': 'last' }]

for database in databases:
    d, c = readDatabase(database['filename'], database['has_id'], database['class_position'])
    training_set, test_set, training_set_classes, test_set_classes = getSets(d.copy(), c.copy())

    stats, centroids = determineCentroids(training_set, training_set_classes)

    ##### OUTPUT #####
    limits = [1000, 500, 50]
    for limit in limits:
        best_solutions, new_centroids = ABC(training_set, training_set_classes,
        centroids.copy(), a_limit = limit, max_iter = 1000)

```



```

print('\n\n## DATABASE: {filename}, limit = {limit}'.format(filename =
database['filename'], limit = limit))

# Test with the centroids
count = 0
print("# Test with the original centroids #")
for i, val in enumerate(test_set):
    cl = nearestCentroidClassifier(test_set[i], centroids)
    if cl != str(test_set_classes[i]):
        #print("Miscl.: {data}; Correct: {correct}; Classif.: {classif}")
        # .format(data = test_set[i], correct = test_set_classes[i], classif = cl)
        count += 1
print("# RESULT -> CEP: {cep}".format(cep = count/len(test_set)))

# Test with the ABC result
count = 0
print("\n\nTest with the ABC result centroids")
for i, val in enumerate(test_set):
    cl = nearestCentroidClassifier(test_set[i], new_centroids)
    if cl != str(test_set_classes[i]):
        #print("Miscl.: {data}; Correct: {correct}; Classif.: {classif}")
        # .format(data = test_set[i], correct = test_set_classes[i], classif = cl)
        count += 1
print("# RESULT -> CEP: {cep}".format(cep = count/len(test_set)))

```

PEARSON

```

import math
def sim_pearson(p1, p2, length):
    r = 0
    a = 0
    b = 0
    c = 0
    d = 0
    e = 0
    for x in range(length):
        a += p1[x]*p2[x]
        b += p1[x]
        c += p2[x]
        d += pow(p1[x], 2)
        e += pow(p2[x], 2)
    f = math.sqrt((d - pow(b, 2))*(e - pow(c, 2)))
    r = (length*a - b*c)/f
    return r

p1 = [1,2,3,4,5]
p2 = [2,4,6,8,10]
a = sim_pearson(p1,p2,5)
a

```

```

import numpy as np
import pandas as pd
UserData = pd.read_csv(r'Anime.data')
UserDataArray = np.array(UserData)
p1 = UserDataArray[0]
c = []
for row in range(len(UserDataArray)):
    p2 = UserDataArray[row]
    distance1 = sim_pearson(p1, p2, 4)
    c.append(distance1)
c

```

DECISION TREE

```

def divideset(rows,column,value):
    # Make a function that tells us if a row is in the first group (true) or the second group (false)
    split_function=None
    if isinstance(value,int) or isinstance(value,float): # check if the value is a number i.e int or
float
        split_function=lambda row:row[column]>=value
    else:
        split_function=lambda row:row[column]==value

    # Divide the rows into two sets and return them
    set1=[row for row in rows if split_function(row)]
    set2=[row for row in rows if not split_function(row)]
    return (set1,set2)
def entropy(rows):
    from math import log
    log2=lambda x:log(x)/log(2)
    results=uniquecounts(rows)
    # Now calculate the entropy
    ent=0.0
    for r in results.keys():
        p=float(results[r])/len(rows)
        ent=ent-p*log2(p)
    return ent
class decisionnode:
    def __init__(self,col=-1,value=None,results=None,tb=None,fb=None):
        self.col=col
        self.value=value
        self.results=results
        self.tb=tb
        self.fb=fb
def buildtree(rows,scoref=entropy): #rows is the set, either whole dataset or part of it in the
recursive call,
                                #scoref is the method to measure heterogeneity. By default it's entropy.
    if len(rows)==0: return decisionnode() #len(rows) is the number of units in a set

```

```

current_score=scoref(rows)

# Set up some variables to track the best criteria
best_gain=0.0
best_criteria=None
best_sets=None

column_count=len(rows[0])-1 #count the # of attributes/columns.
                        #It's -1 because the last one is the target attribute and it does not count.
for col in range(0,column_count):
    # Generate the list of all possible different values in the considered column
    global column_values    #Added for debugging
    column_values={}
    for row in rows:
        column_values[row[col]]=1
    # Now try dividing the rows up for each value in this column
    for value in column_values.keys(): #the 'values' here are the keys of the dictionary
        (set1,set2)=divideset(rows,col,value) #define set1 and set2 as the 2 children set of a
        division

        # Information gain
        p=float(len(set1))/len(rows) #p is the size of a child set relative to its parent
        gain=current_score-p*scoref(set1)-(1-p)*scoref(set2) #cf. formula information gain
        if gain>best_gain and len(set1)>0 and len(set2)>0: #set must not be empty
            best_gain=gain
            best_criteria=(col,value)
            best_sets=(set1,set2)

# Create the sub branches
if best_gain>0:
    trueBranch=buildtree(best_sets[0])
    falseBranch=buildtree(best_sets[1])
    return decisionnode(col=best_criteria[0],value=best_criteria[1],
                        tb=trueBranch,fb=falseBranch)
else:
    return decisionnode(results=uniquecounts(rows))
def classify(observation,tree):
    if tree.results!=None:
        return tree.results
    else:
        v=observation[tree.col]
        branch=None
        if isinstance(v,int) or isinstance(v,float):
            if v>=tree.value: branch=tree.tb
            else: branch=tree.fb
        else:
            if v==tree.value: branch=tree.tb
            else: branch=tree.fb
        return classify(observation,branch)
import pandas as pd

```

```

UserData = pd.read_csv(r'C:\Users\Admin\data.txt')
import numpy
UserDataArray = numpy.array(UserData)
UserDataArray
import numpy
UserDataArray = numpy.array(UserData)
UserDataArray
tree = buildtree(UserDataArray)
def printtree(tree,indent=""):
    # Is this a leaf node?
    if tree.results!=None:
        print(str(tree.results))
    else:
        print(str(tree.col)+':'+str(tree.value)+'? ')
        # Print the branches
        print(indent+'T->')
        printtree(tree.tb,indent+' ')
        print(indent+'F->')
        printtree(tree.fb,indent+' ')
printtree(tree)
import pandas as pd
Dataset = pd.read_csv(r'C:\Users\Admin\dataset1.txt')
import numpy
conan = numpy.array(Dataset)
arrp={}
for i in range(0,54):
    arrp[i]=classify(conan[i],tree)
num=1
for i in range(0,50):

    if num<=8:

        if arrp[i]=={'love':4}:
            print(conan[i])
            num=num+1
        if arrp[i]=={'love':1}:
            print(conan[i])
            num=num+1
        if arrp[i]=={'love':2}:
            print(conan[i])
            num=num+1
        if arrp[i]=={'love':3}:
            print(conan[i])
            num=num+1

hole=1
for i in range(0,54):

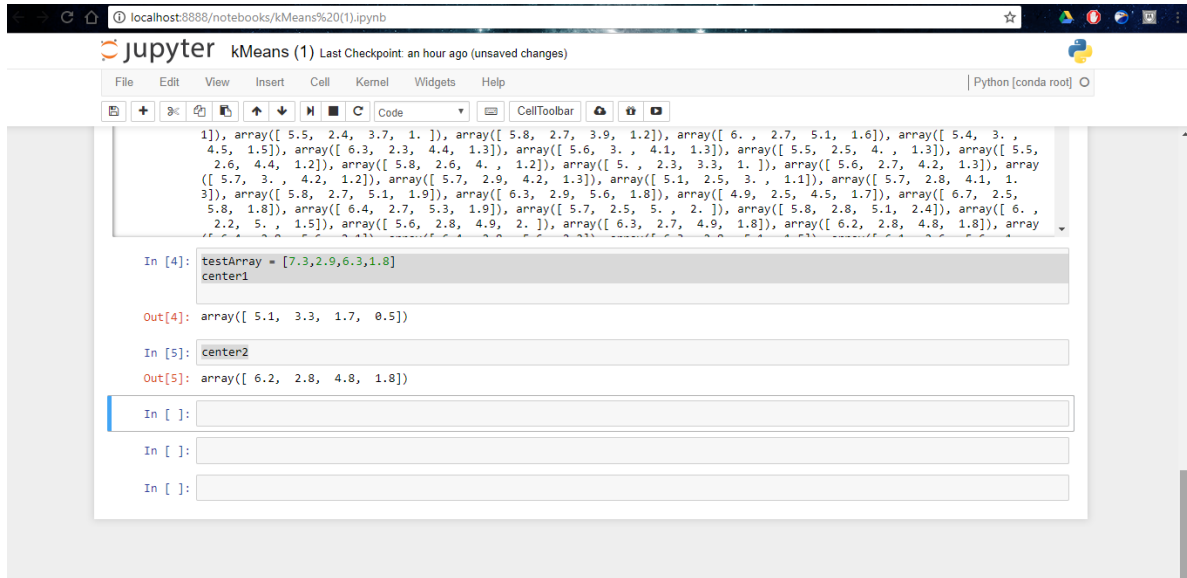
    if hole<=2:

```

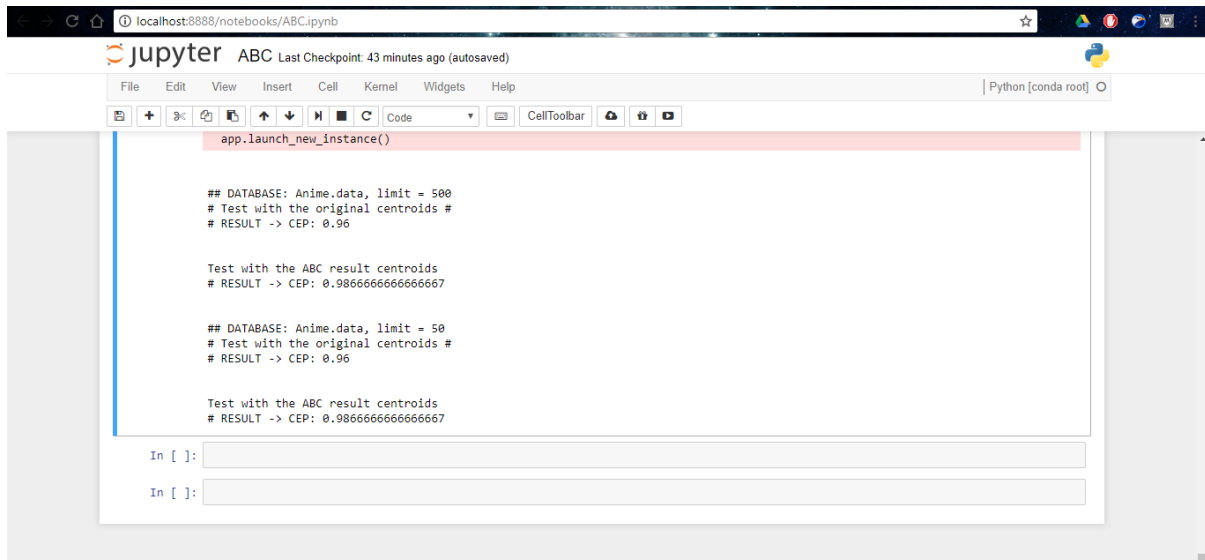
```
if arrp[i]=={'dislike':1}:
    print(conan[i])
    hole=hole+1
if arrp[i]=={'dislike':2}:
    print(conan[i])
    hole=hole+1
if arrp[i]=={'dislike':3}:
    print(conan[i])
    hole=hole+1
if arrp[i]=={'dislike':4}:
    print(conan[i])
    hole=hole+1
```

SYSTEM SNAPSHOT

K-Means



ABC



```

localhost:8888/notebooks/ABC.ipynb
jupyter ABC Last Checkpoint: 43 minutes ago (autosaved)
Python [conda root]

count = 0
print("# Test with the original centroids #")
for i, val in enumerate(test_set):
    cl = nearestCentroidClassifier(test_set[i], centroids)
    if cl != str(test_set_classes[i]):
        #print("Misc.: {data}; Correct: {correct}; Classif.: {classif}")
        # .format(data = test_set[i], correct = test_set_classes[i], classif = cl))
        count += 1
print("# RESULT -> CEP: {cep}".format(cep = count/len(test_set)))

# Test with the ABC result
count = 0
print("\n\nTest with the ABC result centroids")
for i, val in enumerate(test_set):
    cl = nearestCentroidClassifier(test_set[i], new_centroids)
    if cl != str(test_set_classes[i]):
        #print("Misc.: {data}; Correct: {correct}; Classif.: {classif}")
        # .format(data = test_set[i], correct = test_set_classes[i], classif = cl))
        count += 1
print("# RESULT -> CEP: {cep}".format(cep = count/len(test_set)))

## DATABASE: Anime.data, limit = 1000
# Test with the original centroids #
# RESULT -> CEP: 0.96

Test with the ABC result centroids
# RESULT -> CEP: 0.9866666666666667

```

DECISION TREE

```

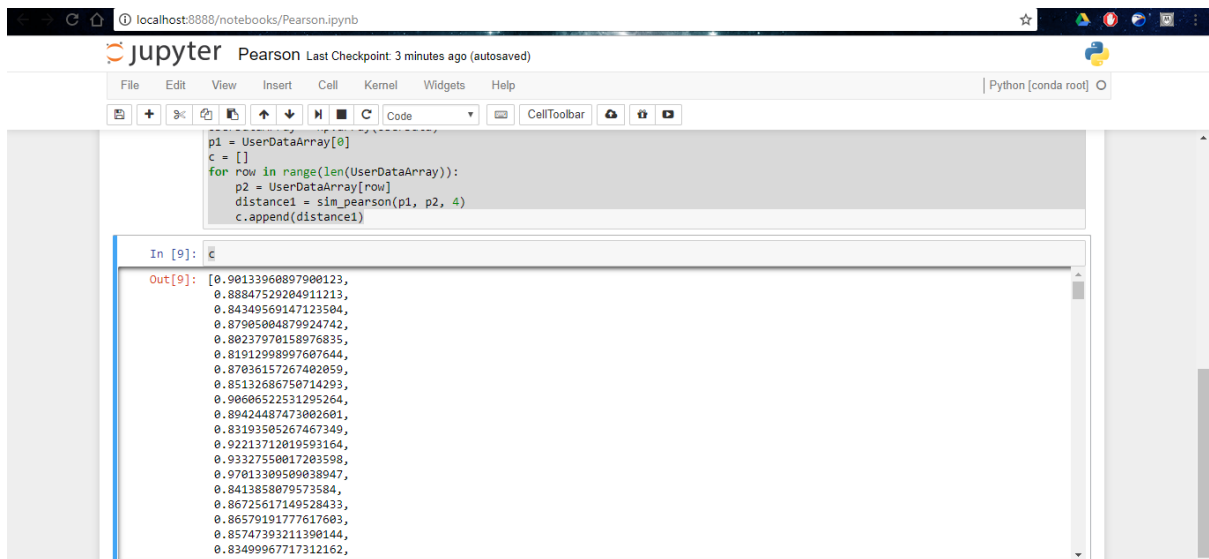
localhost:8888/notebooks/karan.ipynb
jupyter karan Last Checkpoint: 05/24/2017 (autosaved)
Python [conda root]

num=num+1
hole=1
for i in range(0,54):
    if hole<=2:
        if arpp[i]=={'dislike':1}:
            print(conan[i])
            hole=hole+1
        if arpp[i]=={'dislike':2}:
            print(conan[i])
            hole=hole+1
        if arpp[i]=={'dislike':3}:
            print(conan[i])
            hole=hole+1
        if arpp[i]=={'dislike':4}:
            print(conan[i])
            hole=hole+1

['Durarana!!' 'Album' 24L 'Act/Adv' 8.38]
['FairyTail' 'Album' 175L 'Act/Adv' 8.23]
['Kuroshitsuji' 'Album' 24L 'Act/Adv' 8.06]
['K' 'Album' 13L 'Act/Adv' 9.04]
['FairyTail' 'Album' 175L 'act/adv/com' 8.98]
['D.Gray-man' 'Album' 103L 'act/adv/com' 7.91]
['InuYasha' 'Album' 167L 'act/adv/com' 7.89]
['TokyoGhoul' 'Album' 12L 'horror' 8.9]
['SoulEater' 'Album' 51L 'act/adv/com' 8.08]
['Another' 'Single' 12L 'horror' 7.89]

```

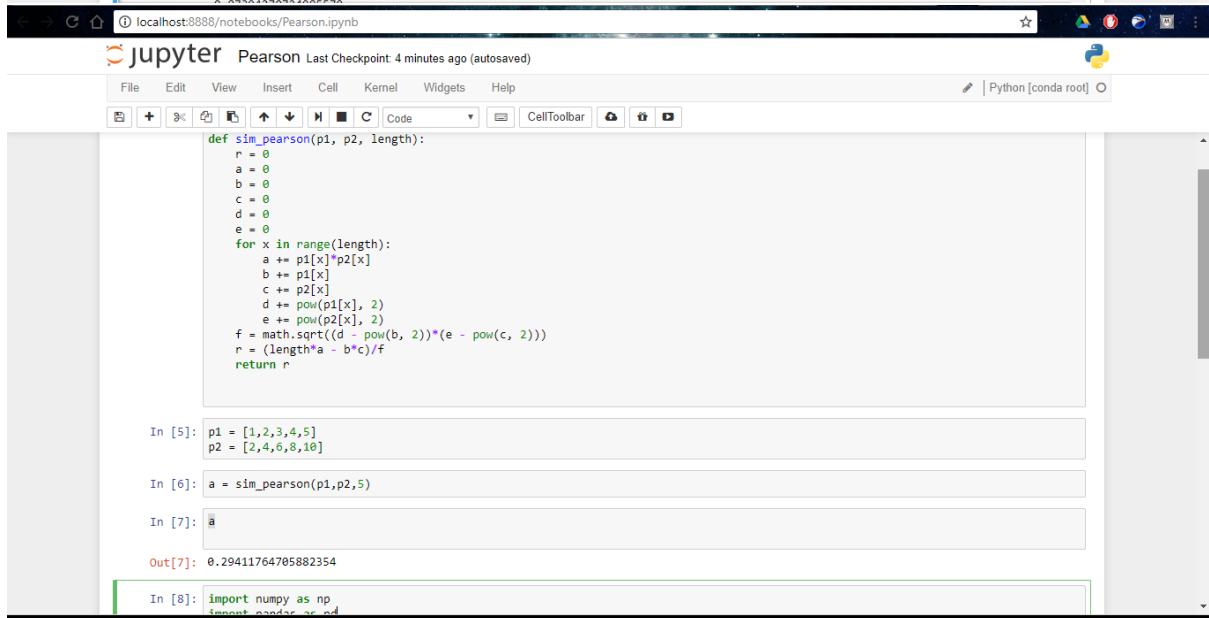
PEARSON CORRELATION



```
p1 = UserDataArray[0]
c = []
for row in range(len(UserDataArray)):
    p2 = UserDataArray[row]
    distance1 = sim_pearson(p1, p2, 4)
    c.append(distance1)
```

In [9]:

```
Out[9]: [0.90133960897900123,
0.88847529204911213,
0.84349569147123504,
0.87905004879924742,
0.80237970158976835,
0.81912998997607544,
0.87036157267402059,
0.85132686750714293,
0.90606522531295264,
0.89424487473002601,
0.83193505267467349,
0.92213712019593164,
0.93327550017203598,
0.97013309509038947,
0.8413858079573584,
0.86725617149528433,
0.86579191777617603,
0.85747393211390144,
0.83499967717312162,
```



```
def sim_pearson(p1, p2, length):
    r = 0
    a = 0
    b = 0
    c = 0
    d = 0
    e = 0
    for x in range(length):
        a += p1[x]*p2[x]
        b += p1[x]
        c += p2[x]
        d += pow(p1[x], 2)
        e += pow(p2[x], 2)
    f = math.sqrt((d - pow(b, 2))*(e - pow(c, 2)))
    r = (length*a - b*c)/f
    return r
```

In [5]:

```
p1 = [1,2,3,4,5]
p2 = [2,4,6,8,10]
```

In [6]:

```
a = sim_pearson(p1,p2,5)
```

In [7]:

```
a
```

Out[7]: 0.29411764705882354

In [8]:

```
import numpy as np
```


List of Publications

Accepted & Presented

1. Kumar A., Soderer N., (2017, Apr). Open Problems in Recommender Systems Diversity. Accepted in ICCCA 2017.