DEPARTMENT OF AI & DS

U23AIV12 / RECOMMENDER SYSTEMS

UNIT WISE QUESTION BANK - ANSWERS

UNIT I INTRODUCTION

Introduction and basic taxonomy of recommender systems - Traditional and non- personalized Recommender Systems - Overview of data mining methods for recommender systems- similarity measures- Dimensionality reduction – Singular Value Decomposition (SVD).

Q.No	Question	Answer
1	Define a recommender system.	A recommender system is a software application that provides suggestions to users for items such as books, movies, products, or music based on their past preferences, behavior, or similarities with other users. It helps reduce information overload and improves user satisfaction by showing the most relevant content.
2	List any two objectives of recommender systems.	The main objectives are: (i) To help users discover relevant items from a large collection without spending much time searching. (ii) To increase business profit and customer engagement by providing personalized recommendations that align with user interests.
3	Differentiate between personalized and non-personalized recommendation.	Personalized recommendations are generated based on individual user preferences, history, and behavior (e.g., Netflix suggesting movies based on your watchlist). Non-personalized recommendations are the same for all users, such as "Top 10 trending movies" on YouTube. Personalized systems are more accurate but need user data, while non-personalized are simple but less specific.
4	What is the role of similarity measures in recommender systems?	Similarity measures quantify how alike two users or items are based on their attributes or ratings. For example, if two users have rated movies in a similar way, their similarity score will be high, and the system can recommend items liked by one user to the other. These measures are essential in collaborative filtering.
5	Mention two real-life applications of recommender systems.	Recommender systems are widely used in (i) **E-commerce platforms** like Amazon to suggest products based on browsing and purchase history. (ii) **Streaming platforms** such as YouTube and Spotify, which recommend videos or songs according to user preferences and listening history.
6	Define traditional recommender systems with an example.	Traditional recommender systems are those that mainly use explicit ratings or implicit user behavior to suggest items. For example, in a **user-based collaborative filtering system**, if User A and User B rate many movies similarly, the system recommends movies liked by User A to User B.
7	What are the main challenges in recommender systems?	The main challenges include: (i) **Cold Start Problem** – difficulty in recommending items to new users with no history. (ii) **Data Sparsity** – when rating data is very limited. (iii) **Scalability** – difficulty in

		handling very large datasets. (iv) **Accuracy vs. Diversity Trade-off** in recommendations.
8	Write the formula for cosine similarity.	Cosine similarity measures the angle between two vectors and is widely used in text and recommender systems. The formula is: Cosine(A,B)= $\frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$, Where A and B are rating vectors of two users or items.
9	What is dimensionality reduction?	Dimensionality reduction is the process of reducing the number of features (dimensions) in a dataset while preserving essential information. In recommender systems, it is used to simplify the large user-item rating matrix into a smaller representation, making computations faster and easier while retaining accuracy.
10	State any two advantages of dimensionality reduction.	Advantages include: (i) It reduces the computational complexity and memory usage by working with fewer dimensions. (ii) It removes noise and redundancy in the data, thereby improving the performance and accuracy of recommender systems.
11	Define Singular Value Decomposition (SVD).	Singular Value Decomposition (SVD) is a mathematical matrix factorization technique that decomposes a given matrix into three smaller matrices: U, Σ, and V^T. In recommender systems, SVD is used to approximate the large user-item rating matrix and discover hidden relationships between users and items.
12	List the components of SVD.	SVD decomposes a matrix into three parts: (i) **U** – an orthogonal matrix representing user features. (ii) **Σ (Sigma)** – a diagonal matrix containing singular values that represent the strength of each feature. (iii) $\mathbf{V^T}$ – an orthogonal matrix-representing item features.
13	Write the formula of Jaccard similarity.	Jaccard similarity is used to measure similarity between two sets and is defined as: $J(A,B) = \frac{\|A \cap B\|}{\|A \cup B\|}$
14	Give one limitation of Euclidean distance in recommender systems.	Euclidean distance treats all features equally and is sensitive to differences in scale. In sparse datasets (like user-item ratings), many missing values lead to inaccurate distance calculations, making it unsuitable in many recommender applications without preprocessing.
15	What is the difference between supervised and unsupervised learning in data mining?	In **supervised learning**, the algorithm is trained using labeled data (input-output pairs), e.g., classification of emails as spam or not spam. In **unsupervised learning**, there are no labels, and the algorithm finds hidden patterns or groups in the data, e.g., clustering users with similar purchase behavior.

16	Mention two popular similarity measures used in recommender systems.	Common similarity measures include: (i) **Cosine Similarity** – measures angle between rating vectors. (ii) **Pearson Correlation** – measures linear correlation between two users' or items' rating patterns. Both are widely used in collaborative filtering.
17	What is the purpose of latent factors in SVD?	Latent factors are hidden features discovered by matrix factorization that explain user preferences and item characteristics. For example, in a movie recommender system, latent factors may represent hidden features like "action preference" or "romantic preference" that are not directly available in the dataset.
18	Define taxonomy of recommender systems.	Taxonomy is the classification of recommender systems into categories such as **Content-Based Filtering** (uses item features), **Collaborative Filtering** (uses user behavior), **Hybrid Systems** (combination), and **Knowledge-Based Systems** (uses domain knowledge).
19	Give any two examples of non-personalized recommendations.	Examples: (i) **Most popular movies** on a platform like IMDb's "Top 250 Movies" list. (ii) **Trending videos** on YouTube, where the same recommendations are shown to all users regardless of their history.
20	What is the main goal of data mining in recommender systems?	The main goal of data mining is to discover useful patterns and relationships in large datasets, which can then be used to predict user preferences. For instance, clustering users based on past purchases helps in making personalized product recommendations.

PART –B

Q.No	Question	Elaborated Answer (Exam Length)
1	Explain the basic taxonomy of recommender systems with neat diagram.	**Taxonomy of recommender systems** classifies them into major categories: (i) **Content-Based Filtering** – recommends items similar to those the user has liked before using features like genre, keywords, etc. (ii) **Collaborative Filtering** – makes predictions based on preferences of similar users or items. Two types: *User-based* (finds users with similar tastes) and *Item-based* (finds items with similar rating patterns). (iii) **Hybrid Systems** – combine two or more approaches to overcome individual limitations. (iv) **Knowledge-Based Systems** – rely on explicit knowledge about items and user needs (e.g., recommending a digital camera based on lens quality). Include a neat diagram with branches. **Advantages:** Handles diverse applications. **Disadvantages:** Increased complexity in hybrid approaches.
2	Discuss the working of traditional recommender systems with suitable examples.	Traditional recommender systems primarily use explicit **ratings** (e.g., 1–5 stars) or implicit **behavioral data** (clicks, purchases). **Collaborative Filtering** is most common: (i) *User-based CF* – if User A and User B have similar past ratings, then items liked by A are recommended to B. (ii) *Item-based CF* – if two items are rated similarly by many users, then liking one suggests liking the other. **Example:**

		MovieLens dataset – if two users rated "Inception" and "Interstellar" highly, the system recommends similar sci-fi movies. **Limitations:** Cold start, sparsity, scalability issues.
3	Differentiate between personalized and non-personalized recommender systems with examples.	**Personalized Systems:** Recommendations vary per user, based on past behavior. Example: Netflix suggesting different movies to two different users. **Non-personalized Systems:** Show the same popular/trending items to all users, e.g., Amazon "Best Seller List." **Comparison:** Personalized = accurate, requires user data; Non-personalized = simple, works for new users. **Table format comparison:** (Criteria → Input data, Accuracy, User dependency, Example).
4	Describe the role of data mining methods in recommender systems.	Data mining extracts useful patterns from large datasets. **Methods used:** (i) **Classification** – predict categories (e.g., spam filtering). (ii) **Clustering** – group similar users/items (e.g., segmenting customers). (iii) **Association Rule Mining** – find item co-occurrence patterns (e.g., "users who bought X also bought Y"). (iv) **Regression** – predict continuous ratings. **Role in recommenders:** Helps in user profiling, similarity discovery, and preference prediction. **Example:** Market basket analysis in Amazon.
5	Explain different similarity measures with mathematical expressions and examples.	(i) **Euclidean Distance:** Measures straight-line distance. $$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$ (ii) **Cosine Similarity:** Based on angle. $$\frac{A \cdot B}{\|A\| \times \|B\|} = \frac{\sum_{i=1}^{n} A_i \times B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \times \sqrt{\sum_{i=1}^{n} B_i^2}}$$ (iii) **Pearson Correlation:** Measures linear relationship. $$r = \frac{\Sigma(X-\overline{X})(Y-\overline{Y})}{\sqrt{\Sigma(X-\overline{X})^2}\sqrt{(Y-\overline{Y})^2}}$$ Where, $\overline{X}$ = mean of X variable, $\overline{Y}$ = mean of Y variable **iv) Jaccard Similarity:** $$J(A,B) = \frac{|A \cap B|}{|A \cup B|}$$
6	Discuss the importance of dimensionality reduction in recommender systems.	In recommender systems, user-item matrices are very large and sparse. **Importance:** (i) Reduces storage and computation requirements. (ii) Removes noise and redundancy. (iii) Helps uncover hidden latent factors. (iv) Improves accuracy and scalability. **Techniques:** PCA, SVD. **Example:** Netflix prize challenge – dimensionality reduction using SVD improved performance significantly.

7	Explain Singular Value Decomposition (SVD) technique with a worked example.	**SVD:** Factorizes a rating matrix MM into $A = U\Sigma V^T$. Here, U = user matrix, Σ = diagonal singular values, V^T = item matrix. **Steps:** (i) Start with a rating matrix (3×3). (ii) Decompose into three smaller matrices. (iii) Approximate by keeping top k singular values (rank reduction). (iv) Use this to predict missing values. **Example:** Small user-movie rating matrix with missing values approximated using top 2 latent factors.
8	Compare Euclidean distance, Cosine similarity, and Jaccard similarity with examples.	**Euclidean Distance:** Geometric, sensitive to scale. **Cosine Similarity:** Focuses on direction not magnitude, good for text and ratings. **Jaccard:** Works for binary/sets, ignores magnitude. **Example:** Compare User A {1,0,1,1} and User B {1,1,0,1} → Euclidean = 1.41, Cosine = 0.81, Jaccard = 0.5. **Comparison Table** with formula, data type, use cases.
9	Describe the challenges and limitations of traditional recommender systems.	**Challenges:** (i) **Cold Start:** No history for new users/items. (ii) **Data Sparsity:** Few ratings given compared to possible ratings. (iii) **Scalability:** Difficult to handle millions of users/items. (iv) **Synonymy:** Different items with similar meaning not recognized. (v) **Privacy Concerns. Limitations:** Often inaccurate for niche users/items. Example: New movies on Netflix suffer cold start.
10	Explain how data mining and dimensionality reduction together improve recommender systems.	**Data Mining Role:** Finds useful patterns, clusters users, predicts preferences. **Dimensionality Reduction Role:** Simplifies large sparse matrices, extracts hidden latent factors. **Together:** Improve efficiency, reduce memory, enhance accuracy. **Example:** Netflix → clustering (data mining) + SVD (dimensionality reduction) → improved RMSE in rating predictions.

PART – C

Q.No	Question	Elaborated Answer (Essay Length)
1	With a neat diagram, explain the architecture and taxonomy of recommender systems.	**Architecture of recommender systems:** (i) **Input Layer:** Collects user data (ratings, clicks, profiles) and item data (attributes, metadata). (ii) **Processing Layer:** Applies filtering algorithms such as collaborative filtering, content-based methods, similarity computations, or matrix factorization. (iii) **Output Layer:** Generates ranked list of recommended items. **Diagram:** Show block diagram (User Data + Item Data → Algorithm → Output). **Taxonomy:** (a) Content-based, (b) Collaborative (user-based, item-based), (c)

		Hybrid, (d) Knowledge-based. **Advantages & Disadvantages:** Content-based – interpretable but over-specialization; Collaborative – powerful but suffers cold start; Hybrid – solves issues but complex.
2	Explain in detail the process of Singular Value Decomposition (SVD) and show how it helps in dimensionality reduction for recommender systems with an example.	**Definition:** SVD decomposes a user-item rating matrix MM into $A = U\Sigma V^T$ **Steps:** (i) Create rating matrix. (ii) Factorize using SVD. (iii) Retain top k singular values (low-rank approximation). (iv) Use reduced representation to predict missing ratings. **How it helps:** Removes noise, compresses data, identifies latent features. **Example:** Consider a 4×4 movie-rating matrix. Using SVD, reduce to rank-2 approximation. Predict missing ratings by reconstructing matrix. **Application:** Netflix Prize challenge – SVD improved predictions by uncovering hidden factors like "action preference" or "romantic preference."
3	Write an essay on similarity measures used in recommender systems. Compare them with advantages, disadvantages, and applications.	**Similarity measures:** (i) **Euclidean Distance:** Simple, geometric measure. Works poorly with sparse data. (ii) **Cosine Similarity:** Focuses on orientation of vectors. Best for text and rating comparisons. (iii) **Pearson Correlation:** Measures linear relationship. Handles rating scale differences well. (iv) **Jaccard Index:** Works for set/binary data, ignores frequency. (v) **Manhattan Distance:** Measures sum of absolute differences, simpler than Euclidean. **Comparison Table:** Include formula, use case, pros/cons. (vi) **Applications:** Euclidean → clustering, Cosine → text mining, Pearson → rating normalization, Jaccard → market basket analysis. (vii) **Conclusion:** Choice depends on data type and application domain.

UNIT II CONTENT-BASED RECOMMENDATION SYSTEMS

High-level architecture of content-based systems - Item profiles, Representing item profiles, Methods for learning user profiles, Similarity-based retrieval, and Classification algorithms.

Q.No	Question	Answer (Elaborate, 2 Marks)
1	What is a content-based recommendation system?	A content-based recommendation system suggests items to a user by analyzing the features or attributes of items and matching them with the user's past preferences. It relies on **item descriptions** (item

		profiles) and the **user's profile**, instead of using ratings from other users.
2	Define item profile in recommender systems.	An **item profile** is a structured representation of an item, describing it through a set of features or attributes. For example, a movie's profile may include title, genre, actors, director, and keywords. These features are used for similarity comparisons with user preferences.
3	What is a user profile in content-based filtering?	A **user profile** represents the interests, preferences, and behavior of a user. It is built from the user's history of interactions (liked movies, purchased books, etc.) and contains weighted features that capture the user's tastes.
4	Differentiate between user profile and item profile.	- **User profile**: Describes the preferences and interests of a user. - **Item profile**: Describes the features/attributes of an item. Content-based filtering works by matching these two profiles.
5	Mention any two advantages of content-based filtering.	1. Provides **personalized recommendations** based on individual user interests.2. Does not require data from other users, hence works well even with **sparse datasets**.
6	Mention any two disadvantages of content-based filtering.	1. **Limited content analysis** – depends on the quality of item features.2. **Overspecialization** – user gets only similar items, lacking diversity.
7	What is high-level architecture of a content-based system?	The high-level architecture consists of three modules: **Item representation (item profiles)**, **User modeling (user profiles)**, and **Recommendation engine** which computes similarity and suggests items.
8	List the steps in content-based recommendation process.	1. Extract features from items → form item profiles.2. Learn user preferences → create user profile.3. Match user profile with item profiles using similarity measures.4. Recommend the most similar items.
9	What are feature extraction methods for item profiles?	Common methods include **keyword extraction, TF-IDF representation, word embeddings**, and **metadata-based attributes** (e.g., author, genre, category). These features help in creating structured item profiles.
10	Define similarity-based retrieval.	**Similarity-based retrieval** is the process of finding and ranking items by comparing item profiles with the user profile using similarity measures such as **cosine similarity, Jaccard coefficient, or Euclidean distance**.
11	What are similarity measures? Give examples.	**Similarity measures** are mathematical functions used to compare user and item profiles. Examples include **Cosine similarity, Pearson correlation, Jaccard similarity, and Euclidean distance**.
12	What is cosine similarity?	**Cosine similarity** measures the cosine of the angle between two vectors (e.g., user profile and item profile). Values range from -1 to +1, where **+1 indicates high similarity**.

13	What is Jaccard similarity?	Jaccard similarity is a measure of similarity between two sets, defined as:**J(A,B) =
14	What is Euclidean distance in recommender systems?	**Euclidean distance** measures the straight-line distance between two profile vectors in multidimensional space. Smaller distance means higher similarity.
15	What are the methods for learning user profiles?	User profiles can be learned using:1. **Explicit feedback** (ratings, likes).2. **Implicit feedback** (clicks, browsing history, purchases).3. **Hybrid methods** combining both.
16	Define classification algorithms in content-based recommendation.	Classification algorithms treat recommendation as a **supervised learning problem** where items are classified into "relevant" or "non-relevant" for a user. Examples: **Naïve Bayes, Decision Trees, SVM**.
17	Give an example of classification algorithm used in content-based filtering.	**Naïve Bayes** is often used, where features of an item (e.g., keywords of a document) are treated as evidence, and the probability that a user likes an item is calculated.
18	Mention one application of content-based filtering.	**Movie recommendation**: Suggesting movies similar to those a user has previously liked based on attributes such as genre, director, and cast.
19	How does overspecialization occur in content-based systems?	Overspecialization occurs when the system recommends only items very similar to what the user already consumed, reducing **novelty and diversity** in recommendations.
20	Differentiate between content-based and collaborative filtering.	- **Content-based**: Relies on item features and user's own history.- **Collaborative filtering**: Relies on ratings and preferences of similar users.

PART –B

Q.No	Question	Answer (Elaborated for 13 Marks)
1	Explain the high-level architecture of content-based recommender systems.	Content-based systems recommend items by analyzing the features of items and matching them with user preferences. The **architecture** includes: (i) **Content Analyzer** – extracts features from items (e.g., keywords, tags, attributes). (ii) **Profile Learner** – builds user profiles based on past interactions. (iii) **Filtering Component** – compares new items with the user profile to recommend similar items. The process: user gives feedback → profile updated → matching with item profiles → recommendations generated. (iv) Example: Netflix recommending movies based on watched genres.

		Diagram: (User → Content Analyzer → Profile Learner → Filtering → Recommendation list).
2	Describe item profiles and methods to represent them.	Item profiles are structured descriptions of items using attributes. **Representation methods:** (i) **Keyword-based** – items represented as keyword sets (e.g., book = {AI, ML, Python}). (ii) **Vector space model** – each item represented as a feature vector with weights (e.g., TF-IDF in text data). (iii) **Structured attributes** – using categorical/numerical fields (genre, author, year). (iv) **Ontology-based** – semantic representation linking related attributes. Importance: Helps compare items and calculate similarity with user profiles. Example: Spotify represents songs with features like tempo, genre, artist.
3	Explain different methods for learning user profiles in content-based systems.	User profiles capture interests/preferences. **Methods:** (i) **Explicit feedback** – direct input (ratings, likes). (ii) **Implicit feedback** – inferred from behavior (clicks, browsing time, purchases). (iii) **Machine learning models** – supervised learning (classification, regression) for predicting user interest. (iv) **Incremental learning** – continuously updating profile as user interacts. (v) **Hybrid methods** – combining explicit + implicit. Example: Amazon learns user profiles from browsing + purchase history.
4	Describe similarity-based retrieval in content-based recommender systems.	Similarity-based retrieval compares **user profiles** with **item profiles** to find the most relevant items. **Steps:** (i) Represent user & item profiles as vectors. (ii) Apply similarity measures (cosine similarity, Jaccard, Euclidean). (iii) Rank items based on similarity scores. (iv) Recommend top-N items. Example: If a user liked a "romantic comedy," system retrieves other items with high similarity in keywords/genre. **Importance:** Improves personalization by retrieving items most related to user's past interests.
5	Explain the role of classification algorithms in content-based recommender systems.	Classification algorithms help predict whether a user will like an item. **Steps:** (i) Training: Build model using user-item interaction data. (ii) Input: Item profile + user profile. (iii) Output: Class label (like/dislike). **Algorithms used:** (a) Naïve Bayes – assumes feature independence, predicts probability of preference. (b) Decision Trees – classify items into liked/not liked based on attributes. (c) SVM – separates preferred vs. non-preferred items in feature space. (d) Neural Networks – learn complex user-item patterns. Example: Classify movies as "recommended" or "not recommended" for a user.
6	Compare item profiles and user profiles with examples.	**Item profile** = structured description of an item (features/attributes). Example:

		Book → {Title: "AI Basics", Genre: "Education", Author: "Smith"}. **User profile** = model of user preferences. Example: User → {Preferred genres: Education, AI; Dislikes: Fiction}. **Comparison:** (i) Item profile → about items; User profile → about preferences. (ii) Item profile → static; User profile → dynamic (updates). (iii) Interaction: Matching item profiles with user profiles generates recommendations. Example: A user profile favoring AI books matches with an item profile of an AI textbook.
7	Discuss advantages and limitations of content-based recommender systems.	**Advantages:** (i) Personalization – tailored to user. (ii) Transparency – easy to explain ("recommended because you liked X"). (iii) No cold-start for items (new items can be recommended if attributes known). (ii) **Limitations:** (i) Cold-start for new users (no profile initially). (ii) Limited content analysis – features may not capture full meaning. (iii) Over-specialization – recommends only similar items, lacks diversity. Example: If user watches only "horror," system keeps suggesting horror only.
8	Explain with example how user profiles are updated in content-based recommender systems.	User profiles are updated by monitoring user interactions. **Methods:** (i) **Explicit update** – user rates new items, profile recalculated. (ii) **Implicit update** – system tracks clicks, time spent, purchases. (iii) **Incremental learning** – profile gradually adjusted. Example: Initially, user profile = {Comedy: 60%, Action: 40%}. If the user watches 3 new action movies, system updates profile → {Comedy: 40%, Action: 60%}. This ensures evolving preferences are reflected.
9	Compare similarity-based retrieval and classification-based recommendation.	**Similarity-based retrieval:** (i) Finds items similar to user profile. (ii) Uses similarity metrics (cosine, Jaccard). (iii) Example: "Liked movie = Titanic → recommends romantic drama movies." **Classification-based:** (i) Learns a predictive model from labeled data. (ii) Uses algorithms like Decision Trees, SVM. (iii) Example: Classify a new movie as "liked/disliked" for user. **Comparison:** Similarity-based = unsupervised ranking; Classification-based = supervised prediction.
10	Write short notes on integrating classification with content-based filtering.	Integration helps improve recommendation accuracy. **Process:** (i) Represent item features. (ii) Collect user feedback. (iii) Train classifier (Naïve Bayes, SVM, Neural Net). (iv) Use classifier output as input to content-based retrieval. **Benefits:** (a)

		More accurate predictions. (b) Handles noisy/incomplete profiles. (c) Supports binary/multi-class preferences. Example: Hybrid system where similarity scores rank items and classification confirms recommendation relevance.

PART – C

Q.No	Question	Answer
1	Explain the **high-level architecture of a Content-Based Recommender System** with a neat diagram.	A Content-Based Recommender System (CBRS) recommends items to a user by analyzing item descriptions and comparing them with the user's preferences. **Components:** 1. **Item Database:** Stores metadata about items (e.g., title, genre, author for books). 2. **Feature Extraction:** Converts item descriptions into structured attributes. 3. **User Profile:** Built from past interactions and preferences. 4. **Learning Module:** Updates user profile dynamically. 5. **Similarity Engine:** Compares user profile with item profiles. 6. **Recommendation Engine:** Produces a ranked list of items. **Diagram (must draw in exam):** [Architecture showing user → profile learner → item profiles → similarity engine → recommendation list]. **Advantages:** Personalized, interpretable. **Limitations:** Overspecialization, cold start problem.
2	Discuss **methods for learning user profiles** with suitable examples.	User profiles capture a user's interests/preferences. **Methods:** 1. **Explicit Feedback:** Ratings, likes/dislikes, reviews. – Example: Netflix asking users to rate movies. 2. **Implicit Feedback:** Clicks, browsing time, purchase history. – Example: Amazon tracking time spent on products. 3. **Content Analysis:** Extracting keywords, tags from consumed items. 4. **Machine Learning Models:** – **Naïve Bayes:** Probabilistic user preference prediction. – **Decision Trees:** Learn rules from past choices. – **Neural Networks:** Capture nonlinear relationships. 5. **Hybrid Approaches:** Combine explicit + implicit (e.g., Spotify). **Challenges:** Profile sparsity, evolving preferences. **Conclusion:** Effective learning = more accurate personalization.
3	Analyze the role of **similarity-based retrieval** in Content-Based Recommendation and compare similarity measures with examples.	**Definition:** Similarity-based retrieval = finding items most similar to user preferences using mathematical similarity measures. **Process:** 1. Represent user profile & item profiles as vectors. 2. Compute similarity score. 3. Rank items in decreasing similarity. **Similarity Measures:** 1. **Cosine Similarity** – angle between two vectors. – Example: Movie profile vectors. 2. **Jaccard Similarity** – overlap between sets. – Example: Comparing tags of two songs. 3. **Euclidean Distance** – geometric distance. – Example: Comparing book feature scores. 4. **Pearson Correlation** – statistical similarity. **Example:** User likes "Action,

		Thriller" → CBRS retrieves movies with similar feature vectors using cosine similarity. **Conclusion:** Choice of similarity measure directly affects recommendation accuracy.

UNIT III COLLABORATIVE FILTERING

A systematic approach, Nearest-neighbor collaborative filtering (CF), user-based and item-based CF, components of neighborhood methods (rating normalization, similarity weight computation, and neighborhood selection

Q.No	Question	Answer
1	What is collaborative filtering?	Collaborative Filtering (CF) is a recommendation approach that predicts user preferences by leveraging the ratings, interests, or behaviors of similar users. Instead of using item content, CF relies on the principle that "users with similar tastes in the past will continue to share preferences in the future."
2	Define user-based collaborative filtering.	User-based CF recommends items by finding users with similar rating patterns (neighbors) and suggesting items liked by them. For example, if User A and User B rate many movies similarly, User A may get recommendations of movies liked by User B.
3	Define item-based collaborative filtering.	Item-based CF focuses on similarities between items instead of users. Recommendations are generated by identifying items similar to those the target user has already liked or rated. For example, if a user likes "Inception," the system may suggest "Interstellar" due to high item similarity.
4	What is the main principle of CF?	The main principle of CF is that people who agreed in the past are likely to agree in the future. This is achieved by analyzing patterns in rating data across a large number of users and items.
5	List the two main types of CF.	The two main types are: (1) User-Based Collaborative Filtering, (2) Item-Based Collaborative Filtering. Both methods use neighborhood-based approaches but differ in whether they focus on user similarity or item similarity.
6	What is neighborhood selection in CF?	Neighborhood selection refers to choosing the top-k most similar users (in user-based CF) or items (in item-based CF) to form a group. This neighborhood is then used to predict the missing ratings and generate recommendations.
7	What is rating normalization in CF?	Rating normalization is the process of adjusting user ratings to remove personal rating biases. For instance, some users give higher ratings on average while others give lower. Normalization ensures fair comparison by centering ratings around each user's average.
8	Mention any two similarity measures used in CF.	(1) Cosine Similarity – measures the cosine of the angle between rating vectors. (2) Pearson Correlation – measures linear correlation between user/item rating patterns.

9	What is similarity weight computation?	It is the process of calculating how strongly two users (or items) are related based on their ratings. The similarity weights are used to give more importance to closer neighbors in making predictions.
10	Give an example of user bias in ratings.	Suppose User X usually gives very high ratings (4–5 stars), while User Y usually rates strictly (2–3 stars). Even if they like the same movies, their raw ratings differ. This bias can affect CF, hence normalization is required.
11	What is cold-start problem in CF?	The cold-start problem occurs when new users or new items enter the system with little or no rating history, making it difficult for CF algorithms to generate recommendations.
12	Differentiate between memory-based and model-based CF.	Memory-based CF uses direct user–item ratings to compute recommendations in real-time (e.g., neighborhood methods). Model-based CF builds predictive models using machine learning (e.g., matrix factorization).
13	What is a similarity matrix?	A similarity matrix is a table that stores similarity scores between all pairs of users or items. It is used by CF algorithms to quickly identify the most similar neighbors during recommendation.
14	Give an example of item-based similarity.	If most users who like “Harry Potter” also like “Lord of the Rings,” then the similarity between these two books will be high. A user who liked “Harry Potter” may be recommended “Lord of the Rings.”
15	What is sparsity in CF?	Sparsity refers to the fact that in real-world recommendation systems, most users rate only a very small subset of all available items. This leads to a sparse user-item rating matrix.
16	Why is Pearson correlation preferred in CF?	Pearson correlation removes individual user rating bias by comparing deviations from the mean. It captures relative preferences better than raw cosine similarity in many cases.
17	Write one advantage of CF.	CF can recommend items without requiring item features or descriptions. It works purely on user interactions, making it widely applicable.
18	Write one disadvantage of CF.	CF suffers from the cold-start problem and data sparsity, which reduce recommendation quality when user–item interactions are limited.
19	What is k in k-nearest neighbor CF?	The parameter **k** specifies the number of neighbors (users or items) to consider when predicting ratings. Higher k provides stability but may reduce personalization, while lower k captures strong similarities but may be noisy.
20	State one difference between user-based and item-based CF.	User-based CF focuses on finding similar users, while item-based CF focuses on finding similar items. User-based adapts to individual preferences, while item-based is more scalable in large datasets.

PART – B

Q. No	Question	Answer (Elaborated – 13 Marks)
1	Explain the systematic approach of collaborative filtering in detail.	Collaborative Filtering (CF) follows a **systematic pipeline** for predicting user preferences: **Step 1: Data Collection** – User–item rating matrix is prepared (users = rows, items = columns, ratings = explicit/implicit). **Step 2: Rating Normalization** – Ratings are adjusted to remove user bias (e.g., some users give consistently high/low ratings). **Step 3: Similarity Computation** – Similarity between users (User-based CF) or items (Item-based CF) is computed using metrics like cosine similarity, Pearson correlation, Jaccard. **Step 4: Neighborhood Selection** – Top-K similar users/items are chosen as neighbors. **Step 5: Prediction** – Missing ratings are estimated by aggregating weighted ratings from neighbors. **Step 6: Recommendation** – Top-N items with the highest predicted ratings are recommended. **Advantages:** Personalized, adaptive, domain-independent. **Limitations:** Cold start, data sparsity, scalability. **Diagram:** User-item matrix with neighborhood illustration.
2	Differentiate between User-based and Item-based CF with example.	**User-based CF:** Finds similar users. If User A and User B have similar rating patterns, recommendations for User A are based on what User B liked. **Item-based CF:** Finds similar items. If Item X is similar to Item Y, then a user who likes X is recommended Y. **Example:** Suppose in a movie recommender: - User A likes *Inception*, *Interstellar*. - User B likes *Inception*, *Matrix*. - User-based: Recommend *Matrix* to User A because User B (similar) liked it. - Item-based: Recommend *Matrix* to User A because *Matrix* is similar to *Inception*. **Comparison Table:** - **Computation:** User-based requires finding neighbors among users; Item-based requires pre-computing item similarities. - **Scalability:** Item-based is better since number of items is usually smaller than users. - **Stability:** Item similarities change less often than user preferences.
3	Describe the components of neighborhood methods in collaborative filtering.	Neighborhood methods in CF consist of three key components: **1. Rating Normalization:** Removes individual user biases. Example: If User A always rates high, subtract user average rating from given rating. Formula: $r'_{ui} = r_{ui} - \bar{r}_u$. **2. Similarity Weight Computation:** Computes closeness between users/items. Common measures: Cosine similarity, Pearson correlation, Adjusted cosine. **3. Neighborhood Selection:** Top-K neighbors (users or items) are chosen. The neighborhood defines whose opinions will influence the prediction. **Example:** In a movie dataset, User A's rating on *Movie X* is predicted using ratings of the top-5 most similar users. **Advantages:** Easy to implement, interpretable. **Diagram:** Flowchart showing normalization → similarity → neighborhood → prediction.

4	Explain similarity measures used in collaborative filtering with formula.	Several similarity measures are used in CF: **1. Cosine Similarity:** Measures cosine of angle between rating vectors. Formula: $sim(u,v) = \frac{\sum_i r_{ui} r_{vi}}{\sqrt{\sum_i r_{ui}^2}\sqrt{\sum_i r_{vi}^2}}$. **2. Pearson Correlation:** Considers rating deviations from mean. Formula: $sim(u,v) = \frac{\sum_i (r_{ui}-\bar{r}_u)(r_{vi}-\bar{r}_v)}{\sqrt{\sum_i (r_{ui}-\bar{r}_u)^2}\sqrt{\sum_i (r_{vi}-\bar{r}_v)^2}}$. **3. Jaccard Similarity:** Used for implicit feedback (likes/dislikes). Formula: $J(A,B) = \frac{\lvert A \cap B \rvert}{\lvert A \cup B \rvert}$
5	Discuss challenges in implementing collaborative filtering.	Collaborative filtering faces the following challenges: **1. Data Sparsity:** User–item matrix is mostly empty, leading to unreliable similarity. **2. Cold Start:** New users or items have no ratings, making recommendations difficult. **3. Scalability:** Large datasets with millions of users/items require heavy computation. **4. Gray Sheep Problem:** Users with unusual tastes cannot be grouped with others. **5. Shilling Attacks:** Fake users may insert biased ratings to manipulate recommendations. **6. Diversity vs. Accuracy Trade-off:** High accuracy may reduce diversity in recommendations. **7. Real-time Updates:** Handling dynamic data streams in CF is challenging. **Solutions:** Hybrid methods, dimensionality reduction (SVD), approximate nearest neighbor search, incremental learning.
6	Explain rating normalization in detail with example.	Rating normalization removes personal bias from ratings. **Why?** Some users always rate items high (lenient raters) and some always rate low (strict raters). Without normalization, similarity scores may be biased. **Types:** - **Mean-Centering:** Subtract average rating of user. - **Z-score Normalization:** Scale ratings by mean and standard deviation. - **Item-based Normalization:** Subtract item average instead of user average. **Example:** User A rates [5,4,1]. Mean = 3.33. Normalized ratings = [1.67, 0.67, -2.33]. **Formula:** rui′=rui−r¯ur'_{ui} = r_{ui} - \bar{r}_u. **Advantages:** Improves fairness and accuracy in similarity computation.
7	Illustrate user-based CF algorithm with a worked-out example.	**Steps:** 1. Prepare user-item rating matrix. 2. Normalize ratings by subtracting user average. 3. Compute similarity between target user and others (e.g., Pearson correlation). 4. Select top-K similar users. 5. Predict missing rating: $\hat{r}_{u,i} = \bar{r}_u + \frac{\sum_{v \in N(u)} sim(u,v) \cdot (r_{v,i} - \bar{r}_v)}{\sum_{v \in N(u)} \lvert sim(u,v) \rvert}$

		$\hat{r}_{u,i}$ = Predicted rating of user u for item i. $\bar{r}_u$ = Average rating of user u. $N(u)$ = Neighborhood (set of most similar users to u). $sim(u,v)$ = Similarity between user u and user v (e.g., cosine similarity, Pearson correlation). $r_{v,i}$ = Actual rating given by user v for item i. $\bar{r}_v$ = Average rating of user v. Example: **User / Movie** | **M1** | **M2** | **M3** U1 | 5 | 3 | ? U2 | 4 | 2 | 5 U3 | 1 | 5 | 4 **Step 1: Compute similarity between U1 and others** We use **Cosine Similarity**: $$sim(u,v) = \frac{\sum(r_{u,i} \cdot r_{v,i})}{\sqrt{\sum r_{u,i}^2} \cdot \sqrt{\sum r_{v,i}^2}}$$ • Similarity(U1,U2) Ratings = (5,3) vs (4,2) → $$sim(U1,U2) = \frac{5\times4+3\times2}{\sqrt{5^2+3^2}\cdot\sqrt{4^2+2^2}} = \frac{20+6}{\sqrt{34}\cdot\sqrt{20}} = \frac{26}{5.83\cdot4.47} \approx 0.996$$ • Similarity(U1,U3) Ratings = (5,3) vs (1,5) → $$sim(U1,U3) = \frac{5\times1+3\times5}{\sqrt{34}\cdot\sqrt{26}} = \frac{20}{5.83\cdot5.10} \approx 0.674$$ • sim(U1,U2) ≈ 0.996 (very similar) • sim(U1,U3) ≈ 0.674 **Step 2: Apply prediction formula** We want $\hat{r}_{U1,M3}$. • Average ratings: • $\bar{r}_{U1} = (5+3)/2 = 4$ • $\bar{r}_{U2} = (4+2+5)/3 = 3.67$ • $\bar{r}_{U3} = (1+5+4)/3 = 3.33$ $$\hat{r}_{U1,M3} = \bar{r}_{U1} + \frac{sim(U1,U2)\cdot(r_{U2,M3}-\bar{r}_{U2}) + sim(U1,U3)\cdot(r_{U3,M3}-\bar{r}_{U3})}{|sim(U1,U2)|+|sim(U1,U3)|}$$ Substitute values: $$= 4 + \frac{0.996\cdot(5-3.67)+0.674\cdot(4-3.33)}{0.996+0.674}$$ $$= 4 + \frac{0.996\cdot1.33+0.674\cdot0.67}{1.67}$$ $$= 4 + \frac{1.325+0.451}{1.67}$$ $$= 4 + \frac{1.776}{1.67} \approx 4+1.06 = 5.06$$ Predicted rating of **U1 for M3 ≈ 5.1** → meaning **U1 is very likely to enjoy M3**.
8	Illustrate item-based CF algorithm with example.	**Steps:** 1. Build item-item similarity matrix (using cosine similarity). 2. For a target user, find items similar to those already rated. 3. Predict missing rating using: **Step 1: Formula for Similarity (Cosine Similarity)** For two items i and j: $$Sim(i,j) = \frac{\sum_{u\in U} r_{u,i}\cdot r_{u,j}}{\sqrt{\sum_{u\in U} r_{u,i}^2}\cdot\sqrt{\sum_{u\in U} r_{u,j}^2}}$$ Where: • $r_{u,i}$ = rating of user u on item i • U = set of users who rated both items

<table>
<tr>
<td></td>
<td></td>
<td>
Step 2: Prediction Formula
The predicted rating of user u on item i is:
$$\hat{r}_{u,i} = \frac{\sum_{j \in N(i)} Sim(i,j) \cdot r_{u,j}}{\sum_{j \in N(i)} |Sim(i,j)|}$$
Where:
• $N(i)$ = neighborhood set of items most similar to i
• $r_{u,j}$ = rating given by user u to item j

Example:
<table>
<tr><th>User</th><th>Item A</th><th>Item B</th><th>Item C</th><th>Item D</th></tr>
<tr><td>U1</td><td>5</td><td>3</td><td>4</td><td>?</td></tr>
<tr><td>U2</td><td>4</td><td>2</td><td>5</td><td>3</td></tr>
<tr><td>U3</td><td>1</td><td>1</td><td>2</td><td>1</td></tr>
<tr><td>U4</td><td>3</td><td>3</td><td>4</td><td>2</td></tr>
</table>
Step A: Compute Item Similarities
Take Item D vs. other items:
• Similarity(Item D, Item A)
$$Sim(D,A) = \frac{(3 \cdot 4) + (1 \cdot 1) + (2 \cdot 3)}{\sqrt{(4^2 + 1^2 + 3^2)} \cdot \sqrt{(3^2 + 1^2 + 2^2)}} = \frac{12 + 1 + 6}{\sqrt{26} \cdot \sqrt{14}} = \frac{19}{\sqrt{364}} \approx 0.996$$
• Similarity(D, B)
$$Sim(D,B) = \frac{(2 \cdot 2) + (1 \cdot 1) + (2 \cdot 3)}{\sqrt{(2^2 + 1^2 + 3^2)} \cdot \sqrt{14}} = \frac{4 + 1 + 6}{\sqrt{14} \cdot \sqrt{14}} = \frac{11}{14} \approx 0.786$$
• Similarity(D, C)
$$Sim(D,C) = \frac{(5 \cdot 3) + (2 \cdot 1) + (4 \cdot 2)}{\sqrt{(5^2 + 2^2 + 4^2)} \cdot \sqrt{14}} = \frac{15 + 2 + 8}{\sqrt{45} \cdot \sqrt{14}} = \frac{25}{\sqrt{630}} \approx 0.996$$
Step B: Prediction for U1 on Item D
U1's known ratings: A=5, B=3, C=4
$$\hat{r}_{U1,D} = \frac{(0.996 \cdot 5) + (0.786 \cdot 3) + (0.996 \cdot 4)}{0.996 + 0.786 + 0.996}$$
$$= \frac{(4.98 + 2.36 + 3.98)}{2.778} = \frac{11.32}{2.778} \approx 4.07$$
So, U1's predicted rating for Item D = 4.07 (likely to be rated as 4).
</td>
</tr>
<tr>
<td>9</td>
<td>Write a note on neighborhood selection strategies in CF.</td>
<td>Neighborhood selection involves choosing the set of similar users/items that influence prediction.
Strategies: - Fixed K-nearest neighbors: Select exactly K most similar neighbors. - Similarity Thresholding: Select all neighbors with similarity ≥ threshold. - Hybrid Strategy: Combine K-nearest with thresholding. Factors affecting selection: - Dataset sparsity. - Size of K (too small = poor predictions, too large = noisy). Example: For K=3, if similarities are [0.9,0.8,0.6,0.2], then top 3 are chosen. Impact: Neighborhood size directly affects recommendation quality.</td>
</tr>
<tr>
<td>10</td>
<td>Compare neighborhood-based CF with model-based CF.</td>
<td>Neighborhood-based CF: - Works by finding similar users/items. - Methods: User-based, Item-based. - Easy to implement and explain. - Slower in large datasets. Model-based CF: - Uses machine learning models to predict ratings (e.g., Matrix Factorization, SVD, Neural Networks). - Faster predictions once trained. - Requires training time but handles sparsity better. Comparison Table: - Accuracy: Model-based usually better. - Transparency: Neighborhood-based easier to interpret. - Scalability: Model-based more</td>
</tr>
</table>

		scalable. - **Example:** Netflix Prize winner solution used model-based (SVD + ensembles).

PART C

<table>
<tr><th>Q. No</th><th>Question</th><th>Answer</th></tr>
<tr><td>1</td><td>Explain in detail the working of user-based collaborative filtering with a systematic algorithm and example.</td><td>Introduction: User-based Collaborative Filtering (UBCF) is a memory-based approach that recommends items to a user based on ratings given by similar users. Steps in Algorithm: 1. Represent the user–item rating matrix. 2. Normalize ratings (optional, to remove user bias). 3. Compute similarity between target user and all other users using Pearson correlation or cosine similarity. 4. Select top-N similar users (neighborhood). 5. Predict missing ratings for target user using weighted average of neighbors' ratings. 6. Recommend top-K items with highest predicted ratings. Formula (Pearson correlation similarity):

Formula (Pearson correlation similarity):
$sim(u,v) = \frac{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)(r_{v,i} - \bar{r}_v)}{\sqrt{\sum_{i \in I_{uv}} (r_{u,i} - \bar{r}_u)^2} \sqrt{\sum_{i \in I_{uv}} (r_{v,i} - \bar{r}_v)^2}}$</td></tr>
<tr><td>2</td><td>Describe in detail the item-based collaborative filtering approach with algorithm, formulas, and example.</td><td>
<table>
<tr><th>Section</th><th>Explanation</th></tr>
<tr><td>Introduction</td><td>Item-Based Collaborative Filtering (IBCF) is a recommendation technique where the similarity is calculated between items (instead of users). The main assumption is that if a user likes an item, they will also like similar items. Unlike user-based CF, which compares users, IBCF computes the closeness between items using rating patterns from all users.</td></tr>
<tr><td>Steps of the Algorithm</td><td>1. Prepare the user–item rating matrix (rows = users, columns = items). 2. Compute similarity between items using measures like cosine similarity, Pearson correlation, or adjusted cosine similarity. 3. Identify top-k similar items for the target item. 4. Generate predictions by aggregating the ratings given by the active user to similar items. 5. Recommend the top-N items with the highest predicted scores.</td></tr>
<tr><td>Formulas</td><td>1. Cosine Similarity (item i, item j):
1. Cosine Similarity (item i, item j):
$\text{sim}(i,j) = \frac{\sum_{u \in U} r_{u,i} \cdot r_{u,j}}{\sqrt{\sum_{u \in U} r_{u,i}^2} \cdot \sqrt{\sum_{u \in U} r_{u,j}^2}}$

2. Adjusted Cosine Similarity (accounts for user rating bias):
$\text{sim}(i,j) = \frac{\sum_{u \in U} (r_{u,i} - \bar{r}_u)(r_{u,j} - \bar{r}_u)}{\sqrt{\sum_{u \in U} (r_{u,i} - \bar{r}_u)^2} \cdot \sqrt{\sum_{u \in U} (r_{u,j} - \bar{r}_u)^2}}$
where $r_{u,i}$ = rating of user u on item i, and $\bar{r}_u$ = average rating of user u.</td></tr>
</table>
</td></tr>
</table>

		Worked Example	Consider a rating matrix: Users vs Items (scale 1–5): • U1: I1=5, I2=4, I3=?, I4=2 • U2: I1=4, I2=?, I3=3, I4=1 • U3: I1=?, I2=2, I3=4, I4=5 **Step 1: Compute similarity** between I1 and I3 using cosine similarity: Ratings on I1: (5,4,?) → (5,4) Ratings on I3: (?,3,4) → (3,4) $sim(I1, I3) = \frac{(5\cdot3+4\cdot4)}{\sqrt{5^2+4^2}\cdot\sqrt{3^2+4^2}} = \frac{15+16}{\sqrt{41}\cdot5} = \frac{31}{\sqrt{41}\cdot5} \approx 0.97$ So, I1 and I3 are highly similar. **Step 2: Predict missing rating of U1 on I3.** Using neighbors (I1 and I2): Suppose similarity(I3,I1)=0.97, similarity(I3,I2)=0.6. U1's ratings: I1=5, I2=4. $\hat{r}_{U1,I3} = \frac{0.97\cdot5+0.6\cdot4}{0.97+0.6} = \frac{4.85+2.4}{1.57} \approx 4.6$ Thus, system predicts U1 will rate I3 ≈ 4.6 → strong recommendation.
		Advantages	- More stable than user-based CF (since items change less frequently than users). - Handles large-scale systems better. - Captures item–item relationships that remain consistent across users.
		Limitations	- Cold-start problem for new items (no ratings available). - Similarity computation can still be expensive for very large item sets.
		Conclusion	Item-Based CF is a robust method for generating recommendations by exploiting similarities between items. With formulas for similarity and prediction, and a worked-out numerical example, it provides accurate recommendations in real-world systems like Amazon (similar products).
3	**Discuss in depth the components of neighborhood methods in collaborative filtering: rating normalization, similarity computation, and neighborhood selection.**	**Introduction:** Neighborhood methods form the backbone of memory-based CF. They require three main components to improve accuracy and reliability of recommendations. **(i) Rating Normalization:** - Purpose: Remove individual user bias (some users rate strictly, some generously). - Methods: Mean-centering, Z-score normalization. - Formula: $r'_{u,i} = r_{u,i} - \bar{r}_u$. **(ii) Similarity Weight Computation:** - Measures how alike two users/items are. - Common measures: Cosine similarity, Pearson correlation, Jaccard similarity. - Example: Cosine similarity treats ratings as vectors. Pearson adjusts for rating scale bias. **(iii) Neighborhood Selection:** - Choose top-N most similar users/items. - Ensures predictions are based on strongest relationships. - Example: For user U1, select top 3 most similar users from similarity matrix. **Worked-out Example:** Construct a 3×3 user-item matrix, normalize ratings, compute cosine similarity, select nearest neighbors, then predict missing rating. **Advantages of Neighborhood Methods:** Intuitive, interpretable, effective for small datasets. **Limitations:** Struggle with sparsity, scalability, and cold-start. **Diagram:** Flowchart showing rating normalization → similarity computation → neighborhood selection → prediction.	

UNIT IV ATTACK-RESISTANT RECOMMENDER SYSTEMS

Introduction – Types of Attacks – Detecting attacks on recommender systems – Individual attack – Group attack – Strategies for robust recommender design - Robust recommendation algorithms.

Q.No	Question	Answer
1	What are attack-resistant recommender systems?	Attack-resistant recommender systems are designed to withstand malicious manipulations such as fake ratings or spam. They maintain the reliability of suggestions by detecting, preventing, or minimizing the effect of adversarial attacks, ensuring trustworthy recommendations for genuine users.
2	Define shilling attack in recommender systems.	A shilling attack refers to a situation where malicious users intentionally add fake ratings or profiles into the system. Their aim is to either **push** (promote) or **nuke** (demote) certain items, thereby biasing the recommendation results.
3	List two major types of attacks in recommender systems.	The two major types are: (i) **Individual attack** – performed by a single malicious user account. (ii) **Group attack** – coordinated by multiple attackers who inject similar fake profiles to maximize the manipulation impact.
4	Why are recommender systems vulnerable to attacks?	Recommender systems heavily depend on user-generated inputs such as ratings, reviews, and clicks. Since these inputs are open and easy to manipulate, malicious users can exploit this weakness to alter recommendation quality.
5	What is an individual attack?	In an individual attack, a single fake user profile is created and injected with biased ratings. Even though the impact is small compared to group attacks, it can still distort recommendations, especially in small datasets.
6	What is a group attack?	A group attack occurs when several attackers collaborate and inject multiple fake profiles with carefully designed rating patterns. This coordinated manipulation can significantly bias the system towards or against specific items.
7	Give examples of push and nuke attacks.	In a **push attack**, the attacker gives very high ratings to the target item so it gets recommended to more users. In a **nuke attack**, extremely low ratings are given to reduce the target item's ranking and visibility.
8	Mention two consequences of shilling attacks.	(i) It reduces user trust in the recommender system as users receive irrelevant or manipulated results. (ii) It leads to poor customer satisfaction, harming the platform's reputation and business.
9	What are camouflage attacks?	Camouflage attacks are advanced profile injection attacks where fake profiles contain both normal ratings (to appear genuine) and biased ratings (to promote/demote target items), making detection difficult.
10	Define robust recommender design.	Robust recommender design involves building algorithms that can tolerate malicious data without losing accuracy. Such systems balance **accuracy, security, and fairness**, ensuring recommendations remain reliable even under attack.
11	List two strategies for detecting attacks.	Common detection strategies include: (i) **Statistical anomaly detection**, which flags unusual patterns like extreme ratings. (ii) **Machine learning classifiers**, trained to distinguish genuine profiles from fake attackers.
12	Differentiate between white-box	**White-box attack**: Attacker knows the recommender's internal algorithm and similarity measures, allowing precise manipulation. **Black-box attack**:

	and black-box attacks.	Attacker does not know system details, but exploits it through repeated trial-and-error.
13	What is the role of anomaly detection in recommender systems?	Anomaly detection identifies suspicious rating patterns that deviate from normal user behavior. By catching such anomalies, the system can filter fake profiles before they influence recommendations.
14	State any two robust recommendation algorithms.	(i) **Bayesian Probabilistic Models** – detect uncertainty and noise caused by fake profiles. (ii) **Trust-based Recommenders** – rely on social trust networks to filter out unreliable or malicious users.
15	Define profile injection attack.	Profile injection attack means creating fake user accounts and filling them with artificially constructed ratings. These fake profiles are injected into the system to manipulate item popularity.
16	Mention one statistical feature used for attack detection.	One common statistical feature is **rating variance**. Attackers often rate all items with maximum or minimum values, resulting in abnormal variance compared to genuine user profiles.
17	Why is diversity important in robust recommender design?	Diversity ensures that recommendations are spread across a wide range of items rather than being concentrated on a few. This reduces the influence of manipulated profiles, making attacks less effective.
18	What is a hybrid defense strategy?	A hybrid defense strategy combines multiple defense methods, such as anomaly detection, clustering, trust modeling, and rating filters, to strengthen the system against both simple and complex attacks.
19	Explain the term "robustness vs. accuracy trade-off".	A highly robust system may reduce accuracy slightly to remain secure. On the other hand, focusing purely on accuracy without robustness makes the system vulnerable. Hence, designers must balance the two.
20	Give one real-world example of attack-resistant recommender use.	Platforms like **Amazon, Netflix, or Spotify** deploy attack-resistant mechanisms that detect fake reviews, spam ratings, or bot-generated clicks. This ensures that recommendations remain fair and relevant to users.

PART –B

Q.No	Question	Answer
1	Explain in detail the **types of attacks** on recommender systems with suitable examples.	Recommender systems are vulnerable to malicious users who try to manipulate the recommendations. The common types of attacks are: **(i) Random Attack** – Attackers give random ratings to non-target items and extreme ratings to the target item. E.g., giving all random ratings but 5 stars to a target movie. **(ii) Average Attack** – Attackers rate non-target items with average system ratings and give extreme ratings to target items. More effective than random. **(iii) Bandwagon Attack** – Attackers rate popular items highly to appear genuine, along with boosting the target item. **(iv) Segment Attack** – Attackers target a particular user segment by rating items from that category plus the target. **(v) Reverse Bandwagon Attack** – Attackers give low ratings to unpopular items to reduce target's rank.

		(vi) Love/Hate Attack – Simple strategy: only rate target item with maximum/minimum score. **Conclusion:** Each attack manipulates similarity computations and biases recommendation lists.
2	Describe **individual attack** and **group attack** in recommender systems with neat comparison.	**Individual Attack:** - Performed by a single malicious user. - Limited impact because only one profile contributes. - Easier to detect due to abnormal rating behavior. **Group Attack:** - Multiple colluding users create fake profiles. - High impact: can drastically alter recommendations. - Harder to detect as attackers mimic genuine diversity. **Example:** If 100 fake users give 5 stars to a product, the system may falsely recommend it widely. **Comparison Table:**
3	Discuss the **strategies for robust recommender design** to defend against attacks.	Designing attack-resistant recommender systems involves several strategies: **(i) Profile Analysis** – Detect abnormal user profiles (e.g., too many extreme ratings). **(ii) Outlier Detection** – Identify users whose ratings deviate from global mean. **(iii) Temporal Analysis** – Detect sudden spikes of similar ratings within a short time. **(iv) Trust Models** – Use reputation-based filtering where only trusted users influence recommendations. **(v) Hybrid Methods** – Combine content-based and collaborative filtering to cross-check results. **(vi) Robust Algorithms** – Weighted average, SVD with attack-resistance, Bayesian approaches. **Conclusion:** Robust design reduces manipulation risk and improves fairness, accuracy, and trustworthiness of recommendations.
4	Explain with example how **attack detection techniques** work in recommender systems.	Attack detection involves identifying malicious profiles before they impact results. Common techniques: **(i) Statistical Detection:** - Detect abnormal patterns (e.g., always max/min ratings). - Example: A user rates 200 items in one day with extreme values – flagged as suspicious. **(ii) Machine Learning Approaches:** - Classify profiles as genuine or fake using classifiers (SVM, Random Forest). - Features: rating variance, deviation, length of profile. **(iii) Similarity Check:** - Malicious users often have highly similar rating patterns. **(iv) Time-based Analysis:** - Fake users usually created at the same time → detection via timestamp clustering. **Example:** A group of 50 users rate an obscure book with 5 stars within 24 hours – flagged as group attack.
5	Write a note on **robust recommendation algorithms** with examples.	Robust recommendation algorithms aim to minimize attack impact. Common techniques: **(i) Weighted kNN:** Assign less weight to suspicious profiles in collaborative filtering. **(ii) SVD-based Robust Methods:** Dimensionality reduction filters out noise and fake profiles. **(iii) Bayesian Models:** Use probabilistic approaches to detect abnormal user behavior. **(iv) Trust-based Models:** Rely on social trust networks, reducing reliance on fake accounts.

		(v) Graph-based Models: Identify clusters of suspicious users. **Example:** Amazon uses hybrid + trust-based models to detect fake reviews and prevent product boosting.
6	Differentiate between **attack-resistant and normal recommender systems**.	**Normal Recommender Systems:** - Focus mainly on accuracy. - Vulnerable to malicious manipulation. - Example: Simple user-based collaborative filtering. **Attack-Resistant Recommender Systems:** - Focus on accuracy + robustness. - Include defense strategies like anomaly detection, trust models. - Example: Hybrid trust-aware collaborative filtering. **Comparison:**
7	With neat sketch, explain the **architecture of an attack-resistant recommender system**.	Architecture typically has the following modules: **(i) Data Collection:** User ratings, profiles, logs. **(ii) Pre-processing:** Cleaning, noise removal. **(iii) Attack Detection:** Identify suspicious profiles via statistical or ML models. **(iv) Robust Filtering Algorithm:** Collaborative/Content/Hybrid with resistance mechanisms. **(v) Recommendation Engine:** Generates final recommendation list after excluding suspicious profiles. **(vi) Feedback & Monitoring:** Continuously track suspicious activities. **Diagram:** (Draw flowchart – Users → Data → Attack Detection → Robust Filtering → Recommendations). **Conclusion:** This architecture ensures both **accuracy** and **security** of recommendations.
8	What is **bandwagon attack**? How can it be detected and prevented?	**Definition:** Bandwagon attack occurs when attackers rate a target item along with highly popular items to appear genuine. **Example:** Fake profiles give 5 stars to "Avengers: Endgame" + target movie "XYZ". **Detection:** - Identify profiles with unusual correlation to highly popular items. - Use similarity clustering: fake profiles often have high overlap with top-N popular items. **Prevention:** - Weight popular items less in similarity computation. - Apply trust/reputation-based filtering. **Conclusion:** Bandwagon attacks exploit popularity bias; robust algorithms reduce their impact.
9	Explain **group shilling attack** with an example.	**Group Shilling Attack:** Multiple attackers create fake user accounts and collude to promote/demote an item. **Example:** 100 fake users all give 5 stars to a new mobile phone → system wrongly recommends it widely. **Features of Group Attack:** - Coordinated, large-scale. - Harder to detect. - Mimics genuine diversity. **Detection:** - Clustering analysis of rating patterns. - Time-based monitoring (sudden spikes). **Conclusion:** Group shilling attacks are **most dangerous** and require strong anomaly detection + trust-based algorithms.
10	Discuss the role of **machine learning** in building attack-resistant recommender systems.	Machine Learning plays a vital role in attack detection and prevention: **(i) Classification Models:** (SVM, Random Forest) – Classify profiles as fake/genuine using features like variance, deviation. **(ii) Clustering Models:** Detect suspicious groups with highly similar behavior. **(iii) Deep Learning:** Capture hidden patterns of shilling attacks in large datasets. **(iv) Ensemble Methods:** Combine multiple ML models for stronger detection. **(v) Reinforcement Learning:** Continuously adapt RS to new attack patterns. **Example:** Netflix uses ML anomaly detection to prevent fake account-based manipulation. **Conclusion:** ML increases robustness, adaptability, and scalability of recommender systems.

PART –C

Q. No	Question	Answer
1	**Explain in detail the types of attacks on recommender systems. Illustrate with examples.**	Recommender systems are vulnerable to various malicious manipulations. Attackers inject fake profiles or ratings into the system to bias recommendations. The main **types of attacks** are: **1. Random Attack:** The attacker assigns random ratings to items, except the target item, which gets an extreme rating (high for push, low for nuke). **Example:** In a movie recommender, an attacker gives random ratings to many movies but rates a particular movie (say, *Movie X*) as 5 stars to push it up. **2. Average Attack:** Instead of random ratings, attackers give ratings close to the average item rating for filler items, making the profile look more genuine. **Example:** If the average rating for *Movie Y* is 3.2, the attacker gives ~3 to all movies but 5 to the target. **3. Bandwagon Attack:** Attackers exploit popular items. They rate popular items positively along with the target item, making their profile blend with real users. **Example:** Rating blockbuster films highly (Avengers, Avatar) plus 5 stars to the target movie. **4. Reverse Bandwagon Attack:** Instead of popular items, attackers use unpopular items negatively, thereby making their fake profiles unique but effective. **5. Love-Hate Attack:** Attacker rates one set of items with the lowest rating and the target item with the highest rating (or vice versa). **6. Group Attacks:** Multiple fake profiles are created, all coordinated to boost or demote an item. This increases attack success probability. **Conclusion:** These attacks exploit the open nature of recommender systems, reducing trust and degrading recommendation quality. Detecting and mitigating them is crucial for system robustness.
2	**Describe the techniques for detecting attacks in recommender systems. How can systems differentiate genuine users from attackers?**	**Attack detection** ensures recommender systems remain robust. The detection methods include: **1. Statistical Analysis of Profiles:** Fake profiles often show unusual rating patterns. For example, rating variance may be too low (all items rated around the same value) or too high. **2. Rating Distribution Check:** Attackers often rate only a small subset of items (target + filler). By checking rating density and filler size, anomalies can be spotted. **3. Similarity Analysis:** Attack profiles are often highly similar to each other (in group attacks). Computing pairwise similarity between user profiles can reveal coordinated behaviors. **4. Temporal Analysis:** Attackers typically inject many ratings in a short time, unlike genuine users who rate gradually. Time-based monitoring helps detect bursts of suspicious activity. **5. Item Popularity Correlation:** Attack profiles often manipulate either very popular or very unpopular items. If abnormal spikes in rating patterns are seen, they may indicate manipulation. **6. Outlier Detection (Machine Learning):** Classifiers like SVM, Random Forest, or clustering techniques can separate normal vs. attack profiles. Features include mean rating, variance, rating deviation, etc. **Conclusion:** By combining **statistical, temporal, and machine learning**

		methods, recommender systems can efficiently identify and reduce the impact of malicious profiles.
3	**Discuss robust recommendation algorithms and strategies for designing attack-resistant recommender systems with examples.**	To ensure security, recommender systems adopt **robust algorithms** and **defensive strategies**: **Strategies for Robust Design:** 1. **User Authentication:** Restricting profile creation with CAPTCHA, verified email, or mobile number reduces fake accounts. 2. **Profile Weighting:** Assign lower weights to suspicious profiles (new users with extreme ratings). 3. **Trust-Based Systems:** Recommendations are generated based on trusted social connections or verified ratings, reducing fake influence. 4. **Diversity Promotion:** Avoid over-reliance on a few ratings by increasing diversity in recommendations. **Robust Recommendation Algorithms:** 1. **Robust Matrix Factorization:** Introduces constraints during factorization to reduce the impact of noisy or malicious ratings. 2. **Bayesian Models:** Use probabilistic methods to detect suspicious deviations in ratings. 3. **Graph-Based Methods:** Represent users and items as graphs; attackers are detected as abnormal nodes or communities. 4. **Hybrid Approaches:** Combine collaborative filtering with content-based methods to cross-check suspicious inputs. **Example:** Suppose an attacker rates *Movie X* as 5 stars and filler movies with random ratings. A robust algorithm identifies the unusual rating variance and down-weights that user's influence, preventing *Movie X* from being over-recommended. **Conclusion:** By integrating **robust algorithms** with **preventive strategies**, recommender systems can balance recommendation accuracy with security, ensuring fairness and trustworthiness.

UNIT V EVALUATING RECOMMENDER SYSTEMS

Evaluating Paradigms – User Studies – Online and Offline evaluation – Goals of evaluation design – Design Issues – Accuracy metrics – Limitations of Evaluation measures

PART- A

Q.No	Question	Answer (Elaborated – 3 to 5 lines)
1	What is the purpose of evaluating recommender systems?	The main purpose of evaluating recommender systems is to measure their effectiveness in providing useful, relevant, and accurate recommendations. Evaluation helps compare algorithms, identify strengths/weaknesses, and ensure the system meets user expectations and business goals.
2	Define evaluation paradigms in recommender systems.	Evaluation paradigms are frameworks or approaches used to assess recommender systems. The three common paradigms are **offline evaluation** (using historical datasets), **online evaluation** (A/B testing with real users), and **user studies** (controlled lab experiments).
3	Differentiate between offline and online evaluation.	**Offline evaluation** uses pre-collected datasets and simulates recommendation results, while **online evaluation** tests the system with real users in real-time. Offline is faster but less realistic; online is more accurate but costly and risky.

4	What is a user study in evaluation?	A user study involves testing the recommender system with a group of real users in a controlled environment. Users provide feedback through tasks, surveys, or interviews, which helps measure subjective qualities like satisfaction, trust, and usability.
5	List the main goals of evaluation design.	The primary goals include: (1) measuring accuracy, (2) ensuring diversity, (3) improving coverage, (4) increasing novelty and serendipity, (5) guaranteeing fairness, and (6) maintaining user satisfaction and trust in recommendations.
6	Define accuracy in recommender evaluation.	Accuracy measures how close the predicted ratings or recommended items are to the user's true preferences. It is commonly measured using metrics like RMSE, MAE, Precision, Recall, and F1-Score.
7	What is RMSE?	RMSE (Root Mean Square Error) is an accuracy metric that measures the square root of the average squared difference between predicted ratings and actual ratings. It penalizes large errors more heavily. Formula: $RMSE = \sqrt{\frac{1}{n}\sum(r_{ui} - \hat{r}_{ui})^2}$.
8	Differentiate between Precision and Recall.	**Precision** measures how many recommended items are relevant, while **Recall** measures how many relevant items were recommended out of all relevant items available. Precision = relevance of results, Recall = completeness of results.
9	Define F1-Score in recommender evaluation.	F1-score is the harmonic mean of Precision and Recall, balancing both metrics into one measure. It is useful when there is an uneven distribution of relevant and irrelevant items. Formula: $: F1 = \frac{2 \times P \times R}{P+R}$.
10	What is Coverage in evaluation metrics?	Coverage measures the percentage of items or users for which the system can generate recommendations. It ensures the recommender does not only suggest a small set of popular items but covers a broad range.
11	Explain novelty in recommender system evaluation.	Novelty refers to how new or unexpected the recommendations are to the user. A system should not only suggest popular items but also expose users to lesser-known or unique content.
12	What is serendipity in recommender evaluation?	Serendipity measures the ability of a system to recommend surprising yet useful items that a user would not have discovered on their own. It improves user engagement and satisfaction.
13	Define robustness in recommender evaluation.	Robustness evaluates how well a recommender system can resist noise, attacks, or manipulated ratings. A robust system should provide consistent results even under adversarial conditions.
14	What are design issues in evaluation?	Evaluation design issues include dataset bias, cold-start problems, choice of metrics, scalability concerns, user diversity, and the balance between accuracy and novelty. Poor design can mislead evaluation results.
15	What are limitations of offline evaluation?	Offline evaluation is faster and cheaper but may not capture real user behavior. It ignores aspects like user satisfaction, long-term engagement, and contextual factors (e.g., time, mood).
16	Why are online evaluations considered more reliable?	Online evaluations, such as A/B testing, directly involve real users and measure actual system performance in a live environment. This makes them more realistic but also more resource-intensive.
17	What is A/B testing in recommender evaluation?	A/B testing is an online evaluation method where users are split into two groups: one uses the new recommender algorithm (A), and the other uses the baseline (B). Their interactions are compared to decide which performs better.

18	Differentiate between objective and subjective evaluation.	**Objective evaluation** uses numerical metrics like RMSE, Precision, and Recall, while **subjective evaluation** gathers user opinions on satisfaction, trust, and usability through surveys and interviews.
19	What is the cold-start problem in evaluation?	Cold-start occurs when the recommender has insufficient data about new users or new items, making it difficult to evaluate performance accurately. It affects accuracy and coverage.
20	Why is it important to combine multiple evaluation metrics?	A single metric cannot capture all aspects of recommendation quality. Combining multiple metrics (accuracy, diversity, novelty, serendipity) ensures a holistic and fair evaluation of the system.

PART - B

Q. No	Question	Answer (Elaborated University Style)
1	Explain the different **evaluation paradigms** used in recommender systems.	**Answer:** Evaluation paradigms are systematic approaches to assess recommender performance. The three major paradigms are: (i) **User Studies** – involve real users interacting with the system. Example: Asking participants to rate recommendations based on satisfaction and usefulness. Advantages: measures actual satisfaction. Disadvantages: costly, time-consuming. (ii) **Online Evaluation (A/B testing)** – deployed system is tested with real users. Group A gets current version, Group B gets modified system. Metrics: CTR (Click Through Rate), Conversion Rate. Advantages: measures real-time impact, scalable. (iii) **Offline Evaluation** – uses pre-collected datasets. Training/test split is applied, and accuracy metrics like RMSE, MAE, Precision, Recall are computed. Advantages: fast, reproducible. Disadvantages: does not capture actual user satisfaction. **Conclusion:** Choice depends on goal (accuracy vs. satisfaction vs. scalability).
2	Describe the role of **user studies** in evaluating recommender systems.	**Answer:** User studies directly involve participants to assess recommendation quality. Process: (i) **Recruitment of participants** – selecting a group representing target users. (ii) **Experiment design** – exposing users to different recommendation methods. (iii) **Evaluation criteria** – asking for ratings on relevance, novelty, serendipity, trust, and overall satisfaction. (iv) **Qualitative feedback** – open-ended responses provide insights into perceived strengths and weaknesses. (v) **Limitations** – expensive, time-consuming, small sample sizes reduce generalizability. **Example:** MovieLens frequently conducts user studies to test usability and recommendation acceptance.
3	Differentiate between **online and offline evaluation** of recommender systems with examples.	**Answer: Online Evaluation:** Conducted on live systems with real users. Techniques: A/B testing, multi-armed bandits. Example: Amazon testing new recommendation ranking on a subset of customers. Metrics: CTR, Conversion Rate. Pros: Realistic, captures behavioral impact. Cons: costly, requires deployment. **Offline Evaluation:** Performed on static datasets without involving real users. Method: Train-test split, cross-validation. Metrics: RMSE, MAE, Precision, Recall, F1. Example: Using MovieLens dataset to benchmark collaborative filtering models. Pros: Fast, repeatable. Cons: Doesn't measure true satisfaction. **Conclusion:** Online = "real-world impact"; Offline = "algorithmic accuracy."
4	Write about the **goals of evaluation design** in	**Answer:** The primary goals include: (i) **Accuracy** – ensuring recommended items match user interests (e.g., RMSE, Precision). (ii) **User satisfaction** – recommendations should be useful, novel, serendipitous. (iii) **Diversity &**

	recommender systems.	**Coverage** – system should recommend varied items, not repetitive or popular-only. (iv) **Trust & Transparency** – users must feel confident and understand why items are recommended. (v) **Business goals** – maximizing sales, engagement, or retention. (vi) **Scalability & Robustness** – system must work for millions of users without being vulnerable to attacks. **Conclusion:** Evaluation must balance technical accuracy with human-centered factors.
5	Discuss the **design issues** in recommender system evaluation.	**Answer:** Design issues arise while structuring evaluation. Key aspects: (i) **Choice of dataset** – public benchmark datasets (MovieLens, Netflix Prize) vs. proprietary real-world datasets. (ii) **Data sparsity** – limited ratings affect accuracy. (iii) **Cold start problem** – new users/items lack data. (iv) **Temporal dynamics** – user preferences evolve over time, affecting reliability. (v) **Evaluation metrics** – choosing RMSE vs. Precision vs. Diversity influences interpretation. (vi) **Sampling bias** – system logs may not represent all user behavior. (vii) **Practical constraints** – cost, time, infrastructure limit testing. **Conclusion:** Proper design ensures fair, meaningful evaluation.
6	Elaborate on the **accuracy metrics** used for recommender system evaluation.	**Answer:** Accuracy measures how close predicted ratings/recommendations are to actual preferences. Common metrics: (i) **RMSE (Root Mean Squared Error):** Penalizes large deviations heavily. Formula: RMSE= $RMSE = \sqrt{\frac{1}{N}\sum(r_{ui} - \hat{r}_{ui})^2}$. (ii) **MAE (Mean Absolute Error):** Measures average deviation. Formula: $\text{MAE} = \frac{1}{n}\sum_{i=1}^{n} \lvert y_i - \hat{y}_i \rvert$
7	Discuss the **limitations of evaluation measures** in recommender systems.	**Answer:** Limitations include: (i) **Overemphasis on accuracy** – ignores novelty, serendipity, and diversity. (ii) **Offline evaluation gap** – real user behavior may differ from dataset assumptions. (iii) **Context ignorance** – evaluations often ignore time, location, and mood. (iv) **Short-term vs. long-term effects** – accuracy today may not mean sustained user engagement. (v) **Bias in datasets** – historical logs reflect popularity bias, reinforcing mainstream items. (vi) **Scalability concerns** – metrics may not capture performance under real-world load. (vii) **Subjective satisfaction** – user happiness cannot be fully measured by numerical metrics. **Conclusion:** Evaluation must combine multiple metrics for holistic assessment.
8	Compare and contrast **accuracy metrics vs. beyond-accuracy metrics**.	**Answer: Accuracy Metrics:** Focus on prediction correctness. Examples: RMSE, MAE, Precision, Recall. Advantage: Easy to compute and compare. Limitation: Do not reflect real-world usefulness. **Beyond-Accuracy Metrics:** Capture aspects like diversity, novelty, serendipity, trust, and coverage. Example: Novelty = recommending unseen items; Diversity = recommending different item categories. Advantage: Closer to real satisfaction. Limitation: Harder to measure quantitatively. **Conclusion:** Both are complementary – accuracy ensures correctness, beyond-accuracy ensures satisfaction and engagement.
9	Explain with example how **offline evaluation** is carried out using accuracy metrics.	**Answer:** Steps: (i) **Dataset split** into training and testing sets. (ii) Train model (e.g., Matrix Factorization on MovieLens). (iii) Predict ratings for test set items. (iv) Compute metrics: RMSE, MAE, Precision, Recall. Example: If true rating of a movie is 4 and predicted is 3, error = 1. After aggregating across users, compute RMSE or MAE. (v) Report average values to compare algorithms. **Example Result:** Matrix Factorization: RMSE=0.91, User-based

		CF: RMSE=1.02. **Conclusion:** Offline evaluation provides fast benchmarking for algorithms.
10	Summarize the **challenges in recommender system evaluation**.	**Answer:** Key challenges include: (i) **Balancing accuracy vs. novelty** – accurate but repetitive recommendations frustrate users. (ii) **Cold start** – new users/items reduce reliability of evaluation. (iii) **Temporal effects** – users' interests change, old data may mislead. (iv) **Data sparsity** – very few ratings make evaluation difficult. (v) **Biases in data** – popularity bias, exposure bias influence results. (vi) **Scalability** – large datasets demand efficient evaluation techniques. (vii) **Measuring true satisfaction** – beyond clicks, long-term loyalty is hard to quantify. **Conclusion:** A hybrid evaluation approach (offline + online + user studies) best addresses these challenges.

PART-C

Q.No	Question	Elaborated Answer (University Style)
1	Explain in detail the **evaluation paradigms** used in recommender systems with examples.	**Evaluation paradigms** define the approaches by which recommender systems are tested and validated. The three main paradigms are: **1. Offline Evaluation**: Uses pre-collected datasets where user-item interactions (ratings, clicks, purchases) are available. Algorithms are tested by splitting data into **training and testing sets** (e.g., 80-20 split). Metrics like **MAE, RMSE, Precision, Recall, MAP** are applied. *Example*: Using the MovieLens dataset to predict ratings. **2. Online Evaluation**: Conducted on a live system where real users interact with recommendations. Uses **A/B testing, interleaving** to compare algorithms. *Example*: Netflix recommending movies in real time and comparing CTR (Click-Through Rate) of two algorithms. **3. User Studies**: Involves controlled experiments with human participants. Users evaluate the quality of recommendations based on **subjective feedback** (e.g., satisfaction, novelty, diversity). *Example*: Asking users to rate usefulness of book recommendations. **Conclusion**: Offline evaluation is faster but may lack realism, online evaluation is more accurate but expensive, and user studies provide insights into **user satisfaction**. A robust system often combines all three paradigms.
2	Discuss the **goals of evaluation design** in recommender systems. Why are they important?	The **goals of evaluation design** determine what aspects of the recommender system need to be measured for effectiveness. They include: **1. Accuracy**: Ensuring recommendations closely match user preferences (e.g., RMSE, Precision, Recall). **2. Diversity**: Providing varied items to avoid redundancy (e.g., recommending different genres of movies). **3. Novelty**: Suggesting items the user has not seen before, increasing system usefulness. **4. Serendipity**: Recommending unexpected but interesting items, enhancing surprise value. **5. Coverage**: Percentage of items/users for which recommendations can be generated. **6. Trust & Transparency**: Building confidence in system outputs by offering explanations. **7. Scalability & Efficiency**: Ensuring system works for millions of users/items. **Importance**: Evaluation design goals ensure that a recommender system is not only accurate but also **usable, fair, and user-friendly**. For example, Netflix balances accuracy with diversity and novelty to maintain **long-term user engagement**.
3	Write an essay on the **limitations of evaluation**	Evaluation metrics, while useful, have several limitations: **1. Over-reliance on Accuracy Metrics**: Metrics like RMSE focus only on numerical prediction error but fail to capture **user satisfaction**. *Example*: A system predicting a

	measures in recommender systems with suitable examples.	rating of 4.2 instead of 4.0 has low error, but the recommendation may still be irrelevant to the user. **2. Lack of Context Awareness**: Metrics often ignore **temporal, situational, or location-based context**. A restaurant recommended at midnight might not be useful. **3. Offline vs. Online Gap**: Offline evaluation may show high accuracy, but real users may not click or engage. *Example*: Amazon may predict books correctly but users may ignore due to lack of novelty. **4. Subjectivity in User Studies**: Human feedback can be **biased or inconsistent**, making evaluations unreliable. **5. Scalability Issues**: Running live evaluations (A/B testing) is costly for large platforms. **6. Ignoring Long-Term Impact**: Metrics measure **short-term accuracy**, not long-term goals like **customer loyalty or churn reduction**. **Conclusion**: Evaluation measures must be **multi-dimensional**, combining accuracy, novelty, diversity, and long-term effects. A hybrid evaluation framework is essential to overcome these limitations.