ACCR: An Efficient Algorithm for Single User Recommendation

Jaitri Taraphdar

ACCR: An Efficient Algorithm for Single User Recommendation

Report submitted to

Adamas University for the award of the degree

Of

M.Sc (Tech) Statistics and Data Science

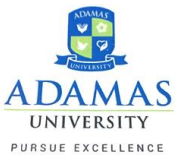
By

Jaitri Taraphdar

Under the supervision of

Dr. Vaskar Sarkar

Associate Professor, Adamas University



DEPARTMENT OF MATHEMATICS

SCHOOL OF BASIC AND APPLIED SCIENCES

ADAMAS UNIVERSITY, KOLKATA -700126

June, 2023

ACCR: An Efficient Algorithm for Single User Recommendation

Report submitted to

Adamas University for the award of degree

Of

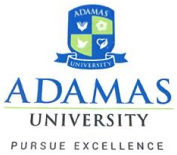
M.Sc (Tech) Statistics and Data Science

By

Jaitri Taraphdar

Under the supervision of

Dr. Vaskar Sarkar



DEPARTMENT OF MATHEMATICS

SCHOOL OF BASIC AND APPLIED SCIENCES

ADAMAS UNIVERSITY, KOLKATA-700126

June, 2023

IS APPROVED FOR THE DEGREE OF

Masters in Science (MSc) in Statistics and Data Science

External Examiner	Thesis Supervisor
HOD, Mathematics	Dean, School of Basic and Applied Sciences

Dedicated To….

My Family &Teachers

Certificate of Approval

---

*This is to certify that the dissertation entitled "**ACCR: an Efficient Algorithm for Single User Recommendation**" submitted by **Jaitri Taraphdar based** upon her original work carried out in the Department of Mathematics, Adamas University. She has worked really hard for the project. Her performance was excellent during the dissertation period. I wish her all the success in future.*

Supervisor's Name

Dr. Vaskar Sarkar

Associate Professor
Department of Mathematics

ADAMAS UNIVERSITY
PURSUE EXCELLENCE

June, 2023

Words of Thanks

I would like to extend my hearty appreciation and gratitude to my supervisor, **Dr. Vaskar Sarkar, Associate Professor, Adamas University**, for his invaluable guidance and mentorship throughout my research journey. His expertise and support have been instrumental in shaping my work.

I would also like to express my gratitude to **Dr. Arup Roy, Assistant Professor at Budge Budge Institute of Technology,** for his encouragement, brilliant suggestions, and insightful information. His guidance has been invaluable not only for my research but also for my overall career development.

Furthermore, I would like to convey my sincere thanks to all the faculty members of the department, particularly **Dr. Nav Kumar Mahato, Head of the Department of Mathematics**, and all the professors in the Mathematics Department. Their unwavering support and assistance have been instrumental in my academic and research endeavors.

I would especially like to thank Adamas University for providing me all the necessary facilities that required for the completion of this project.

A special thanks to our families, friends and classmates. Words cannot express how grateful I am for their continuous love and support.

Abstract

A possible problem of information overload that prevents timely access to things of interest on the Internet has been caused by the exponential development in the volume of digital information available and the number of Internet users. This issue has been largely resolved by information retrieval systems like Google, Devil Finder, and Altavista, but prioritisation and personalization of the information (where a system matches the available content to the user's interests and preferences) were lacking. As a result, recommender systems are more in demand than ever. By selecting important information fragments from a large amount of dynamically created material based on the user's choices, interests, or observed behaviour about the item, recommender systems are information filtering systems that address the issue of information overload. The information is filtered by recommendation systems in order to find their amazing discoveries. Recommenders use combinations of parameters that can be used to create a thorough analysis and detailed proposal. The filtering process is crucial in recommendation systems that select the best products depending on customer demand. In a recommendation system, some well-known filtering techniques are essentially collaborative, content-based, and hybrid. Collaborative filtering (CF), one of the methods, has become a popular recommendation method over time. This project suggests using the dragonfly algorithm to help classify clients based on their personalities, solve the sparsity problem, and minimise the number of comparisons required during processing. The outcome of the collaborative filtering has been jeopardised by the sparsity and slow start, nevertheless. Altered Client-based Collaborative Filtering (ACCF) for Single User Recommendation is a collaborative filtering algorithm that is suggested as a more effective solution to these issues. The Dragonfly Algorithm is used by ACCF to handle the sparsity and neighbour selection. The validation of ACCF is tested using a restaurant recommendation system. A comparative analysis has been included with the end objective of performance assessment, and it shows that the proposed method successfully minimises the sparsity problem. Even when applied to a tiny sample of data, the experimental results show that ACCF offers more Coverage, Precision, and F-Measure than user-based collaborative filtering.

Contents

List of Symbols:

Symbols	Description
it	Set of items co rated by x and y
Ux,it	Rating of x on the it-th item
Uy,it	Rating of y on the it-th item
p	the set of neighbors
x	position of a dragonfly
x_j	position of j-th dragonfly in the neighborhood
	of z
n	total number of dragonflies in the
	neighborhood
v_j	velocity of the j-th dragonfly in the
	neighborhood
x^+	the food source position
x^-	Enemy position
s	separation operator weight
s_i	separation of i-th dragonfly in the neighborhood
a	alignment operator weight
a_i	i-th dragonfly alignment in the neighborhood
c	operator weight
c_i	i-th dragonfly cohesion in the neighborhood
f	attraction operator weight
f_i	i-th dragonfly attraction towards the food source
h	distraction operator weight
h_i	i-th dragonfly distraction from the enemies
w	inertia weight
q	iteration index
gt	a target client
trnt gt	No of recommendations provided by n_t to gt

tnrnt gt	No of minimal ratings (i.e. ratings less than the minimum allowable rating) provided by nt to gt
ωnt gt	No of suggestions provided by nt to gt
nt ϑgt	No of suggestions provided by gt to nt
pgt nt	No of co-ratings exist between gt and nt
gt	Target client
d	maximum allowable rating
η_{gt}^{k}	gt rating to the k-th restaurant
η_{nt}^{k}	nt rating to the k-th restaurant
rk- 1 gt	gt rating to the (k-1)-th restaurant
щk-1 3	standard deviation of the ratings of all nt's to
щk 3	the standard deviation of the ratings (k-1)-th restaurant including gt all nt's to the k-th restaurant excluding gt
vγ	threshold of recommendation
r	No of ratings predicted
p	No of ratings tested
o	total no. of user ratings

List of Abbreviations:

Abbreviation	Full form
CF	Collaborative Filtering
UBCF	User Based Collaborative Filtering
DA	Dragonfly Algorithm
ACCF	Altered Client Based Collaborative Filtering
ONSA	Optimal Neighbor Selection Algorithm
PCCA	Personality-based Client Classification Algorithm

1. Introduction:

ACCR (An Efficient Algorithm for Single User Recommendation) is a project aimed at developing an advanced recommendation algorithm specifically designed for single-user recommendation systems. The project focuses on addressing the challenges faced by traditional recommendation algorithms when dealing with individual users' preferences and providing accurate and personalized recommendations.

In the current digital era, personalised recommendations are essential for improving user experiences on a variety of platforms, including social media sites, streaming services, and e-commerce websites. Collaborative filtering and content-based filtering, two common traditional recommendation algorithms, frequently struggle to effectively capture the unique preferences and interests of individual users. They frequently rely largely on aggregated data or general patterns, which can lead to recommendations that don't entirely reflect a person's particular tastes and preferences.

Traditional recommendation algorithms often suffer from scalability issues, as they struggle to handle large datasets and real-time recommendation requests. Moreover, these algorithms may not adequately capture the specific preferences and interests of individual users, resulting in suboptimal recommendations.

The goal of the ACCR project is to get beyond these constraints by creating a powerful algorithm that makes use of cutting-edge methods like deep learning, natural language processing, and data mining. With the use of these innovative technologies, ACCR hopes to offer each user extremely precise and customised recommendations, resulting in an enhanced user experience.The key features and benefits of ACCR include:

- Efficiency: ACCR is designed to be highly efficient, capable of handling large-scale datasets and delivering real-time recommendations. The algorithm optimizes resource utilization and minimizes computational overhead, ensuring quick response times even in dynamic environments.
- Personalization: ACCR places a strong emphasis on personalization by employing advanced machine learning models that can effectively capture individual user preferences. By analyzing user behavior patterns, historical data, and contextual information, ACCR generates tailored recommendations that align with the specific interests and needs of each user.
- Accuracy: Modern recommendation algorithms that may generate accurate recommendations are incorporated into ACCR. Accurate Collaborative Content Recommendation (ACCR) makes sure that the suggested items are highly pertinent and likely to appeal to the user by integrating collaborative filtering, content-based filtering, and other pertinent strategies.

- Adaptability: ACCR is designed to adapt and learn from user feedback. By incorporating feedback loops, the algorithm continuously refines its recommendations, adapting to evolving user preferences and providing a personalized experience that improves over time.

The ACCR initiative has the potential to greatly improve the standard and applicability of recommendations for specific consumers, leading to increased user satisfaction, engagement, and ultimately, business success. By leveraging the power of efficient and personalized recommendation algorithms, ACCR aims to revolutionize the way recommendations are generated and consumed in various domains, including e-commerce, content streaming, social media, and more.

As a result, it can suggest a wide variety of products, including books, films, songs, travel, humour, and more. From a functional standpoint, it recommends the products to consumers based on what people with similar profiles or past ratings might find appealing.

The value of restaurant recommendations in terms of profitable financial return has spread rapidly throughout the world. The user's preferences for eateries served as the inspiration for this project. To address it, the "Altered Client-based Collaborative Filtering" (ACCF) for single user suggestion client-based collaborative filtering algorithm has been proposed. The research study proposes a dragonfly method to help with client personality classification, sparsity minimization, and a reduction in the number of comparisons required during implementation.

2. Literature Review:

The recommendation systems has witnessed significant advancements in recent years, with researchers and practitioners exploring various techniques to provide personalized recommendations to users. In this literature review, I delve into the key studies and approaches related to the ACCR (An Efficient Algorithm for Single User Recommendation) project, focusing on the development of efficient and accurate algorithms for single-user recommendations.

Table 1: Some research paper with algorithm on CF and restaurant recommendation:

Researchers	**Algorithm**	**Purpose**
J. Breese, D. Heckerman, C. Kadie, Empirical analysis of predictive algorithms for collaborative filtering, Proceedings of the 14th Conference on Uncertainty in Artificial Intelligence (1998) 43–52	Case amplification based CF	Improved neighbor selection
R. Jin, J.Y. Chai, L. Si, An automatic weighting scheme for collaborative filtering, Proceedings of the 27th SIGIR (2004) 337–344	Item weighting based CF	Improvised collaboration
M.R. McLaughlin, J.L. Herlocker, A collaborative filtering algorithm and evaluation metric that accurately model the user experience, Proceedings of the 27th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval (2004) 329–336.	Modified CF	Improvised similarity measure
X. Amatriain, N. Lathia, J.M. Pujol, H. Kwak, N. Oliver, The wisdom of the few-a collaborative filtering approach based on expert opinions from the web, Proceedings of the 32nd SIGIR (2009) 532–539.	Expert neighbor based CF	Minimization of the noisy ratings, scalability, and privacy issues

S.G. Mohammad, A. Nematbakhsh, Enhancing memory-based collaborative filtering for group recommender systems, Experts Syst. Appl. 42 (7) (2015) 3801–3812.	Weighted Pearson correlation based CF	Improvised similarity measure
E. Hallstrom, Group Recommender System for Restaurant Lunches, Royal Institute of Technology (KTH), Stockholm, Sweden, 2013 (Master's Thesis).	Bayesian network based model	Multi-criteria decision making based mapping of the individual preferences to the group preference
H.F. Tan, R. Rotabi, H. Giang, T. Nguyen, Using Ranking Support Vector Machines for Group Recommendations: Restaurant Recommendations on Yelp Data, 2018 shftan.github.io/papers/nyas15.pdf	Support vector machine based model	Personalized recommendation
M. Gartrell, K. Alanezi, L. Tian, R. Han, Q. Lv, S. Mishra, SocialDining: design and analysis of a group recommendation application in a mobile context, in: Computer Science Technical Reports, 2014.	Ranking support vector	Maximizes minimum happiness
J. Huang, S. Rogers, E. Joo, Improving restaurants by extracting subtopics from yelp reviews, Proceedings of the I-Conference Social Media Expo (2014).	Hybridized recommendation mode	Improved personalized recommendation
A. Jameson, B. Smyth, et al., Recommendation to groups, in: P. Brusilovsky (Ed.), The Adaptive Web, 2007, pp. 596–627	BomApetite	Enhanced consensus

In conclusion, the literature review highlights the advancements in recommendation systems, including CF, Content-Based filtering, hybrid techniques, scalability recommendations. ACCR project contributes to this body of knowledge by proposing an efficient algorithm specifically tailored for single-user recommendation scenarios. By integrating insights from these research areas, ACCR aims to provide accurate, personalized, and timely recommendations, enhancing user experiences in various domains.

3. Recommender System Fundamentals:

Recommendation systems have become crucial in the age of modern technology, where a wealth of information and options are readily available. This makes it difficult for consumers to sort through the confusing options. A recommendation system is a piece of technology that screens data to present users with pertinent suggestions based on their preferences, interests, and previous actions. In a variety of industries, including e-commerce, entertainment, social networking, and content streaming platforms, these systems are essential. Recommender systems successfully mitigate the effects of information overload by sifting through the enormous volume of dynamically created data to isolate the most important bits of information. Additionally, these systems are intelligent enough to foresee a user's preference for one product over another, enabling them to offer tailored recommendations. Recommender systems need to offer quality recommendations.

Different Types of Recommender Systems:

Recommender systems can be broadly classified into three categories: content-based filtering, collaborative filtering, and hybrid filtering. Figure 1 provides an overview of this classification.

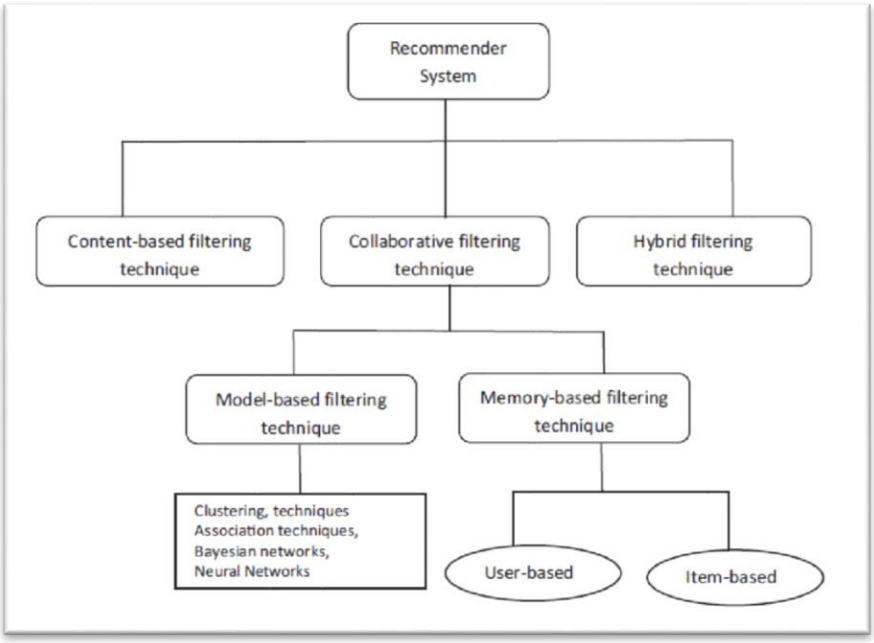


Figure 1: Types of Recommender System

3.1. Content Based Filtering:

To generate recommendations, this filtering technique depends on two crucial bits of data. The attributes given to each item, which offer extra information about the things, are the first piece of

data used. The user profile, which includes details on the things the user has previously engaged with as well as their characteristics, is the second piece of information used. In comparison to other attributes, those that are frequently shared by several items for the user are given more weight. A user preference model is developed by taking these attribute weights and the user's past into account. The scores are then assigned depending on how closely the model matches the user profile for each object in the database. These scores are used to make recommendations.

3.2. Collaborative Filtering:

This way of filtering is mostly based on how people make decisions by asking their friends and family for advice on products they already own. Memory-based and model-based collaborative filtering algorithms can be separated into two groups. Predictions in the memory-based method make use of the whole user database. The system uses statistical techniques to locate a group of neighbours or other users who have similar interests to the current user. Either an item-item or a user-user model can be used to construct a memory-based system. In item-item collaborative filtering, the relationship between several things that are regularly bought together is the main focus.

3.2.1. Limitations of Collaborative Filtering:

The recommendation system has proven to be highly effective in increasing users' propensity to make purchases and generating company income, drawing growing interest from enterprises. However, the demand for recommendation system applications across numerous areas is also growing as Internet data continues to rise quickly and application scenarios change. In the field of recommendation systems, there are a great deal of fresh problems and difficulties that need immediate solutions. Data sparsity, the cold start problem, scalability, accuracy, and diversity, the top attack problem, context awareness, and privacy protection problems are only a few of these difficulties.

- **Data Sparsity:** Data sparsity, which results from insufficient or missing valuable information, is a critical difficulty in recommendation systems. The number of users and the diversity of projects and services have both skyrocketed along with the rapid development of internet platforms and systems. However, given the enormous number of services and projects, the rating information and historical data are frequently scarce. This results in a high-dimensional and unequally distributed data set with a sparse matrix of important historical information between users and projects.
- **Data sparsity:** Due to insufficient or missing relevant information, recommendation systems have a big issue. The number of users and the diversity of projects and services have both skyrocketed along with the rapid development of internet platforms and systems. However, given the enormous number of services and projects, the rating information and historical data are frequently scarce. This results in a high-dimensional and unequally distributed data set with a sparse matrix of important historical

information between users and projects.In recommendation systems, the cold start problem is an extreme instance of data sparsity. The system fails to produce reliable forecasts or give sound suggestions based on past data when there is a shortage of user input data. As a result, the system is unable to accurately collect user interest choices.

- **Cold Start Problem**: A recommendation system's difficulty with new users and new objects is known as the "cold start problem." In particular, new objects in collaborative filtering systems require a specific amount of time for users to observe and assess them via clicks, scores, comments, and other interactions. Analysing and endorsing these objects is impossible before a significant number of assessments have been acquired. The cold start problem is often handled by the use of multiple recommendation mechanisms, unlike the new user problem. To increase the potency of recommendations, strategies like item entropy, user personality traits, and popularity are used.
- **Scalability:** In recommendation systems, scalability is a crucial problem, especially for big online platforms. In order to produce suggestions for target consumers, these systems frequently need to serve millions of users and objects, necessitating real-time comparison searching throughout the whole user space. Therefore, traditional recommendation systems have substantial hurdles in terms of real-time performance and algorithm scalability as the number of users and goods increases quickly. There is a trade-off between real-time performance and recommendation accuracy, according to numerous academic studies. For recommendation systems, maintaining high-quality recommendations while satiating real-time needs continues to be a difficulty. A critical issue that needs to be addressed is finding a compromise between real-time performance, scalability, and suggestion quality.

In conclusion, the aforementioned issues place restrictions on the advancement and use of recommendation systems. They stand for difficult problems that demand careful consideration and creative solutions.

3.3. Hybrid Filtering:

Together with overcoming each method's own weaknesses, hybrid filtering combines the best aspects of collaborative and content-based filtering. To include collaborative and content-based filtering techniques into a hybrid system, there are various solutions. 1. Separate Implementation and Combination: The outcomes of the implementation of the collaborative and content-based filtering techniques are combined subsequently. 2. Using Collaborative Filtering features in Content-Based techniques and Vice Versa: In this strategy, collaborative filtering features are used in content-based techniques and vice versa to improve the recommendation process. 3. Combination Model: In order to create suggestions, a combination model is constructed that combines both content-based and collaborative filtering capabilities. The purpose of this study is to review several algorithms for recommendation systems and to categorise them according to the methodology and application domain.

3.4. User Based Collaborative Filtering:

A technique called User-Based Collaborative Filtering (UBCF) is used in recommendation systems to anticipate the products that a user will like based on the ratings supplied by other users who have similar likes to the target user. Collaborative filtering is often used by websites to build their recommendation algorithms. Considering their history data, UBCF locates users who share the target user's preferences. These users are referred to as the target user's neighbours. The three steps of the UBCF's operation are (i) neighbourhood identification, (ii) user rating prediction, and (iii) target user suggestion.

3.5. Steps for UBCF:

- Step 1: Identification of the neighbors:

UBCF framework use Pearson-Correlation Co-efficient to calculate the similarity of x and y:

$$sim(x,y) = \frac{\sum_{it \in IT} (U_{x,it} - \overline{U}_x) \times (U_{y,it} - \overline{U}_y)}{\sqrt{\sum_{it \in IT} (U_{x,it} - \overline{U}_x)^2 \times \sum_{it \in IT} (U_{y,it} - \overline{U}_y)^2}}$$

- Step 2:Rrating prediction of the target user:

Prediction of the target user's rating is the second step of UBCF.

$$prd(x,it) = \overline{U}_{x,it} + \frac{\sum_{y \in \rho} (U_{y,it} - \overline{U}_y) \times sim(x,y)}{\sum_{y \in \rho} sim(x,y)}$$

Where;

The phrase "co-rated" refers to a group of objects that have been rated by both user x and user y in the context of information technology (IT). User X's rating of the ith item is indicated by the notation UX, it, while User Y's rating of the identical item is shown by UY, it. In addition, (U_x) indicates the typical rating provided by user x, and (U_(y)) reflects the typical rating provided by user y. Last but not least, stands for the group of neighbours, or users with comparable tastes or inclinations.

4. DA: Dragonfly Algorithm:

The behaviour of dragonflies served as the basis for the meta-heuristic optimisation technique known as the Dragonfly technique. It imitates how dragonflies fly as they look for food, mate, and stay away from predators. Although the swarming optimisation problem is the major application for the dragonfly algorithm, it can also be utilised in other fields, such as recommendation systems.

4.1. Overview of Dragonfly Algorithm (DA):

An technique called the Dragonfly Algorithm (DA) mimics the group behaviour of dragonflies, which is either related to migration or hunting. Hunting causes static swarming while migration causes dynamic swarming. Small groups of dragonflies pursue insects in a constrained region during static swarming, demonstrating local movements and abrupt shifts. Dynamic swarming, on the other hand, involves numerous dragonflies forming a single group and travelling large distances in a steady direction. The Dragonfly Algorithm was primarily inspired by these swarming behaviours.

The exploration and exploitation phases of meta heuristic optimisation algorithms are aligned with the static and dynamic swarming behaviours. Figure 3 shows two types of dragonfly swarming behaviours: static and dynamic. The exploration and exploitation phases of meta-heuristic optimisation algorithms are aligned with the static and dynamic swarming behaviours. Figure 3 shows the behaviours of dragonflies in static and dynamic swarming. Five weights are used: separation weight (s), alignment weight (a), cohesion weight (c), feeding factor (f), enemy factor (e), and inertia weight (w) to direct artificial dragonflies along various courses. Low alignment and high cohesion weights are used to exploit the search space, while high alignment and low cohesion weights are utilised to explore the search space. Additionally, the neighborhood's radius grows in direct proportion to how many iterations are required to move from exploration to exploitation. Adaptively modifying the swarming weights (s, a, c, f, and w) is another strategy for balancing exploration and exploitation.

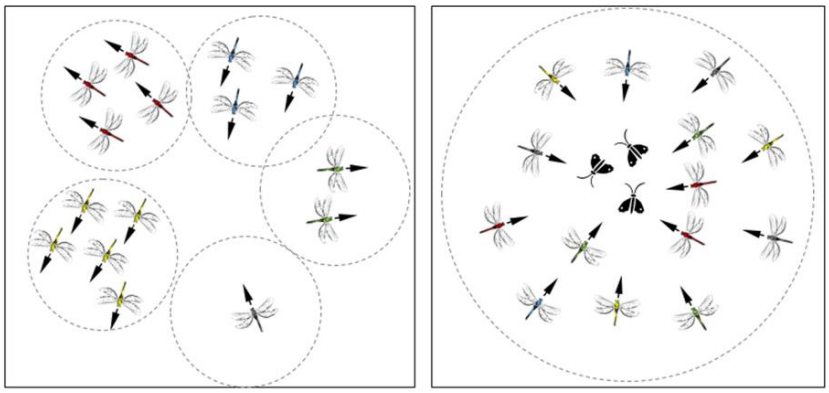


Figure 2: Static and Dynamic Swarm of Dragonfly

- **Separation**- The minimization of static collisions between entities that are close to one another is what is meant. The separation of the i-th dragonfly: if separation is represented by the vector S:

$$S_i = -\sum_{j=1}^{N} X - X_j$$

In the equation previously mentioned, X stands for the current individual's position, Xj for the neighbouring dragonfly's position (specifically, the j-th neighbouring dragonfly), N for the number of neighbours in the dragonfly swarm, and S for the i-th individual's motion towards separation.

- **Alignment-** It describes creatures that match the speed of other nearby ones. If the vector A represents alignment:

$$A_i = \frac{\sum_{j=1}^{N} V_j}{N}$$

The i-th individual's alignment motion is represented by Ai in the preceding equation, and the j-th neighbouring dragonfly's velocity is denoted by V.

- **Cohesion**- It refers to the centre of the dragonfly population when the insects are hunting. It is symbolised by C:

$$C_i = \frac{\sum_{j=1}^{N} X_j}{N} - X$$

Where Ci denotes the cohesion for the ith individual, N denotes the size of the neighbourhood, Xj denotes the location of the jth neighbouring dragonfly, and X is the present dragonfly individual.

- **Attraction towards the food source –** This term refers to a dragonfly's attraction towards a food source. It is symbolised by the letter F and is given in the equation below:

$$F_i = X^+ - X$$

The i-th dragonfly's attraction to food is represented by Fi in the equation above. X+ designates the location of the dragonfly with the highest observed objective function, which serves as the food source. X indicates where the current dragonfly is located.

- **Distraction from enemy:** The capacity of dragonflies to avoid enemies in order to assure protection is referred to as distraction from the enemies. The R stands for this factor and is stated as follows:

$$R_i = X^- + X$$

Artificial dragonflies use the step vector X and the position vector X to update positions in the search space. The velocity vector used in the Particle Swarm Optimisation (PSO) algorithm is analogous to the step vector. Therefore, the PSO algorithm's basic structure serves as the foundation for the position updating process. The step vector is described as follows:

$$\Delta X_{t+1} = (sS_i + aA_i + cC_i + fF_i + rR_i) + w\Delta X_t$$

In the equation, Si stands for the separation effect for the i-th dragonfly, Ai for the alignment effect, Ci for the cohesion effect, Fi for the effect of food source for the i-th individual, Ei for the effect of enemy position for the i-th dragonfly, w for inertia weight, and t for iteration counter.

$$X_{t+1} = X_t + \Delta X_{t+1}$$

Where, t represents the current iterations.

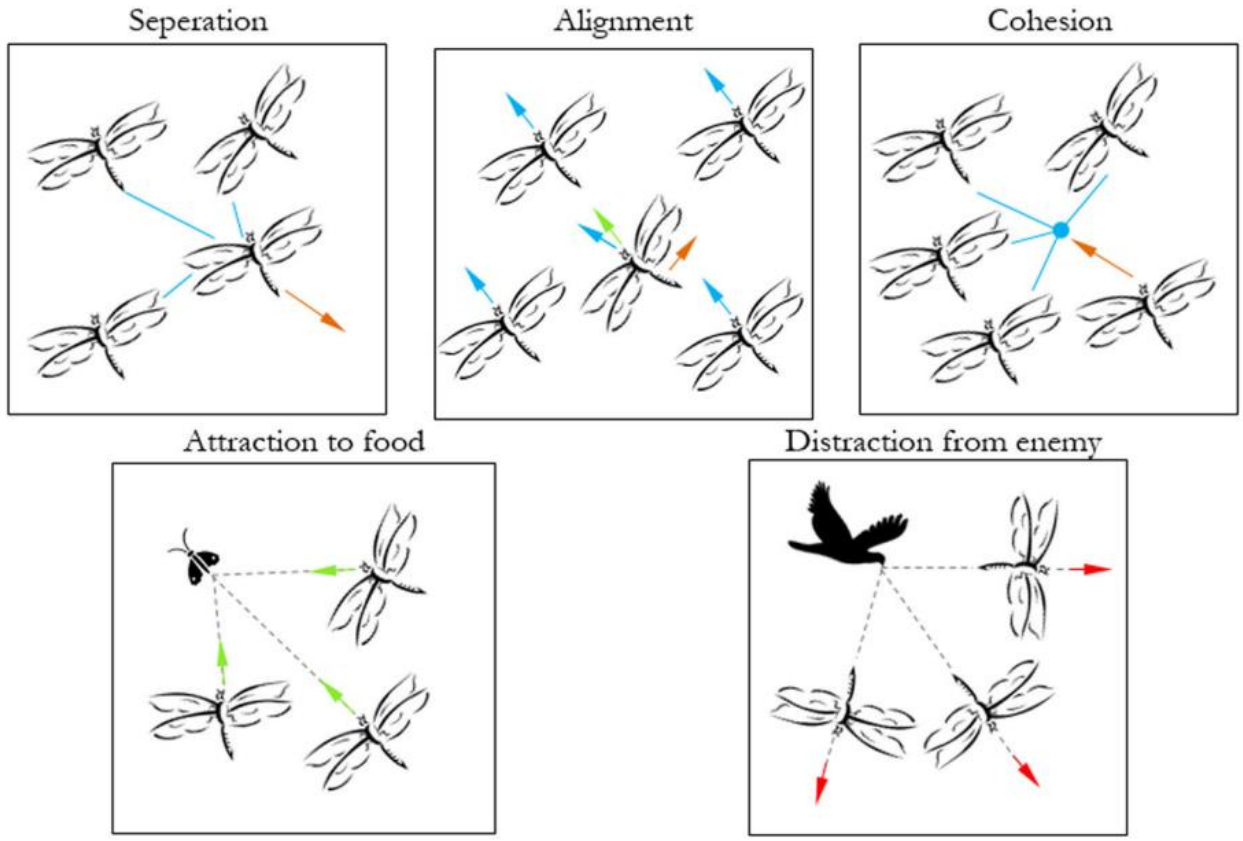


Fig 3: Patterns of Dragonflies Swarming

During the optimisation search process, a variety of exploratory and exploitative behaviours can be attained thanks to the incorporation of five factors: separation (s), alignment (a), cohesion (c), food (f), and adversary (e). It is intended that each artificial dragonfly would have a neighbourhood with a predetermined radius because neighbours are important to dragonflies. This neighbourhood can be depicted as a circle in two dimensions, a sphere in three dimensions, and a hyper-sphere in n dimensions. Figure 4's image uses the suggested mathematical notation to show an example of the swarming behaviour of dragonflies with an expanding neighbourhood radius.

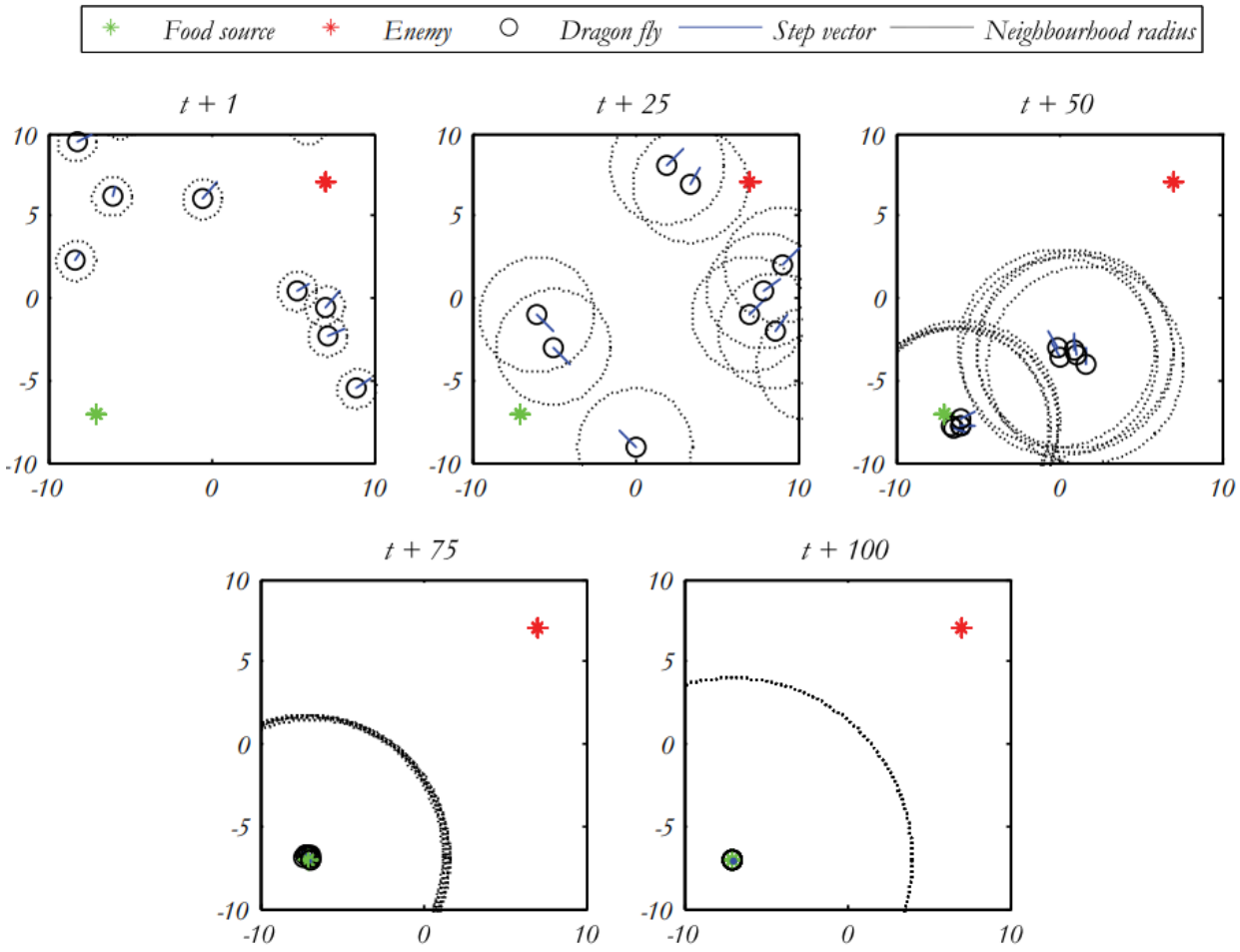


Figure 4: Simple reformative patterns among individuals based on five main factors in a swarm

5. Relation between DA and Single User Recommender:

Sl.	DA	Single User recommender
1	A dragonfly	A client
2	Dragonfly's position	Restaurant recommended by the client
3	Food Sources	Good restaurant
4	Enemy	Bad Restaurant
5	Separation	No. of restaurants which is uniquely recommended by a client
6	Alignment	Average gap between consquitive recommendations by a client
7	Cohesion	No of common recommendations between M recommendations of all potential neighbour and client
8	Attraction	Top n recommendation neighbor size given a client to target client
9	Distraction	n recommendation neighbor size given a client to target client
10	Step Vector	Total affinity Client towards the target client
11	Position Vector	Client's neighbor selection score

Table 2: Relation of DA and Restaurant recommendation

6. ACCF: Altered Client Based Collaborative Filtering (single user):

One of the most effective and well-liked recommendation algorithms among the different filtering strategies used in recommendation systems is collaborative filtering (CF). It has demonstrated success in a variety of recommendation systems, including as those for single clients, groups, and context-aware scenarios. A target client is given recommendations by CF based on the preferences of neighbours who share similar tastes. The objective is to recommend pertinent goods to the target client as recommendations. Choosing your neighbours is an important stage because the wrong neighbours can produce recommendations that are irrelevant. Neighbourhood selection is frequently carried out using common similarity coefficients, which incorporate previous ratings as an input. Examples of these are the Pearson correlation coefficient and the Cosine similarity coefficient. "Altered Client-based Collaborative Filtering" (ACCF), a novel neighbour selection technique designed exclusively for single user recommendation systems, is proposed to alleviate the CF algorithm's complexity. Swarm intelligence and statistical methods are used in the proposed algorithm to achieve optimal neighbour selection, with the application domain of restaurant suggestion being the main focus. With regards to managing client personalities, sequence suggestion, and satisfaction modelling, the adaptable architecture of a single user recommendation system frequently has serious flaws. A basic set-based client classification is suggested as a solution to these problems. By managing client personalities, handling redundant clients and attributes skillfully, and improving the recommendation process, this method seeks to address a number of issues. CF is an effective recommendation algorithm overall, and the proposed ACCF method and promising ways to strengthen neighbour selection and overcome issues with single user recommendation systems are provided by rough set-based classification.

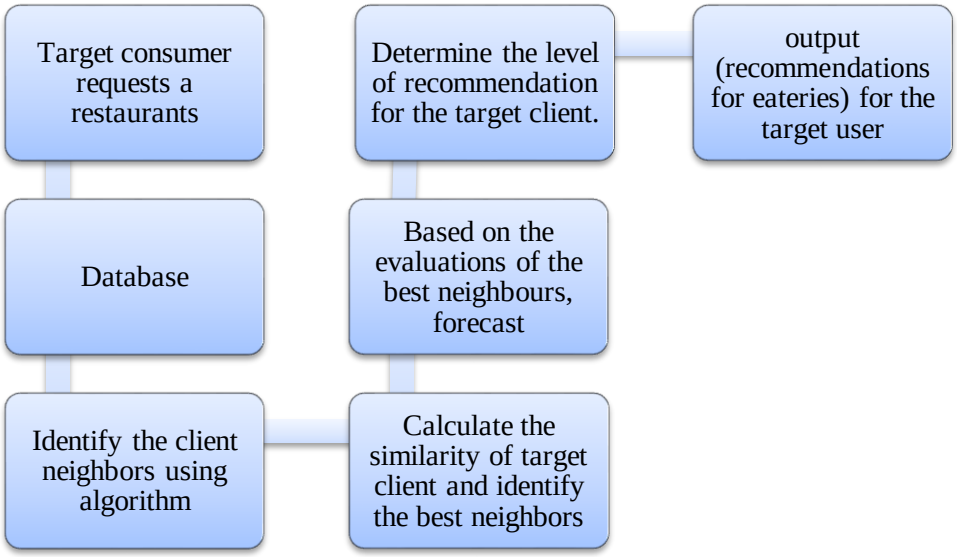


Figure 5: Block Diagram of ACCF

7. Methodology:

The goal of the ACCF algorithm is to categorise clients according to their characteristics and find the best neighbours to recommend. The Personality-based Client Classification Algorithm (PCCA) and the Optimal Neighbour Selection Algorithm (ONSA) are the two basic algorithms that make up ACCF.

7.1. ONSA:

One of the algorithms used by ACCF is ONSA. ONSA adheres to both the group preference model and the profile aggregation model. It operates in a number of steps. ONSA generally entails the following actions:

(1) Neighbors Identification,

(2) Best neighbors identification,

(3) Prediction of Rating,

(4) Generation of recommendation

7.1.1. Identification of the neighbors:

Identifying neighbours is the first stage in ONSA, and it is essential for tackling the sparsity issue and lowering the number of clients required for similarity computation. Because of this, the quantity of comparisons necessary to choose the best neighbour is significantly reduced. During the optimisation stage, it is made sure that the chosen neighbours are appropriate for the similarity computation. For the purpose of locating these neighbours, ONSA uses the Dragonfly Algorithm (DA). By using operators for separation, attraction, and distraction, the artificial dragonflies' intelligent behaviour efficiently minimises sparsity and maximises system coverage. Additionally, DA takes into account the ratings of all dragonflies simultaneously and carefully evaluates their affinity towards the target client without the necessity for reciprocal comparisons. The method used by ONSA to determine who your neighbours are in the contrary to UBCF's (User-Based Collaborative Filtering) methods. It focuses on neighbours with the greatest dissimilarity from the target client by explicitly analysing the evaluations of neighbours who have not been assessed by the target client. Additionally, the ratings are divided into two groups: higher ratings, which are considered recommendations, and lower ratings, which are considered non-recommendations. In the beginning, a connection is made between the DA parameters and the user-specific recommendation process.

When identifying neighbours, the attraction and distraction operator is crucial. Equations that can be used to determine these operators' values are provided below:

$$F_{n_t} = \begin{cases} tr_{n_t}{}^{g_t} & \text{if } tr_{n_t}{}^{g_t} \leq nar \\ 0 & \text{otherwise} \end{cases}$$

$$H_{n_t} = \begin{cases} tnr_{n_t}{}^{g_t} & \text{if } tnr_{n_t}{}^{g_t} \geq 1 \\ 0 & \text{otherwise} \end{cases}$$

Where,

g_t	A target client
n_t	A non-target client
$tr_{n_t}{}^{g_t}$	Number of recommendations provided by n_t t g_t
	Number of allowable top-n recommendations
$tnr_{n_t}{}^{g_t}$	Number of minimal ratings (i.e. ratings less than the minimum allowable rating) provided by n_t to g_t

The aforementioned equation has been modified to track a client's affinity for the target client. Because all operators are equally crucial in figuring out Zntq+1, they are all given the same weight in this improved form. Similarly, the iterations were given the same weights because restaurants were given the same priority. The following equation can be used to mathematically calculate the expression for the modified step vector with regard to nt (Zntq+1):

$$\Delta Z_{n_{t_{q+1}}} = w_1 \times (S_{n_t} + A_{n_t} + F_{n_t} + H_{n_t}) + w_2 \times \Delta Z_{n_{t_q}}$$

Where, w1 = w2 and 1= w1 + w2

Below is a presentation of the objective function for locating neighbours. In order to find - neighbours with a greater neighbour selection score, taking into consideration q iterations, it takes into account L clients, denoted as (nt, t = 1, 2,..., L).

$$MaximizeNSS_t = \sum_{o=1}^{q} Z_{n_{t_{q+1}}}$$

Where,

Subject to:

(i) $(\omega_{nt}{}^{gt} - \vartheta_{gt}{}^{nt}) \geq 1$

(ii) $\rho_{gt}{}^{nt} > 0$

(iii) $A_{nt} \leq GT$

Feasible Solution: [nt]

7.1.2. Identification of the best Neighbors :

Finding the best neighbours happens during the second stage of ONSA. Using the target client's rating similarity, a best neighbour is selected. The rating difference between the target client (gt) and the neighbour (nt) is calculated for the k-th restaurant to determine how comparable it is.

$$I_k(g_t, n_t) = \frac{d - \left| r_k^{g_t} - r_k^{n_t} \right|}{d}$$

The total similarity of the target client (gt) and the neighbour (nt) is then determined. Sim(gt, nt) indicates the overall similarity. which is evaluated using:

$$sim(g_t, n_t) = \frac{\sum_{k=1}^{\delta} w_k(g_t, n_t) \times I_k(g_t, n_t)}{\sum_{k=1}^{\delta} w_k(g_t, n_t)}$$

For Numeric Value:

$$W_k(g_t, n_t) = \begin{cases} 0 & \text{, k-th restaurant not co-rated by gt and nt} \\ 1 & \text{, otherwise} \end{cases}$$

7.1.3. Prediction of Rating:

The third stage of the Optimal Neighbour Selection Algorithm (ONSA) is the rating prediction stage. In this stage, the algorithm forecasts the evaluations that the intended customer would make of unrated establishments. It makes judgements about whether a new restaurant would appeal to the target clientele by acting as an expert system. The best neighbours' ratings are taken into account when calculating the anticipated rating. The algorithm calculates the prospective rating that the target customer would give a particular restaurant by using the ratings offered by these neighbours. This prediction technique is essential for creating tailored recommendations that fit the target client's interests. which indicates:

$$\mathrm{r}_{\mathrm{k}}'^{\,g_t} = \left[\left(\mathrm{r}_{\mathrm{k-1}}^{\,g_t} - \eta_{\mathrm{k-1}}^{\,\chi} \right)^2 \times \eta_{\mathrm{k}}^{\,\chi} \times \Delta\Gamma \right] \quad \text{where, } \Delta\Gamma = \left(\frac{\eta_{\mathrm{k}}^{\,\chi}}{\eta_{\mathrm{k-1}}^{\,\chi}} \right)$$

7.1.4. Generating recommendation :

The Optimal Neighbour Selection Algorithm (ONSA) ends with the phase of recommendation creation. The system decides whether to suggest a new restaurant to the target customer during this phase. This choice is made by contrasting the restaurant's expected rating with a predetermined threshold. The following inequality establishes the requirement for recommendation generation:

$$\left(r'_k{}^{g_t} - V_\gamma\right) \geq -1$$

In the given context, the variables have the following meanings:

"rk-1gt" denotes the rating given by customer "gt" to the (k-1)th restaurant; "k-1" denotes the standard deviation of all customer "nt" ratings to the (k-1)th restaurant, including the customer "gt" rating; "k" denotes all customer "nt" ratings to the k-th restaurant, excluding the customer "gt" rating; and "V" denotes the threshold value for recommendations.

Finally, four categories—very recommended, recommended, not recommended, and highly not recommended—are applied to the recommendations produced by the Optimal Neighbour Selection Algorithm (ONSA). The target client's recommendations are more pertinent thanks to this additional classification. The ONSA algorithm, which chooses the best neighbours in a two-way manner, is described in the stages below: 1. Set up the data structures and algorithmic settings. 2. Use the Dragonfly Algorithm (DA) to locate "nearby" nodes using swarm intelligence. 3. Determine how similar the target client (gt) and the nearby clients are. 4. Arrange the neighbours in descending order depending on the similarity metric. 5. Decide which of the top k neighbours would be the ideal neighbours. 6. Use a new similarity metric to assess the -best neighbours even more. 7. Sort the neighbors 8. Generate recommendation.

7.2. ONSA algorithm:

- **Input:** Enter restaurant patron reviews.
- **Out put:** The ideal neighbours and referrals for the target client are the output.

- **Step 1:** Start
- **Step 2 :** Divide the target client's overall ratings (gt) into q equal halves.
- **Step 3:** Give each split a positive integer to reflect the iteration number**.**
- • **Step 4:** Calculate Zntq+1 for each iteration while taking into account each client nt.
- **Step 5:** Based on the value of the objective function, choose m-neighbors.
- **Step 6:** Determine the target client's and each neighbor's similarity score (Sim(gt, nt)).

- **Step 7:** If the neighbour (nt) is deemed to be the target client's (gt) best neighbour based on the similarity score (Sim(gt, nt)) between the two.
- **Step 8:** Based on the required decrease of, choose the best nt's.
- **Step 9:** Based on the chosen neighbours, forecast the gt rating.
- **Step 10:** Strongly advise the K-th restaurant to gt if the anticipated rating is above a particular threshold and the target customer has not yet given the K-th restaurant a rating.
- **Step 11:** Recommend the K-th restaurant to gt if the anticipated rating is over a specific threshold and the target customer has not yet given the K-th restaurant a review.
- **Step 12:** Do not propose the K-th restaurant to gt if the anticipated rating is below a specific level and the target customer has not given the K-th restaurant a review.
- **Step 13:** Do not highly recommend the k-th restaurant if the anticipated rating is below a specific threshold and the target customer has not given it a rating.
- **Step 14:** Stop

8. Experimental Results:

The efficiency of recommendation systems is evaluated based on the accuracy and precision of the recommendations provided. Accurate recommendations indicate a lower level of errors, which is measured as the difference between the predicted and actual ratings. Precision, on the other hand, refers to the ability of recommenders to offer relevant recommendations. Various standard metrics are commonly used to assess accuracy and precision, including Mean Absolute Error, Root Mean Squared Error, Precision, Pearson's product-moment correlation, Kendall's tau, Mean Average Half-Life Utility, Receiver Operating Characteristic, Jaccard coefficient, Tanimoto coefficient, and Binary Cosine. However, more powerful metrics such as Coverage, Recall, and F-measure are increasingly utilized to evaluate the relevance of recommendations. These metrics provide a broader perspective on the effectiveness of recommendation systems. In this article, the accuracy and precision of the system are tested using metrics like Coverage, Root Mean Squared Error, Precision, and F-Measure. Finally, the results are compared with the standard User-Based Collaborative Filtering (UBCF) algorithm to gauge the performance improvement achieved.

8.1. Performance metrics :

- Coverage:
  A metric called coverage measures how well a recommender system can produce recommendations for a variety of user-entity pairs. It gauges how well the system can provide recommendations for different people, places, or things. Simply said, coverage measures how well the recommender system can respond to a wide range of user preferences and interests by providing a diversified set of entities for recommendation. A

higher coverage score denotes a wider range of recommendations, suggesting that the system can make recommendations for more user-entity pairs.

$$\text{Coverage} = \frac{\tau}{\rho}$$

Where,

τ	Number of ratings predicted
ρ	Number of ratings tested

- RMSE:

A metric called Root Mean Squared Error (RMSE) is used to measure prediction error and assess how accurate recommendation systems are. A larger RMSE value denotes less accurate forecasts, whereas a lower one denotes more accurate predictions. The predicted ratings are compared to a test set of known ratings to determine the RMSE. RMSE emphasises greater errors more than mean absolute error, another widely used statistic. The following equation can be used to calculate the RMSE mathematically:

$$\text{RMSE} = \sqrt{\frac{1}{o}\sum_{\{\xi,\zeta\}}\left(\upsilon_{\xi,\zeta} - \phi_{\xi,\zeta}\right)^2}$$

Where,

$\upsilon_{\xi,\zeta}$	Predicted rating for user on item
$\phi_{\xi,\zeta}$	Actual rating for user on item

- Precision:

Precision is formally a metric that assesses the proportion of pertinent recommendations to the overall amount of suggestions. It gives a sign of how accurate and pertinent a recommender system's recommendations are. The Root Mean Squared Error (RMSE), another measurement of precision. The following equation can be used to calculate precision in terms of RMSE:

$$\text{Precision} = 1 - \frac{\text{RMSE}}{\text{Maximum Possible Error}}$$

- F- Score/ F- Measure:

F-Measure offers a thorough assessment of relevance by taking into account both precision and recall. Recall captures the coverage of pertinent recommendations, whereas precision represents the correctness of recommendations. F-Measure determines the harmonic mean between recall and precision to offer a fair assessment of relevance. Additionally, the metrics of coverage and

precision can be used to derive F-Measure. The following equation can be used to calculate the coverage and precision of F-Measure mathematically:

$$\text{F-Measure} = \frac{2 \times \text{Precision} \times \text{Coverage}}{\text{Precision} + \text{Coverage}}$$

8.2. Dataset:

Rafael Ponce Medelln and Juan Gabriel González Serna of Mexico's National Centre for Research and Technological Development provided a dataset that was used to validate the ACCF algorithm. Data from collaborative filtering plus data from contextual filtering make up the dataset. Information on users, items, and ratings may be found in the file "rating_final.csv," which houses the collaborative filtering data. The user reviews are further separated into reviews of the food and reviews of the service to reflect preferences for various restaurant characteristics. The ratings in the dataset, which range from 0 to 2, should be noted because they guarantee consistency across the data.

rating_final.csv

	A	B	C	D	E
	userID	**placeID**	**rating**	**food_rating**	**service_rating**
	Number	Number	Number	Number	Number
1	userID	placeID	rating	food_rating	service_rating
2	U1077	135085	2	2	2
3	U1077	135038	2	2	1
4	U1077	132825	2	2	2
5	U1077	135060	1	2	2
6	U1068	135104	1	1	2
7	U1068	132740	0	0	0
8	U1068	132663	1	1	1
9	U1068	132732	0	0	0
10	U1068	132630	1	1	1
11	U1067	132584	2	2	2
12	U1067	132733	1	1	1
13	U1067	132732	1	2	2
14	U1067	132630	1	0	1
15	U1067	135104	0	0	0

Table 3: Customer rating dataset of a restaurant

8.3. Analysis and Results:

The "rating final.csv" file was used in the experiments for this study. UserID, placeID, and rating were the attributes taken into consideration for the experiment. The criteria of the objective function were initially used to choose the first 20 clients. This strategy was used to simplify computation and quicken the validation procedure. Consequently, at the neighbour identification stage, 20% of the neighbours were removed. During the process of choosing the best neighbours, an additional 20% of the neighbours were eliminated. Additionally, in addition to non-zero ratings, the trivial rating of 0 was also taken into consideration. With these parameters, the Accuracy-Based Collaborative Filtering (ACCF) method and the User-Based Collaborative Filtering (UBCF) method focused on Particle Swarm Optimisation (PSO) were contrasted.

The following results were obtained during the data pre-processing stage, following the implementation of ACCF and UBCF in 'MATLAB'.

- The simulation results are shown in Figure 6, which contrasts the coverage provided by ACCF and UBCF when employing the neighborhood sizes of 3, 6, 9, and 12. The graph's Y-axis shows the percentage of coverage, and the X-axis shows the size of the neighborhood. A single preference model was applied to the experiments.
- The experimental findings show that as neighborhood sizes increase, the coverage percentage of ACCF considerably rises, reaching 22%, 27%, 36%, and 47%, respectively. However, after a few iterations, UBCF performs poorly, with a fixed coverage percentage of 30%. These results demonstrate how much better the ACCF performs in terms of coverage than the UBCF.

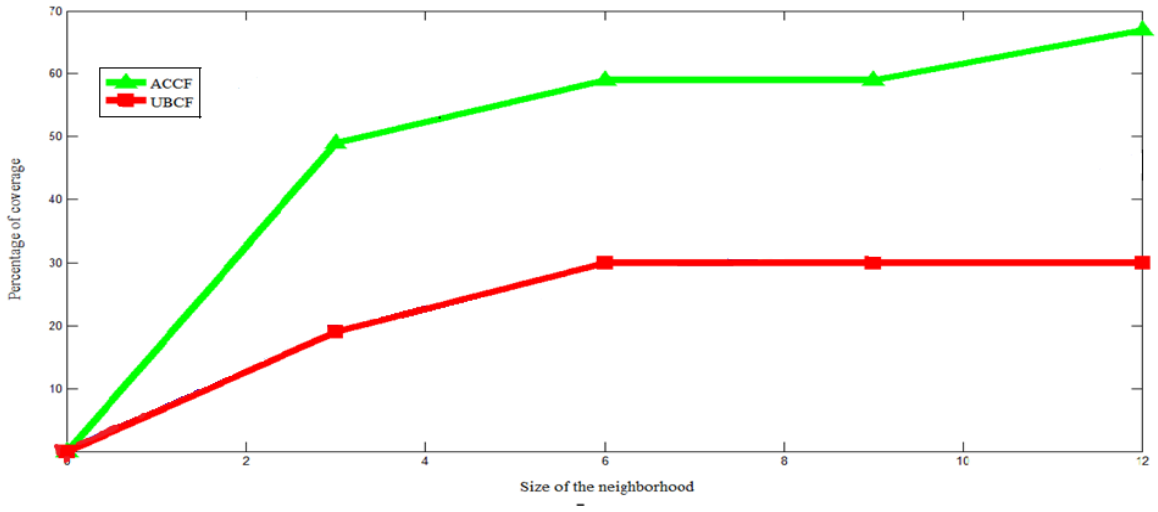


Figure 6: Coverage Percentage (According the neighborhood size)

- Figure 7 compares the RMSE for the projected ratings made using the ACCF and UBCF techniques, taking into account the varied test set sizes for restaurants, namely 2, 4, 6, and 8.
- • The graph's Y-axis displays the RMSE value, while the X-axis displays the size of the test set. The graph shows the scope of erroneous advice as well as the profitability of ACCF.
- The experimental results show that the RMSE of ACCF is greater (1.57, 1.34, and 1.24) compared to UBCF (0.70, 0.86, and 0.91) for test sizes 2, 4, and 6. The RMSE of ACCF, however, falls with test set size and, for a test set size of 8, it becomes smaller (1.11) than UBCF (1.23), despite the fact that UBCF is still larger (1.23).

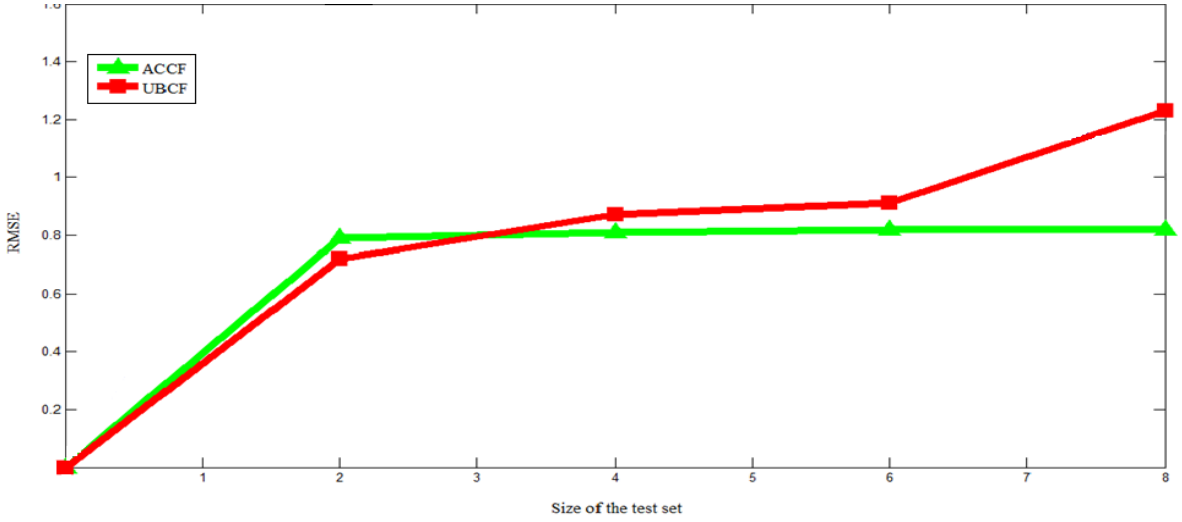


Fig 7: RMSE (According the testing set size)

- Figure 8 displays a comparison of precision between ACCF and UBCF, considering different sizes of the test set (2, 4, 6, and 8). The test set is created using the available ratings of the target client U1001 in the dataset.
- The X-axis of the graph represents the size of the test set, while the Y-axis represents the precision value. The figure illustrates the extent of personalized or relevant recommendations generated by the proposed algorithm.
- The experimental results indicate that in the initial stages, the precision of UBCF (0.27) is slightly higher than ACCF (0.21). However, the precision of UBCF decreases rapidly (0.13, 0.089, and 0) after a certain number of iterations, while the precision of ACCF increases monotonically (0.35, 0.40, and 0.45) with larger test sets.

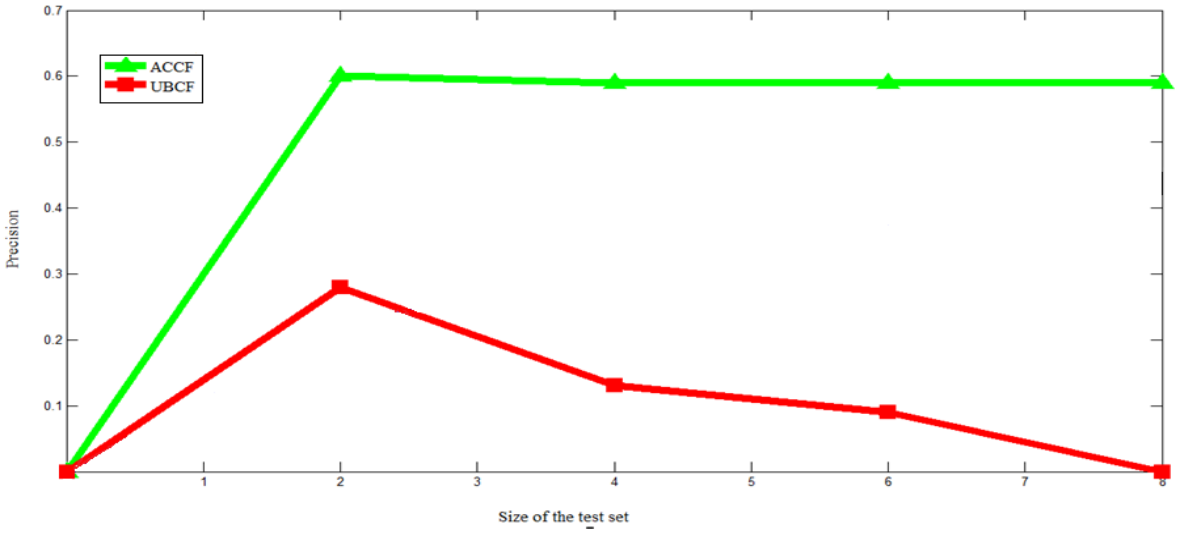


Fig 8: Precision (testing set size)

- Figure 8 shows a comparison of the precision between the ACCF and UBCF taking into account various test set sizes (2, 4, 6, and 8). The target client U1001's available ratings from the dataset are used to generate the test set.
- The graph's Y-axis displays the precision value, while the X-axis shows the size of the test set. The scale of personalized or pertinent recommendations produced by the suggested algorithm is shown in the figure.
- According to the experimental findings, UBCF (0.22) has a little greater starting F-score than ACCF (0.21) at this point. However, after a given number of repetitions, the UBCF's F-Score drops down quickly (0.13, 0.11, and 0), whereas the ACCF's F-score rises steadily (0.25, 0.29, and 0.27) as test sizes increase.

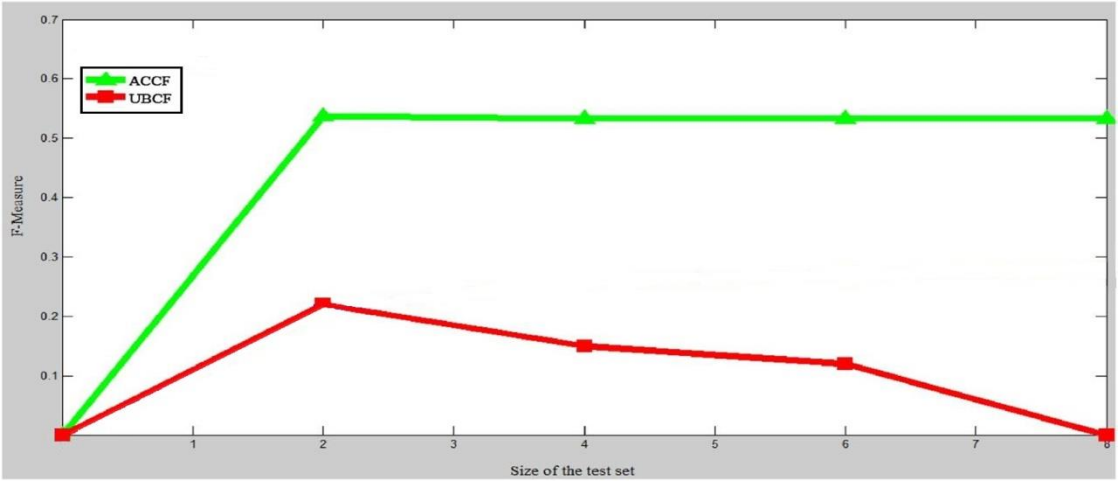


Figure 9: F- measure (according to testing set size)

9. Discussion of analysis:

In the setting of a single user recommendation system, the algorithm suggested in the preceding section displays its capacity to handle vast volumes of data. Using association rule mining, the rough set classifier is trained with flexibility in rough set estimation. Using MAX-MIN distance-based collaborative filtering (CF), it is possible to address different member preferences. By modifying the Dragonfly Algorithm (DA), the filtering procedure is further optimised. This lowers the number of neighbours needed for similarity comparison and resolves sparsity-related difficulties. A framework for single user recommendation that is quick to implement results from this optimisation. The proposed comparability measure differs from the conventional Pearson coefficient in that it bases user cooperation on ratings of similarity. When computing covariance, unlike the Pearson coefficient, which takes into account all prior evaluations, the suggested index only uses ratings from the present, avoiding the buildup of ratings from the past, which could result in inaccurate computations. By taking into account the ratings of the present item and its predecessor, the rating projection procedure is made simpler. This paradigm gives equal precedence to all clients, including best and nearby neighbours, based on how closely their preferences match. To enable more individualised recommendations, recommendations are divided into four categories: highly recommended, recommended, not recommended, and very not recommended. Overall, this method uses association rule mining, modified DA optimisation, machine learning approaches, and massive dataset management to produce a personalised and effective recommendation system.

10. Conclusion and Future Work:

An algorithm for a single user recommendation system is presented in the study work. In order to create groups, the algorithm uses pertinent features to remove redundant qualities from a huge number of characteristics. The article discusses the issue of preference aggregation and suggests a solution. Affinity and Neighbor Selection Score are combined to create a neighbour selection algorithm that finds the ideal neighbors for the target client. The Restaurant and Consumer dataset is used as a test bed to analysis the proposed algorithm and show how effective it is when compared to other collaborative filtering algorithms.

It is advised that future study test the usefulness of the given method using different information indices. A reliable similarity computation mechanism, a versatile search space reduction method, and the application of techniques for common optimization can be investigated. Beyond only one user-based recommender system, the performance of the suggested method can be assessed in a variety of other applications. Alternative recommender systems like the Particle Swarm Algorithm, Termite Swarm Algorithm, and Ant Colony Algorithm can be taken into consideration when the algorithm is expanded to serve a group of customers.

11. References:

1. Lü, L., Medo, M., Yeung, C.H., Zhang, Y.C., Zhang, Z.K. and Zhou, T., 2012. Recommender systems. *Physics reports*, *519*(1), pp.1-49.
2. Resnick, Paul, and Hal R. Varian. "Recommender systems." *Communications of the ACM* 40.3 (1997): 56-58.
3. Melville, Prem, and Vikas Sindhwani. "Recommender systems." *Encyclopedia of machine learning* 1 (2010): 829-838.
4. Ricci, Francesco, Lior Rokach, and Bracha Shapira. "Recommender systems: introduction and challenges." *Recommender systems handbook* (2015): 1-34.
5. Rahman, Chnoor M., and Tarik A. Rashid. "Dragonfly algorithm and its applications in applied science survey." *Computational Intelligence and Neuroscience* 2019 (2019).
6. Meraihi, Yassine, et al. "Dragonfly algorithm: a comprehensive review and applications." *Neural Computing and Applications* 32 (2020): 16625-16646.
7. Mirjalili, Seyedali. "Dragonfly algorithm: a new meta-heuristic optimization technique for solving single-objective, discrete, and multi-objective problems." *Neural computing and applications* 27 (2016): 1053-1073.
8. Shaikh, Mr Salim G., B. Suresh Kumar, and Geetika Narang. "Development of optimized ensemble classifier for dengue fever prediction and recommendation system." *Biomedical Signal Processing and Control* 85 (2023): 104809.
9. Schafer, J. Ben, et al. "Collaborative filtering recommender systems." *The adaptive web: methods and strategies of web personalization* (2007): 291-324.
10. Herlocker, Jonathan L., Joseph A. Konstan, Loren G. Terveen, and John T. Riedl. "Evaluating collaborative filtering recommender systems." *ACM Transactions on Information Systems (TOIS)* 22, no. 1 (2004): 5-53.