FacetDRS: Multi-faceted Feature Matching for Personalized Product Recommendations in Dynamic E-Commerce

Abstract: Dynamic Product Recommender systems play a crucial role in e-commerce, guiding users towards products that align with their preferences and enhancing the shopping experience. However, existing systems often struggle to capture the nuances of user preferences, particularly when dealing with multiple features via keyword queries and filters. Additionally, scalability challenges arise as the volume of products grows with features and values. In addition, the static filters of existing recommender systems limit the exploration by strictly adhering to user selections, hindering discovery of potentially better features within the preferred budget or category. This can lead to user dissatisfaction and missed opportunities to discover more desirable options. This research addresses these limitations by introducing FacetDRS, a novel dynamic product recommendation system that leverages flexible feature matching and a scalable architecture. FacetDRS empowers the recommendations with three distinct matching methods to cater to diverse preference nuances. Exact Match ensures precise adherence to user specifications, while High-Impact Partial Match prioritizes core features with flexibility in non-critical aspects. Extended Partial Match suggests relevant options beyond initial preferences, leveraging predefined thresholds. This multi-faceted approach captures the full spectrum of user desires while maintaining accuracy. To evaluate the effectiveness of FacetDRS, a rigorous experimental setup was conducted, comparing its performance across Keyword Match Score, First Selections in Top-K and Average Response Time metrics with established recommender systems such as DeepRec, APGNN, and TADCF. The results demonstrate that FacetDRS achieves superior performance than its counterparts in all comparison aspects. The system's flexible matching methods, combined with its robust architecture, offer a valuable solution for enhancing user satisfaction and driving conversions in the mobile commerce landscape.

Keywords: FacetDRS, Flexible feature matching, User preferences, Dynamic Recommender Systems, and E-commerce Infrastructure

I. INTRODUCTION

The product recommender systems [1] based e-commerce landscape is booming, driven by the convenience and accessibility of shopping on the go. But to truly thrive, this landscape requires personalized recommendations [2] that guide users towards products that resonate with their unique desires and needs. Recommender systems play a critical role in this equation, analyzing user data and suggesting relevant items that enhance the shopping experience and drive conversions.

However mobile commerce blooms on convenience and accessibility, personalized product recommendations remain stagnant, failing to adapt to the dynamic and nuanced preferences of users. Existing systems [3,4 and 5] struggle with inflexible feature matching, restricting users to rigid filters that miss subtle variations in desired features. This often leads to dissatisfaction [6, and 7] and missed opportunities. Users are willing to pay slightly more for better features (i.e. 5G network than 4G, Octa core than Quad core etc.) than their expectations. Current systems often miss these nuances, leading to recommendations that feel impersonal and irrelevant. Furthermore, scalability challenges [5, 6 and 7] arise as the volume and variety of products explodes, overwhelming existing systems and hindering performance. These limitations pose a significant challenge, in achieving the user satisfaction and conversion rates are:

- Existing systems primarily rely on rigid static filters [8 and 9] that categorize products based on specific feature values. This approach fails to capture a user's willingness to compromise on certain features (e.g., price) for more desirable ones (e.g., higher performance).
- Limited personalization and prioritization [2 and 10] in product recommendation due to lack of multi-dimensional processing in product selection
Improper dataset management [12] in existing systems struggle to handle the increasing data volume and computational demands, resulting in slow response times and compromised performance.

To address these limitations, we introduce FacetDRS, a novel dynamic product recommendation system that empowers users with flexible feature matching and a scalable architecture. FacetDRS revolutionizes the online shopping experience by offering a personalized and dynamic approach to product recommendations. By addressing the limitations of existing systems, FacetDRS allows users to discover products that truly resonate with their unique needs and desires, ultimately leading to increased satisfaction and conversion rates. To manage the ever-growing data needs, FacetDRS proposed the light weight data management approaches are Rich Feature Sets and Global Dataset Indexing and Hashing. FacetDRS proposed multi-facet feature selection model employs Exact Match, High-Impact Partial Match, and Extended Partial Match algorithms to intelligently select products, ensuring a seamless and efficient process. Using multi-dimensional processing and prioritization techniques, FacetDRS personalizes recommendations based on various user data points, not just purchase history. This enriched understanding leads to more relevant and desirable suggestions, fostering trust and engagement. The major contributions of this research are:

- Introduced the innovative FacetDRS framework, a dynamic and personalized product recommendation system that revolutionizes the conventional approach. It replaces static filtering techniques [3, 5] with flexible dynamic product selection algorithms to find the user interests by efficiently utilizing the input features from keyword queries and filters.
- Developed and implemented multi-facet product selection techniques within FacetDRS. Introduced Exact Match, High-Impact Partial Match, and Extended Partial Match algorithms, enabling users to receive recommendations based on precise matches, enhanced features, and flexible preferences.
- Propose a robust global dataset indexing and hashing mechanism within FacetDRS to efficiently handle vast and evolving e-commerce datasets. This approach accelerates the product selection process [13], ensuring users can quickly find optimal matches without the limitations of traditional search methods.

- Conducted a rigorous set of experiments using the FacetDRS prototype on Amazon Product Dataset [14], evaluating its efficiency and scalability Keyword Match Score, First Selections in Top-K and Average Response Time. Compared the results with existing recommendation systems such as DeepRec [15], APGNN [16], and TADCF [17], providing insights into the superior performance of FacetDRS.

Our FacetDRS contributions collectively advance the field of product recommendation systems, offering a comprehensive solution that addresses the limitations of existing models. The FacetDRS framework introduces novel techniques and algorithms, promising a more flexible, efficient, and personalized user experience in the realm of e-commerce.

II. LITERATURE REVIEW

The exponential growth of e-commerce has revolutionized how we shop, offering unparalleled convenience and access to a vast array of products. However, navigating this vastness can be overwhelming for users, leading to decision fatigue, and missed opportunities. To bridge this gap, product recommendation systems (RS) emerged as intelligent assistants, guiding users towards products that resonate with their unique needs and preferences. Effective RSs go beyond static suggestions, leveraging data and machine learning to curate personalized recommendations. This personalization is paramount for user satisfaction, ensuring users discover relevant products, save time and effort, and ultimately complete satisfying purchases. Furthermore, personalized recommendations translate into business success by driving conversions, increasing customer engagement, and fostering brand loyalty.

Early RSs navigated the e-commerce seas with two main anchors: collaborative filtering (CF) and content-based filtering (CBF). CF relied on user behavior (e.g., purchases, ratings) to find users with similar tastes and recommend similar products. CBF matched user profiles (e.g., demographics, interests) to product features. However, these approaches struggled with several intrinsic challenges.

Traditional RSs employed rigid filters based on past behavior or product features. Srifi et al. (2020) highlight that collaborative filtering using review logs face limitations due to cold start problems, hindering recommendations.
for new users or products. Alhijawi and Kilani (2020) proposed genetic algorithm relies solely on user-item interaction data, it might struggle to capture nuanced user preferences or product features beyond simple interactions [18, 19]. This could lead to inflexible recommendations based on limited information. Naumov et al. (2019) highlight that DLRMs can be susceptible to overfitting, leading to inflexible recommendations based on past behavior. Bobadilla et al. (2020) proposed RSs primary focus, relies on user-item interactions and may not effectively capture the dynamic and multifaceted nature of user preferences [2, 20]. This can limit the ability to explore new options and adapt to users' evolving preferences. While using an expert system and collaborative filtering, Walek & Fojtik's et al. (2020) movie recommender might suffer from static/limited recommendations due to rigid feature sets and potential cold start issues, hindering user satisfaction and exploration [21]. The limitations found in collaborative and content based former research works [18-21] restricts the RSs ability to explore options beyond exact matches, hindering their flexibility and potentially missing products they might love with slight adjustments (e.g., higher price, better features). These traditional RSs often treated user preferences as static, failing to adapt to their changing needs and priorities over time. This resulted in outdated recommendations, leaving users feeling disconnected and dissatisfied.

These limitations highlight the need for more dynamic and flexible RSs that can adapt to the ever-evolving needs of users in the vast ocean of e-commerce. Gong et al. (2012) [22] propose a flexible e-commerce system adapting recommendation methods on-demand (collaborative, content-based, etc.). While offering flexibility, this approach might increase complexity, potentially hindering system management and user understanding. Alphy et al. (2015) [23] introduces a dynamic recommender system, utilizing swarm intelligence inspired by bluegill fish behavior to tackle challenges like information overload and dynamic user behavior, outperforming traditional collaborative filtering in precision. This system may pose challenges due to the complexity introduced by swarm intelligence while providing diverse, customized recommendations from ever growing data records. Annam et al. [24] introduced a hybrid recommendation system, applying fuzzy and Analytic Hierarchy Process (AHP), to addresses common issues like low accuracy and lack of personalization. Complexity of the hybrid algorithms may require careful consideration of computational resources and may pose challenges in balancing accuracy and computational efficiency. The major limitations found in former Dynamic Product Recommender Systems are: Static Feature Matching, Limited Personalization and Prioritization, Scalability and Performance Challenges. Today, personalization techniques tailor recommendations to individual user preferences, leading to a more satisfying and engaging experience. Goldenberg et al (2021) paper delves into real-world applications of personalization methods, providing insights into their practical implementation [25, 26]. Their hybrid fuzzy mechanisms contain limitations in finding the products from vast dataset due to the product selection algorithms complexity. Ge et al. (2020), introduces a novel risk-aware recommendation framework that integrates machine learning and behavioral economics, addressing the limitations of existing recommendation systems by incorporating users' risk attitudes. The practical implementation and interpretability of such a framework might pose challenges, particularly for users unfamiliar with the underlying behavioral economic principles [26].

**Limitations Identified:** Traditional RSs rely on static filters [8, 9, 16 and 18] based on past behavior or product features, hindering flexibility and exploration. This limits their ability to adapt to user preferences as they evolve and explore new options. Existing algorithms [9 and 18] often struggle with matching user preferences to product features effectively. Additionally, managing and processing vast datasets can pose scalability challenges [13 and 27], impacting accuracy and recommendation efficiency. Most systems fail to consider the broader context of user needs, such as time, location, or intent. This can lead to irrelevant recommendations that miss the mark. The identified gaps and limitations in existing literature [8, 9, 18 and 13] underscore the need for a more dynamic and flexible recommendation system. Rigid filters, limited personalization, and challenges in feature matching algorithms and dataset management have prompted the exploration of innovative solutions like FacetDRS. In this paper, out FacetDRS addresses the identified challenges in existing recommendation systems. By incorporating multi-facet product selection and dynamic dataset indexing, FacetDRS aims to offer a more adaptive, personalized, and efficient recommendation experience, overcoming the limitations observed in traditional models.

III. DATASET MODELING

This paper proposed FacetDRS experiments delves into the domain of mobile phone preferences using a comprehensive dataset obtained from Amazon's product offerings in 2022 [14]. Encompassing a diverse range of
1000 mobile devices, this structured dataset, presented in a table format within a (.csv) file, provides valuable insights into the features that shape consumer choices within the ever-evolving mobile landscape. Each record precisely captures key attributes that influence buying decisions, including model, price, user ratings, SIM type, processor, RAM, battery capacity, display technology and size, camera specifications, memory card support, and operating system. By precisely analyzing these features and their intricate relationships, we aim to unveil valuable insights into:

- Evolving consumer preferences: Identifying the features that hold the most sway in purchase decisions, revealing shifts in user priorities over time.
- Price-performance sweet spots: Exploring the optimal balance between affordability and desired features, guiding buyers in best product selection.
- Emerging trends: Uncovering novel features or combinations gaining traction, anticipating future purchase directions.

This rich and granular Amazon product dataset presents a unique opportunity to delve into the intricacies of consumer behavior within the product shopping and recommendation.

3.1 Knowledge Base Preparation

To ensure the quality of recommendations generated by our multifaceted dynamic product recommender system (FacetDRS), we first board on a careful data preprocessing for cleaning and preparation process to generate the flexible knowledge base [28] from dataset. This initial stage serves as the foundation for accurate feature extraction, categorization, and ultimately, the delivery of personalized product suggestions.

*Ensuring Data Consistency and Normalization:* We precisely scan product names, descriptions, and feature values to identify and rectify errors, ensuring consistency and accuracy in our data. To facilitate seamless analysis, text data is converted to lowercase [29], eliminate punctuation and special characters, and divide text into individual words or phrases (tokenization).

*Unifying Feature Representation:* Our preprocessing approach tackle inconsistencies in feature naming and representation, such as synonyms, abbreviations, and typos, to ensure accurate feature identification and extraction across the dataset. This involves techniques like stemming or lemmatization to reduce words [29] to their root forms, as well as the creation of feature dictionaries or ontologies to establish a consistent vocabulary. By diligently addressing these aspects of data cleaning and preparation, we establish a cohesive and reliable foundation for subsequent feature categorization and ranking.

3.2 Feature Categorization and Ranking

To empower our FacetDRS with a profound understanding of product nuances, we carefully categorize and rank features, orchestrating a semi-automatic approach that blends computational efficiency with human expertise.

*Categorization:* We establish a well-structured hierarchy of features, mirroring the complexities of product characteristics. This model encompasses primary categories, which are further branching into descriptive subcategories like Display (Screen Type, Resolution, Refresh Rate), Camera (Resolution, Sensor Size, Optical Zoom), Processor (Brand, Model, Speed, Cores), Memory (RAM, Storage Type, Capacity), Battery (Capacity, Charging Speed) and Operating System (Version, Update Availability) etc.

For categorization, a set of regular expressions [29 and 30] and text mining techniques are employed on dataset to precisely extract relevant features from product descriptions to create a smart knowledge base. This process involves identifying keywords, patterns, and numerical values that signify specific product attributes. To enhance the accuracy in categorization, the domain knowledge and feature dictionaries [31] are incorporated, which ensure consistent identification and interpretation of features across the dataset.

For well-defined categories with clear boundaries, we employ rule-based classification techniques [27 and 32], automating the process of assigning features to their appropriate categories. This category contains features like Display Screen Type (LCD, OLED, AMOLED, Super AMOLED), Camera resolution (10, 50, 100 MP) etc. Technical specifications and numerical values can be directly extracted from the data and used to rank features within these categories.
To extract the features from structured data with well-defined patterns, regular expressions [27] are employed. Here are some sample regular expressions that can be used to extract relevant features (i.e., Screen size, Screen Type and Storage) from product descriptions:

\[ \text{Screen Size: } r^n b \left( \text{d} \% \text{otF} 0\text{2B} \left( ? \text{ d} s \right) \text{ sinch(} \text{e}) \right)^n \]

\[ \text{Screen Type: } r^n b \left( \text{LCD} | \text{OLED} | \text{AMOLED} | \text{Super AMOLED} \right) \]

\[ \text{Storage: } r^n b \left( \text{d} \% \text{uF} 0\text{2B} \right) s \text{GB} s \left( \text{storage|ROM} \right) \]

For ambiguous cases or subjective features that require a deeper understanding of context and sentiment, we integrate human intervention. In example, while brand names are distinct, their perceived value and reputation can be subjective and vary depending on user preferences and market trends. To extract the features from ambiguous descriptive text, the text mining features [28 and 31] are applied. At first the text description is tokenized for splitting text into words or phrases, later the stemming, part-of-speech tagging and named entity recognition techniques [29] are applied over the tokens followed by applying TF-IDF [33] to identify important keywords and phrases.

This approach ensures a comprehensive categorization that captures both objective and subjective aspects of product attributes. Finally, we leverage domain expertise and dataset analyst to ensure the relevance and comprehensiveness of these categories and subcategories, capturing the essential nuances that differentiate products is shown in figure-1.

![Figure-1. Mobile Products data Feature Categorization and Ranking](image)

**Ranking Features**: To ensure our FacetDRS to understand the true value of each feature, we eventually rank them within their categories, incorporating both objective technical prowess and subjective user perspectives. For Objective Criteria, we utilize technical specifications, such as screen resolution, processor speed, and battery capacity, to objectively rank features within their respective categories. This creates a clear hierarchy based on measurable performance indicators. For example, a higher resolution (e.g., 4K vs. Full HD) offers sharper visuals and greater detail, ranking it higher in the Display category.

For Subjective Perspectives, we integrate expert ratings and user reviews, shedding light on subjective aspects of product features that may not be fully captured by technical specifications alone. This integration captures user experiences and preferences, providing a more holistic view of product quality. For example, a phone with a popular brand might rank higher based on user ratings compared to others. This balanced approach, considering both objective technical merits and subjective user perspectives, allows us to build a nuanced and comprehensive ranking system that reflects the true value of each feature for different users.

IV. FACETDRS MODEL APPROACH

In the dynamic landscape of online product exploration, striking a balance between accuracy and efficiency in recommendation systems is paramount. To address this challenge, we propose a multifaceted model (Facet-DRS) that embraces knowledge-driven prioritization, semantic comprehension, and dynamic adaptation to deliver personalized recommendations. This model seamlessly utilizes the user input, knowledge base insights, and efficient feature matching algorithms to guide product discovery effectively.

4.1 User Input
The first step in our RSs journey understands the user's desires and aspirations. In our model the users enters the keywords and filters in single (first) step to shares their intentions and preferences. Keywords are the descriptive terms act as models, guiding the system towards products that embody the user's ideal features. Imagine a user searching for a "Best 5G mobiles under 20000." These keywords, acting as specific search directions in the vast amount of products from dataset, illuminate options that boast high-performance processors, budget under 20000, and features like 5G network.

Similarly the user specified filters [19 and 20] serve as rudders, allowing the user to refine their search and steer clear of unwanted options. Imagine the user setting a price range or specifying a preferred brand. These filters act as navigational constraints, ensuring that the recommended products stay within the user's desired course and budget. By combining these two inputs, the user specifies a vivid picture of their ideal product, empowering the system to navigate the vast ocean of possibilities and deliver an expecting and relevant recommendation.

4.2 Rich Feature Set Tree Generation

The journey towards optimal product selection starts with a deep understanding of user needs and preferences. This section unveils the "Rich Feature Set Tree Generation" algorithm, a powerful tool that unlocks the knowledge base and extracts the most relevant features, tailored to each user's unique desires. By combining natural language processing [30], tree exploration, A* search [34] and rule-based filtering [32][37], this algorithm builds a comprehensive Rich Feature Set Tree, serving as the roadmap for personalized product recommendations. The below algorithm describes the user input into a feature-rich map that guides the path to perfect product matches.

4.2.1 Rich Feature Set Tree Generation Algorithm:

Input: User keywords (vecK) and filters (vecF), Knowledge tree (treeG) and Feature selection logic (nicFlog)

Output: Rich Feature Set Tree (treeRFS)

Begin

Preprocessing

- Tokenize user keywords vecK and filters vecF using Word2Vec.
- Match keywords vecK and filters vecF to corresponding nodes in the knowledge tree treeG based on labels and descriptions.
- Calculate the cosine similarity (cosθ) scores for each matched node based on keyword/filter match strength and feature rank.
- Apply similarity threshold (δ) to find the match nodes in treeG.

Ruled A* Search

- Initialize a priority queue (Q) with frontier nodes (f1…fn) where cos θ ≥ δ

  - Define the root of Rich Feature Set Tree (RFSroot)
  
  While Q(χ) != null and x ≤ len(Q)
  
  node = POP(Q(χ))
  
  matchNode = findFeatureMatch(node, Gtreevec)
  
  tmpMatchGraph = exploreDescendants(matchNode)
  
  if(f_node = getDescendant(tmpMatchGraph)) => featureNode
  
  addChild(RFSvec, f_node)
  
  Fvec = findFeatureSelectionLogic(Flogic, f_node)
  
  valGraph = exploreDescendants(f_node)
  
  While (c_node = getChild(valGraph)) != null
In this algorithm, at first user keywords and filters are broken down into word vectors \((K_{vec} and F_{vec})\) using Word2Vec [35], a word embedding technique that captures semantic relationships between words. The algorithm matches the vectorized keywords and filters to corresponding nodes in the knowledge tree based on their labels and descriptions. This establishes an initial connection between user input and relevant features in the knowledge base. To compare how closely a product description matches a user's keywords \((K_{vec})\) and filters \((F_{vec})\), we use a mathematical measure called cosine similarity \((\cos \theta)\) [36].

\[
\cos \theta = \frac{A \cdot B}{\|A\| \|B\|}
\]

Here, \(A\) and \(B\) represent the numerical vectors of the two text pieces being compared (e.g., a user's input vectors \((K_{vec} or F_{vec})\) and a product description). \(A \cdot B\) denotes the dot product of vectors \(A\) and \(B\), which calculates the sum of the products [30] of their corresponding components and \(\|A\|\) and \(\|B\|\) represent the magnitudes (or lengths) of vectors \(A\) and \(B\), respectively. It's like calculating the angle between the vectors representing the words. The result of the cosine similarity \((\cos \theta)\) calculation ranges from -1 to 1, where value of 1 indicates perfect alignment between the vectors \(\angle 0^\circ\), suggesting strong semantic similarity and value of 0 signifies orthogonal vectors \(\angle 90^\circ\), implying no semantic relationship. Similarly value of -1 indicates oppositely directed vectors \(\angle 180^\circ\), suggesting antonym-like relationships, which means the smaller the angle, the more closely related the words are in meaning. This approach allows us to find products that might not have the exact same keywords as the user's search, but still capture the same ideas.

For each matched node, a cosine similarity score \((\cos \theta)\) [36] is calculated, measuring the degree of similarity between the user input \((K_{vec} and F_{vec})\) and the node's content. This score incorporates both keyword/filter vectors match strength and the feature's rank in the knowledge tree. A similarity threshold \((\delta)\) is applied to filter out nodes that don't meet a certain level of relevance to the user input. This ensures that only the most promising features are considered for further exploration.

Figure-2. Labeled Rich Feature Set Tree from User Input Mapping

To generate the labeled rich feature set tree from knowledge base, a custom heuristic method is defined as Ruled_A*Search method. In this method, a priority queue is created, containing the frontier nodes (matched nodes that passed the threshold) and their corresponding cosine similarity scores. This priority queue is used to guide the search process, focusing on the most relevant nodes first. The algorithm iteratively pops the node with the highest score from the priority queue and explores its neighboring nodes in the knowledge tree. This exploration uncovers potentially relevant features that might not have been directly matched in the preprocessing stage. If a neighbor
node is identified as a feature node (representing a specific product feature), it's added to the Rich Feature Set Tree ($RFS_{tree}$). This tree gradually accumulates the relevant features discovered during the search. The algorithm applies the feature selection logic ($F_{logic}$) to each identified feature node ($f_{node}$), ensuring that the extracted features align with user preferences and constraints. This logic might involve rules like "High/Equal Good (+/=)", "Less/Equal Good (−/=)", or "Equal is Good (=)", depending on the application's requirements. For each feature node, the algorithm explores its descendant nodes ($c_{node}$) in the knowledge tree to extract available values and rankings ($valRankMap$) for that feature. This information is essential for product selection and recommendation in later stages. The algorithm builds the Rich Feature Set Tree ($RFS_{tree}$) by adding edges between feature nodes and their associated value nodes, along with their corresponding rankings. This creates a structured representation of the relevant features, their possible values, and their importance, ready for further product selection processes.

The algorithm returns the root node of the generated Rich Feature Set Tree, which serves as a comprehensive summary of the extracted features, their values, and rankings, tailored to the user's input and preferences. This tree provides a valuable foundation for subsequent product selection and recommendation tasks.

4.3 Multi Facet Product Selection and Recommendation

At this stage, Facet-DRS process the user input vectors ($K_{vec}$ and $F_{vec}$), rich feature set tree ($RFS_{tree}$), and dataset ($D$) contextual factors for personalized recommendations. With a Rich Feature Set Tree in hand, the FacetDRS model embarks on a quest to uncover products that resonate with user desires. This section delves into the heart of product selection, where multifaceted techniques seamlessly merge with intelligent algorithms to reveal optimal choices. To witness the interplay of Exact Match, High-Impact Partial Match, and Extended Match with Thresholds, each facet meticulously sculpting a refined set of recommendations from a vast dataset.

4.3.1 Global Dataset Indexing and Hashing: The e-commerce world is constantly evolving, with new products and features emerging at a rapid pace. Traditional search methods often struggle to keep up, leaving users frustrated and unable to find what they're truly looking for. This is where FacetDRS's innovative global dataset indexing and hashing [37] come into play, transforming the system from a simple search engine into a powerful recommendation engine.

![Figure-3 Hash Model of Mobile features Categorization](image)

As shown in figure-3 and figure-4, FacetDRS leverages global dataset indexing and hashing ($I_{tree}$) to efficiently handle millions of products and their intricate combinations of features in a customized tree structure. This ensures that even in a vast digital marketplace, users can quickly and easily find the perfect match for their needs. Hashing acts as a fast-track to identify potential matches based on keywords and features, while indexing allows for precise comparisons of product attributes. This dynamic duo significantly reduces the time it takes to find relevant products, delivering a seamless and efficient experience for users.
At the heart of our approach lies the ability to maintain feature value ranking implicitly. We achieve this by arranging feature hash values in ascending order of rank. Additionally, to address data management complexity, we employ specialized KD trees [38] for each product type. These trees, indexed by product hash sets, efficiently handle all possible values of any feature, making it possible to index a large number of products of the same type with just a single KD tree. This model significantly accelerates feature comparison and product selection during the search process.

To retrieve selected product information at high speed, we map product hashset values (Prod_Hash_ID) with their actual IDs (Prod_Actual_ID) as shown in figure-5. These IDs reference the product object trees, which hold all the relevant feature information within the dataset. This innovative approach eliminates the need to access the actual dataset product information for each user query from secondary storage devices, significantly reducing processing time and improving user experience.

4.4 Product Matching Methods

With both user interests represented as the RFS_{tree} and the global dataset indexed I_{tree}, we are now ready for the final step: matching the user's needs against the dataset and retrieving relevant products using our three distinct matching methods: Exact Match, High-Impact Partial Match, and Extended Partial Match. Each method offers varying degrees of flexibility to cater to different user preferences and search intents.

4.4.1 Exact Match: The "Exact Match" method in FacetDRS aims to retrieve products from the global index and hash dataset I_{tree}, that precisely match the user's preferences expressed in RFS_{tree}. Here the RFS_{tree} works as a map guiding the search, with each branch representing a desired feature and its first-level value serving as the...
exact target. On the other hand, the global data tree $I_{tree}$ acts as a vast library, meticulously organized with indexed and hashed features and values for every product.

**Exact Match Algorithm**

Input: $RFS_{tree}$, $I_{tree}$ and $f_{logic}$

Output: Product Set ($P_{1_set}$)

Initialization:

$RFS_{root} = getRoot(RFS_{tree})$

$I_{root} = getRoot(I_{tree})$

$P_{set}$, $P_{id_set}$, $P_{imp} = Set()$

Exact Matching:

while ($(RFS_{feature} = getFeature(RFS_{root})) \neq null$)

$RFS_{val} = getFirstLe velVal(RFS_{feature})$

$RFS_{valRank} = getValRank(RFS_{feature}, RFS_{val})$

$I_{feature} = bitwiseMat ch(RFS_{feature}, I_{root})$

if ($I_{feature} \neq null$)

$I_{val} = getVal(I_{feature})$

$I_{valRank} = getValRank(I_{feature}, I_{rank})$

$F_{valMatch} = exactMatch(RFS_{val}, RFS_{valRank}, I_{val}, I_{valRank}, RFS_{feature}, f_{logic})$

if ($F_{valMatch} \geq \delta$)

$P_{id_set}.add(getProdIDs(I_{feature}, I_{val}))$

$P_{imp}.add(P_{id_set})$

continue

$P_{1_set} = setIntersection(P_{imp})$

return $P_{1_set}$

The exact match algorithm embarks on its journey by setting the stage: it grabs the starting points of both trees (the roots) and prepares empty sets to collect potential matches. Then, it begins traversing the features in the $RFS_{tree}$, one by one. For each feature, it extracts the specific value the user has specified. After that it starts
finding this feature within the global data tree ($I_{tree}$). The algorithm employs a meticulous bitwise comparison, ensuring an exact match for the feature itself. If a match is found, it's time to delve deeper into the values and their ranks. Here's where the algorithm goes beyond strict equality, embracing the concept of "better value" matches. It carefully scrutinizes both the user's specified feature ($RFS_{feature}$), value ($RFS_{val}$) and its rank ($RFS_{valRank}$) against the corresponding information $I_{feature}$, $I_{val}$ and $I_{valRank}$ in the global data tree. If the values align exactly, or if the product offers a "better" value based on $f_{logic}$, indicates the $F_{valMatch}$ is found and it is compared against the match probability threshold ($\delta$) to ensure the exact match. At this moment, the algorithm gleefully gathers the product IDs ($P_{ID\_Set}$) associated with this successful match and stores them in a temporary set ($P_{imp}$) dedicated to that specific feature.

After examining all features in the $RFS_{tree}$, the algorithm has collected temporary sets, each containing product IDs that matched a specific feature and its value criteria. But the job isn't finished yet. To ensure only products that match all user-specified features qualify, the algorithm performs a crucial set intersection operation. The products that reside in the areas where all the circles overlap are the ones that have successfully navigated all feature and value hurdles. These are the true products as a result set $P_{1\_set}$, which contains the product IDs - the fruits of its meticulous search through the feature landscape. They represent the products that perfectly align with the user's desires, offering precisely the features and values they seek.

### 4.4.2 High Impact Partial Match:

This matching method expands FacetDRS capabilities beyond the strict exact matches. It aims to identify the products that might not perfectly align with all user preferences but offer superior value in key features, potentially exceeding expectations. Imagine a user searching for a phone with 6GB RAM, 128GB ROM, and a 50MP camera, but the "High-Impact Partial Matching" knows that a phone with 8GB RAM, 256GB ROM, and a 48MP camera at the same price might be an even better deal for them.

#### High Impact Partial Match Algorithm

**Input:** $RFS_{tree}$, $I_{tree}$ and $f_{logic}$

**Output:** Product Set ($P_{2\_set}$)

**Initialization:**

$RFS_{root} = getRoot(RFS_{tree})$

$I_{root} = getRoot(I_{tree})$

$P_{set}, P_{ID\_Set}, F_{ID\_Set}, P_{imp} = Set[]$

**High Impact Partial Matching:**

$TMP_{tree} = reviseFeatureWeights(RFS_{root}, K_{vec}, F_{vec}, Z)$

$TMP_{tree} = weightBaselineFeatureSortASC(TM_{tree}, RFS_{root})$

$HI_{tree} = selectHighImpactFeatures(TM_{tree})$

while ($HI_{feature} = getFeature(HI_{tree}) \neq null$)

$I_{feature} = bitwiseMat(ch(HI_{feature}, I_{root}))$
The High Impact Partial Match algorithm precisely examines the user’s preferences $RFS_{tree}$, looking for keywords and filters that explicitly mention features. Based on how often and prominently these features appear in $K_{vec}$ and $F_{vec}$, the algorithm assigns and revises their weights, reflecting their potential impact. Based on these revised weights, the features are sorted in ascending order, which helps to select the high impact features in product selection. While selecting the high impact feature tree the domain knowledge ($Z$) is used in conjunction with user preferences ($RFS_{tree}$). After the high impact feature tree ($HI_{tree}$) is constructed, iteratively the features ($HI_{feature}$) are extracted and compared against the global dataset ($I_{tree}$) using bitwise matching, to find the global dataset matching features ($I_{valRank}$).

To perform the partial match with high impact feature values, the $HI_{feature}$ related values and ranks map list $HI_{valRankMapList}$ is compared with $I_{valRankMapList}$ using the $f_{logic}$. Instead of demanding exact matches, this algorithm employs fuzzy matching techniques along with the Jaccard similarity [24 and 27], which helps in finding similar feature values as list ($F_{valMatchList}$) in the global data tree by considering rank differences. Iteratively the matched values ($F_{valMatch}$) of all features are extracted and their matching probabilities are compared against the threshold ($\delta$) value. Product IDs are obtained from the feature value references are formed as a set ($P_{ID.Set}$) and finally all these sets are explored by the intersection process to find the final product set ($P2_{set}$) with all satisfied features.

4.4.3 Extended Partial Match: This matching process builds upon the foundation of high-impact partial matching. It recognizes that sometimes, slightly adjusting user preferences for flexible features can open doors to a wider range of products that might still align well with user’s overall needs. This match selects the best products.
which are waiting outside the boundaries of the flexible features (i.e. price, delivery time) just by leveraging its values based on the predefined extension thresholds. Domain experts provides the predefined features and thresholds at each category of products are selected for extended partial match.

For each predefined flexible feature, a separate threshold is defined to specify the acceptable range of deviation from the user’s initially specified value. For example, if the user's budget is 15000 rupees, a positive 5% threshold would translate to an extended range of 15750 rupees. After leveraging the flexible feature values, the leveraged features are set as the high impact features and the same are updated in user preferences \( (HI_{tree}) \). At this moment our model feed the updated \( HI_{tree} \) along with \( I_{tree} \) as inputs for the High Impact Partial Match algorithm to obtain the boundary level hidden best product objects are recommendation in the form of result set \( P^3_{set} \).

4.4.4 Result set De duplication and Integration: Our FacetDRS generates three result sets are: \( P^1_{set} \) for exact matches, \( P^2_{set} \) for high-impact partial matches, and \( P^3_{set} \) for extended partial matches. Each set prioritizes user preferences differently, with \( P^1_{set} \) being the most stringent and \( P^3_{set} \) the most flexible. However, this independence can lead to unwanted duplication, where the same product might appear in multiple result sets. To address this, a multi-level de-duplication process ensures distinct product recommendations across varying levels of match strictness. Based on the results sets relevance to user preferences, the \( P^1_{set} \) sets having the high priority followed by \( P^2_{set} \) hand \( P^3_{set} \). To eliminate the duplicates from result sets, the de-duplication process starts by comparing product IDs within \( P^1_{set} \) against those in \( P^2_{set} \) and \( P^3_{set} \). In case a product ID of \( P^1_{set} \) match is found in \( P^2_{set} \) and \( P^3_{set} \), the most stringent and the same are meticulously removed from them, to prevent the same product appearing in multiple lower-priority sets. Similarly, the \( P^2_{set} \) product ID matches are removed from \( P^3_{set} \) to ensure the result sets uniqueness. Finally the three result sets are integrated in same order from \( P^1_{set} \) to \( P^3_{set} \) generate the final result set \( R_{set} \), which will be refined based on purchase history and user profile information.

4.4.5 Personalization of Product Recommendations: Soon after the redundancy elimination and consolidation of three result sets into \( R_{set} \), it’s time to personalize the product recommendations [39] further. This personalization step aims to refine the existing weights of products within \( R_{set} \) based on the user's unique purchase history and profile information. Think of it as adding a layer of personal touch, guiding the user towards options that resonate more deeply with their preferences.

For this, we begin by delving into the user's purchase history, treating it like a treasure trove of past choices and preferences. We analyze the categories, brands, features, and price points of their previous purchases, identifying patterns and recurring themes, which helps to find their established preferences and comfort zones. Simultaneously, we delve into the user's profile information like age, gender, location, hobbies, and online behavior to understand their needs and priorities [40]. After gathering the profile and purchase history information, for each product, our model calculates a multiplicative factor based on its alignment with the user's purchase history and profile. This factor can be derived from various sources, such as the frequency of similar purchases [41], the presence of desired features, or the match between user demographics and product target audience.

Products demonstrating stronger alignment receive a higher factor, effectively amplifying their weight within the result set. This updated weight reflects the combined influence of both general product characteristics and the user's individual preferences. Now products with weights reflecting both product quality and personal relevance, it's time to showcase the final results. Sort the products in \( R_{set} \) based on their updated weights, ensuring that those most aligned with the user's unique preferences occupy the top positions. This personalized ranking ensures the user encounters the most relevant options first, saving them time and effort in navigating the product landscape.
V. EXPERIMENTAL ANALYSIS

This section delves into a meticulously designed experiment plan to rigorously assess the efficiency and scalability of FacetDRS against its counterparts.

5.1 FacetDRS Prototype

The development of the FacetDRS prototype involves a systematic approach using Java and associated technologies. Initial steps include environment setup with the latest JDK, Eclipse or IntelliJ IDEA as the IDE, and integration of libraries such as Apache Spark, H2O.ai, and Spring Boot. Data preparation encompasses defining a schema, loading datasets into a Spark DataFrame, and preprocessing to handle missing values and scale features. The model implementation incorporates rule-based exact matching, fuzzy matching for high-impact features, and extended matching based on user-defined thresholds. Features are weighted and ranked considering user profiles and purchase history. The prototype development utilizes Spark MLlib for machine learning tasks, H2O.ai for complex algorithms, and Spring Boot for creating a web application with a REST API.

This prototype conducts the experiments on Amazon's Mobile Product Dataset-2022 [14] with 1000 plus mobile products and the obtained results are evaluated to prove the efficiency and scalability of the proposed FacetDRS by comparing with its counterparts like DeepRec [15], APGNN [16], and TADCF [17]. Here we set up total 20 users to use the FacetDRS prototype and its counterparts for testing their personalized recommendation across 16 requests for each user.

5.2 First Selections in Top-K

Table 1 provides a comprehensive analysis of the first selections made by user in top-k results of four recommendation systems—FacetDRS, TADCF [17], APGNN [16], and DeepRec [15] across 20 users and 16 queries per user. Each user simulated 16 interactions by submitting diverse queries based on their preferences, representing real-world shopping scenarios. The recorded values represent the counts of products recommended in the top-5 (T-5), top-10 (T-10), and top-15 (T-15) for each user and recommendation system.

<table>
<thead>
<tr>
<th>User</th>
<th>FacetDRS T-5</th>
<th>TADCF T-5</th>
<th>APGNN T-5</th>
<th>DeepRec T-5</th>
<th>FacetDRS T-10</th>
<th>TADCF T-10</th>
<th>APGNN T-10</th>
<th>DeepRec T-10</th>
<th>FacetDRS T-15</th>
<th>TADCF T-15</th>
<th>APGNN T-15</th>
<th>DeepRec T-15</th>
</tr>
</thead>
<tbody>
<tr>
<td>U1</td>
<td>9</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>9</td>
<td>1</td>
<td>5</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>U2</td>
<td>9</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>U3</td>
<td>10</td>
<td>4</td>
<td>3</td>
<td>8</td>
<td>3</td>
<td>9</td>
<td>2</td>
<td>5</td>
<td>6</td>
<td>7</td>
<td>3</td>
<td>9</td>
</tr>
<tr>
<td>U4</td>
<td>7</td>
<td>4</td>
<td>5</td>
<td>8</td>
<td>2</td>
<td>9</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>10</td>
<td>3</td>
<td>10</td>
</tr>
<tr>
<td>U5</td>
<td>7</td>
<td>3</td>
<td>4</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>9</td>
<td>7</td>
<td>2</td>
<td>9</td>
</tr>
<tr>
<td>U6</td>
<td>11</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>8</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>6</td>
<td>4</td>
<td>6</td>
</tr>
<tr>
<td>U7</td>
<td>5</td>
<td>3</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>2</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>U8</td>
<td>13</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>13</td>
<td>5</td>
<td>3</td>
<td>13</td>
</tr>
<tr>
<td>U9</td>
<td>9</td>
<td>4</td>
<td>6</td>
<td>7</td>
<td>1</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>4</td>
</tr>
<tr>
<td>U10</td>
<td>8</td>
<td>4</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>4</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>5</td>
</tr>
<tr>
<td>U11</td>
<td>7</td>
<td>6</td>
<td>5</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>3</td>
<td>12</td>
<td>8</td>
<td>8</td>
<td>8</td>
<td>8</td>
</tr>
<tr>
<td>U12</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>7</td>
<td>1</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>3</td>
<td>7</td>
<td>4</td>
<td>7</td>
</tr>
<tr>
<td>U13</td>
<td>7</td>
<td>5</td>
<td>4</td>
<td>7</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>10</td>
<td>3</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>U14</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>5</td>
<td>3</td>
<td>7</td>
<td>6</td>
<td>4</td>
<td>17</td>
<td>7</td>
<td>7</td>
<td>7</td>
</tr>
<tr>
<td>U15</td>
<td>11</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>13</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>U16</td>
<td>10</td>
<td>3</td>
<td>7</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>U17</td>
<td>8</td>
<td>3</td>
<td>5</td>
<td>6</td>
<td>2</td>
<td>5</td>
<td>4</td>
<td>3</td>
<td>9</td>
<td>5</td>
<td>5</td>
<td>5</td>
</tr>
<tr>
<td>U18</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>2</td>
<td>6</td>
<td>3</td>
<td>5</td>
<td>9</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>U19</td>
<td>10</td>
<td>4</td>
<td>5</td>
<td>5</td>
<td>2</td>
<td>7</td>
<td>6</td>
<td>3</td>
<td>11</td>
<td>3</td>
<td>3</td>
<td>3</td>
</tr>
<tr>
<td>U20</td>
<td>9</td>
<td>3</td>
<td>6</td>
<td>6</td>
<td>2</td>
<td>4</td>
<td>4</td>
<td>5</td>
<td>10</td>
<td>4</td>
<td>4</td>
<td>4</td>
</tr>
</tbody>
</table>

Figure 6 and 7 displays the average counts and percentages of successful first recommendations within the T-5, T-10, and T-15 positions across four distinct recommendation systems-FacetDRS, TADCF, APGNN, and DeepRec.

FacetDRS maintains a high precision, with an average count of 8.6 successful first recommendations in T5, indicating users consistently receive relevant suggestions within the top 5. The counts gradually decline for T10 (3.5) and T15 (2.7), demonstrating the system's effectiveness in offering precise recommendations. TADCF [17] achieves an average of 5.7 successful T5 recommendations, showcasing a moderate capability for delivering relevant initial suggestions. While maintaining consistency in T10 (5.4), a slight decline is observed in T15 (3.8), suggesting a nuanced precision shift in the extended subset. APGNN [16] exhibits commendable precision, with average counts of 6.7 in T5, 4.3 in T10, and 3.9 in T15. The system sustains a high level of precision in the top 5, gradually decreasing in subsequent ranks. DeepRec [15] excels in T10, recording an average count of 7.7,
highlighting its strength in providing relevant first recommendations within the top 10 positions. The system achieves an average count of 3.9 in T5 and 3.3 in T15.

Figure- 6. First Match Selection Count Average from Top-K Results

FacetDRS achieves a success rate of approximately 57.8% in T5, with decreasing percentages for T10 (23.1%) and T15 (19.1%), illustrating the distribution of successful recommendations across different subsets. TADCF exhibits consistent alignment, achieving success rates of 38.1% in T5, 34.4% in T10, and 27.5% in T15, indicating its ability to maintain relevance even in extended subsets. APGNN demonstrates balanced performance, with success rates of 44.4% in T5, 28.1% in T10, and 27.5% in T15, showcasing its ability to provide recommendations across different subsets. DeepRec excels with a success rate of 53.1% in T5, and while exhibiting lower percentages in T10 (21.3%) and T15 (25.6%), the system demonstrates a notable presence within the top 10.

Figure- 7. Comparison of First Match Selection percentage from Top-K Results

Results from table -1 presenting that, the FacetDRS demonstrates outstanding performance in delivering the relevant recommendations in terms of successful first selections with average and percentage metrics is emphasizing its efficiency in personalized dynamic recommendations.

5.3 Keyword Match Scores

The keyword matching score between User Preferences and Recommendation Systems is crucial for evaluating the effectiveness of recommendation systems. It provides a quantitative measure of how well the recommendations align with user-defined queries and filters using cosine similarity [36], offering insights into the system's ability to understand and cater to individual preferences.

Table 2. Comparison of Various Models generated Match Score between User Preferences and Recommendation System

<table>
<thead>
<tr>
<th>Results</th>
<th>FacetDRS</th>
<th>DeepRec</th>
<th>APGNN</th>
<th>TADCF</th>
</tr>
</thead>
<tbody>
<tr>
<td>Top 10</td>
<td>88%</td>
<td>80%</td>
<td>82%</td>
<td>75%</td>
</tr>
<tr>
<td>Top 15</td>
<td>85%</td>
<td>75%</td>
<td>73%</td>
<td>78%</td>
</tr>
<tr>
<td>Top 20</td>
<td>82%</td>
<td>72%</td>
<td>70%</td>
<td>74%</td>
</tr>
</tbody>
</table>
Table-2 provides a concise and comprehensive overview of the alignment between user preferences and four distinct recommendation systems: FacetDRS, DeepRec, APGNN, and TADCF. Each row corresponds to different subsets of recommendations, including Top 10, Top 15, Top 20, and the entirety of results (All Results), while the columns represent keyword match scores for each recommendation system.

Figure- 8. Keyword Match Score Comparison for various Recommendation Systems

In the Top 10, FacetDRS excels with an 88% match score, outperforming others. APGNN closely follows with 82%, showing strong alignment, DeepRec and TADCF score 80% and 75%, respectively. Expanding to the Top 15, FacetDRS maintains a high 85% match score. DeepRec slightly drops to 75%, while APGNN and TADCF score 73% and 78%. In the Top 20, FacetDRS leads with 82%, while DeepRec, APGNN, and TADCF score 72%, 70%, and 74%, respectively. Finally, across All Results, FacetDRS maintains an impressive 80% match score. DeepRec, APGNN, and TADCF score 70%, 68%, and 70%, reflecting their relevance across the entire set. Figure-8 presents the graphical view of the overall performance across the top results, FacetDRS maintains robust and sustained alignment with user preferences.

5.4 Average Response Time

The average response time [42, 44, 45, and 46] is a crucial metric in evaluating the scalability of recommendation systems. It directly reflects the efficiency and speed with which a system processes user queries and delivers recommendations. In the context of recommendation engines, low average response times are indicative of improved scalability, demonstrating the system's ability to handle increasing loads and user requests without compromising performance. This metric is particularly significant for real-time or interactive applications, where timely and responsive recommendations are essential for user satisfaction.

The presented table-3 encapsulates the average response times for each user’s 16 requests across four different recommendation models: FacetDRS, DeepRec, APGNN, and TADCF. The response times are measured in seconds, providing a comprehensive overview of the temporal efficiency of each system. The table not only offers individual user response times but also computes the average response time for all users, facilitating a comparative analysis of the models.

Table-3 Presentation of Average Response Time of four recommender system in processing user queries
The average response time comparison in figure-9 presents that, the FacetDRS stands out as the most scalable recommendation model, with an impressive average response time of 69.3 seconds. This highlights its efficiency in rapidly processing user queries, making it an optimal choice for applications requiring swift and responsive recommendations. DeepRec shows competitive scalability with an average response time of 88.2 seconds, reflecting its ability to handle diverse user requests. APGNN exhibits a higher average response time of 101.5 seconds, indicating a moderate level of scalability compared to the other models. TADCF presents an average response time of 86.4 seconds, positioning itself as a scalable recommendation model, though slightly trailing behind FacetDRS.

**Figure-9. Comparison of Average Response Time for Each Model and User**

In overall the FacetDRS not only excels in terms of average response time, showcasing its scalability, but also outperforms its counterparts in critical aspects such as First Selections in Top-K, keyword match scores, and overall system performance. Compared to DeepRec, APGNN, and TADCF, FacetDRS emerges as a comprehensive solution that excels better performance in all possible dimensions. This multi-faceted superiority positions FacetDRS as a preferred choice for applications where a balance between responsiveness, accuracy, and relevance is crucial, underscoring its prominence in the domain of recommendation systems.

**VI. CONCLUSION**

In this research we presented FacetDRS, a novel dynamic product recommendation system that addresses the limitations of existing solutions in e-commerce. While current systems often struggle with nuanced preferences and scalability, FacetDRS tackles these challenges with its innovative and multifaceted approach. By offering three distinct matching methods - Exact Match, High-Impact Partial Match, and Extended Partial Match - FacetDRS caters to diverse user needs and preferences, ensuring both accuracy and exploration beyond initial selections. This enables users to discover potentially better options within their desired budget or category, ultimately leading to a more satisfying and engaging shopping experience. Rigorous experimental evaluations confirmed FacetDRS's superior performance compared to established systems, demonstrating its effectiveness in the mobile commerce landscape. Moving forward, research could explore further personalization strategies and
integrate context-aware recommendations to provide even more tailored and dynamic product suggestions. With its flexible and scalable architecture, FacetDRS paves the way for a future of personalized and efficient e-commerce recommendations, driving user satisfaction and business success.

REFERENCES


Zareena Noorbasha earned her Bachelor’s degree in Computer Science & Engineering from Nagarjuna University in 2006 and her Master’s degree in Computer Science from School of IT JNTU Hyderabad in 2009. She is currently a research scholar at department of Computer Science and Systems Engineering, Andhra University. She is working as an Assistant Professor in the department of CSE at RVR & JC college of Engineering, AP . She is a member of ACM professional and life member of IAENG. She got 15 years of experience in teaching and her research interest includes Machine Learning and NLP.

Email: zareenanoorbasha@gmail.com

Dr. B Tarakeswara Rao working as Professor in the Department of Computer Science and Engineering at Kallam Haranadhareddy Institute of Technology, Guntur, Andhra Pradesh. He completed M.Tech from Acharya Nagarjuna University, Guntur in 2007 and completed his Ph.D from Acharya Nagarjuna University in 2012. He published more than 50 Research papers in various International Journals and presented more than 10 research papers in various National and International Conferences. He published 10 book chapters in various books. He got 21 years of experience in teaching and research. His research interests are Artificial Intelligence, Data Science, Machine Learning, Big Data and Deep Learning.

Email: tarak7199@gmail.com.