General Goals of Evaluation Design: Accuracy, Coverage, Confidence and Trust, Novelty, Serendipity, Diversity, Scalability, Segmenting the Ratings for Training and Testing, Accuracy Metrics in Offline Evaluation.

General Goals of Evaluation Design in Recommender Systems

- Recommender systems are evaluated based on several key factors beyond just accuracy. Below are the main evaluation goals:

Accuracy

- Measures how well a recommender system predicts ratings or rankings.
- **Entry-specific error:** $e_{uj} = \hat{r}_{uj} - r_{uj}$ (difference between predicted and actual rating).
- **Common accuracy metrics:**
  - **Mean Squared Error (MSE):** Measures squared differences.

$$MSE = \frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}$$

  - **Root Mean Squared Error (RMSE):** Square root of MSE.

$$RMSE = \sqrt{\frac{\sum_{(u,j)\in E} e_{uj}^2}{|E|}}$$

  - **Accuracy of rankings:** Uses rank correlation and utility-based measures.

$$R = \begin{bmatrix} 5 & 3 & ? \\ 4 & ? & 1 \\ ? & 2 & 3 \end{bmatrix}$$

$$\hat{R} = \begin{bmatrix} 4.8 & 3.2 & 4.2 \\ 3.9 & 3.1 & 1.2 \\ 3.8 & 2.1 & 3.1 \end{bmatrix}$$

For observed values, we calculate the error:

$$e_{uj} = \hat{r}_{uj} - r_{uj}$$

Extracting observed values:

$$e = \begin{bmatrix} (4.8-5) & (3.2-3) & - \\ (3.9-4) & - & (1.2-1) \\ - & (2.1-2) & (3.1-3) \end{bmatrix}$$

$$e = \begin{bmatrix} -0.2 & 0.2 & - \\ -0.1 & - & 0.2 \\ - & 0.1 & 0.1 \end{bmatrix}$$

- **Mean Squared Error (MSE):**

$$MSE = \frac{1}{|E|} \sum_{(u,j)\in E} e_{uj}^2$$

$$MSE = \frac{(-0.2)^2 + (0.2)^2 + (-0.1)^2 + (0.2)^2 + (0.1)^2 + (0.1)^2}{6}$$

$$MSE = 0.025$$

- **Root Mean Squared Error (RMSE):**

$$RMSE = \sqrt{MSE} = \sqrt{0.025} = 0.1581$$

Note: Lower MSE and RMSE indicate better prediction accuracy.

Coverage

- Measures the proportion of users or items for which meaningful recommendations can be made.
- **User-space coverage**: Fraction of users for whom at least **k** ratings can be predicted.
- **Item-space coverage**: Fraction of items that can be recommended to users.
- **Catalog coverage**: Fraction of unique items appearing in at least one recommendation list.

$$CC = \frac{|\bigcup_{u=1}^{m} T_u|}{n}$$

Confidence and Trust

- **Confidence**: System's certainty in its predictions (e.g., providing confidence intervals).
- **Trust**: User's faith in the system, which can be increased with explanations.
- **Comparison of confidence estimates**:
  - If two systems provide 95% confidence intervals, the one with a **smaller width** is better.
  - Cannot compare confidence intervals if different levels (e.g., 95% vs. 99%) are used.

$$R = \begin{bmatrix} 5 & 3 & ? & 4 & ? \\ 4 & ? & 1 & ? & 2 \\ ? & 2 & 3 & 5 & ? \\ 1 & ? & 4 & 3 & 5 \end{bmatrix}$$

- The "?" represents missing ratings, meaning the user has not rated that item.
- Our goal is to evaluate **coverage, confidence, and trust** metrics.

1. User-Space Coverage

- User-space coverage refers to the fraction of users for whom at least k ratings can be predicted.
- Assume that the recommendation system can predict missing values based on collaborative filtering.
- If we set $k = 3$, and check which users have at least 3 ratings predicted:

User	Predicted Ratings ≥ 3
User 1	✅
User 2	✅
User 3	✅
User 4	✅

- **User-Space Coverage** = $\frac{4}{4} = 100\%$

2. Item-Space Coverage

- Item-space coverage is the fraction of items for which ratings of at least k users can be predicted.
- If we set $k = 2$, we check which items can be rated by at least two users:

Item	Predicted Ratings ≥ 2
Item 1	✅
Item 2	✅
Item 3	✅
Item 4	✅
Item 5	✅

- **Item-Space Coverage** = $\frac{5}{5} = 100\%$

3. Catalog Coverage

- This metric is calculated as:

$$CC = \frac{|\bigcup_{u=1}^{m} T_u|}{n}$$

where T_u is the list of top k recommended items for each user.

- Suppose we generate the following top-3 recommended lists for each user:

User	Top-3 Recommended Items
User 1	{2, 3, 5}
User 2	{1, 4, 5}
User 3	{1, 2, 3}
User 4	{2, 3, 4}

- The total unique recommended items:

$$\{1, 2, 3, 4, 5\}$$

- **Catalog Coverage** = $\frac{5}{5} = 100\%$ (since all 5 items appear in at least one recommendation list)

Step 3: Confidence Estimation

- Confidence represents the system's certainty in its recommendations.
- Suppose we estimate confidence intervals for predicted ratings:

User	Item	Predicted Rating $\hat{r}$	95% CI
1	3	4.2	(3.8, 4.6)
2	2	3.1	(2.7, 3.5)
3	1	3.8	(3.4, 4.2)

- **Comparison of Confidence:**
  - If two systems provide 95% confidence intervals, the system with the smaller width is considered more reliable.
  - Example: System A has CI width of **0.8**, while System B has CI width of **0.6**. System B is better.

Step 4: Trust Estimation

- Trust is the user's faith in the system.
- If explanations are provided (e.g., "You might like this because you rated a similar item"), users may trust recommendations more.
- A user survey could be used to measure trust:

User	Do you trust the recommendations? (Yes/No)
1	Yes
2	No
3	Yes
4	Yes

- **Trust Score** = $\frac{3}{4} = 75\%$

Novelty

- Measures the system's ability to recommend items that the user has **never seen before**.
- **Evaluation methods:**
  - **Online experiments**: Users are asked if they were previously aware of an item.
  - **Offline approximation**:
    - Hide ratings after a certain timestamp t_0.
    - Penalize recommendations for items rated before t_0 and reward those rated after.

Serendipity

- Measures the **surprise factor** in recommendations.
- **Difference from novelty**:
  - Novel items = Items the user has never seen.
  - **Serendipity** = **Unexpected but useful items.**
- **Example**:
  - A user who likes **Indian food** gets a recommendation for **Pakistani food** (novel but not surprising).
  - A recommendation for **Ethiopian food** is **serendipitous.**
- **Evaluation methods:**
  - **Online**: Ask users if recommendations are both useful and unexpected.
  - **Offline**: Compare against "obvious" recommendations from a simple content-based model.

- # Diversity

- Ensures that the recommendations are **varied** across genres, styles, etc.
- Example: A user gets **three movie recommendations**:
  - **Low diversity**: All are horror movies.
  - **High diversity**: Horror, comedy, and action.
- **Measurement**:
  - Compute **pairwise similarity** between items in the recommendation list.
  - Lower average similarity = Higher diversity.

- # Robustness and Stability

- A recommender system should remain stable when exposed to:
  - **Fake ratings** (spam attacks, biased ratings).
  - **Evolving user preferences** (changing trends over time).
- **Example**:
  - A book's author might enter fake **positive** ratings.
  - A competitor might add fake **negative** ratings.
- **Evaluation**:
  - Measure how much recommendations change when fake ratings are added.

Scalability

- Recommender systems should handle **large datasets efficiently**.
- **Key scalability measures:**

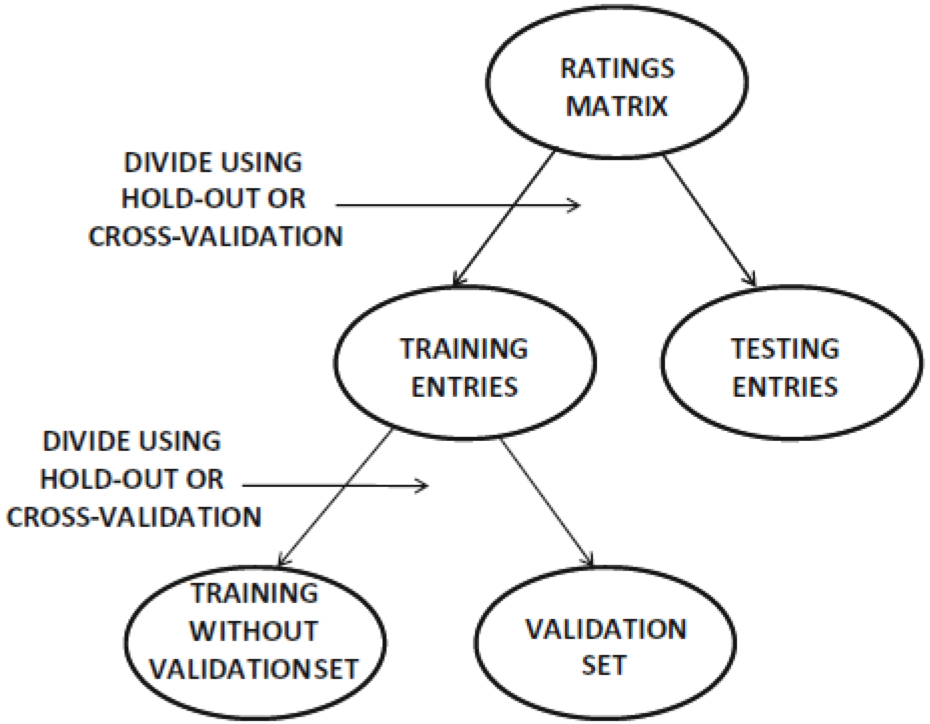
1. **Training time:** Should be reasonable (hours, not days).
2. **Prediction time:** Should be **low-latency** for real-time recommendations.
3. **Memory requirements:** System should handle large rating matrices without excessive RAM usage.

Segmenting the Ratings for Training and Testing in Recommender Systems

- Recommender systems require **training, validation, and test** datasets for evaluation.
- Since real-world datasets are not pre-partitioned, they must be split systematically.
- The two most common methods are **hold-out** and **cross-validation**.

1. Hierarchical Division of Ratings Data

- The dataset is first split into **training** and **testing** sets.
- The **training set** is further divided into:
  - **Model-building data**
  - **Validation set** (optional)
- This helps evaluate models without overfitting.



2. Hold-Out Method

- A **fraction of the dataset** is hidden (test set), and the rest is used for **training**.
- Accuracy is measured on the **held-out test set**.
- Prevents **overfitting** since the model doesn't see test data during training.
- **Drawbacks:**
  - **Underestimates true accuracy** (as all data is not used for training).
  - **Bias issue:** If the hidden ratings are **higher than the dataset's average**, accuracy evaluation is **pessimistic**.

3. Cross-Validation Method

- The dataset is divided into **q equal sets.**
- Each fold is used as the **test set** once, while the remaining $q-1$ folds are used for **training**.
- Accuracy is averaged across all **q experiments.**
- **Special Case: Leave-One-Out Cross-Validation (LOO-CV)**
  - Uses all data except **one entry** for training.
  - Evaluates performance for **each missing rating individually**.
  - **Downside:** Expensive when dataset is large (requires **|S| training runs**).

4. Comparison with Classification Design

- Collaborative filtering is similar to **classification**, as both **predict missing values**.
- **Key difference:**
  - **Classification:** Splits data **row-wise** (some users for training, some for testing).
  - **Collaborative Filtering:** Splits data **entry-wise** (some ratings for training, some for testing).
- **Challenge:** Sample selection bias.
  - Hidden ratings are from **items users chose to consume**, meaning they are **higher than randomly missing ratings.**
  - Results in **optimistic performance estimates** compared to real-world settings.

Accuracy Metrics in Offline Evaluation

- Offline evaluation measures a recommendation system’s performance using **historical data** before real-world deployment.
- It helps refine algorithms without exposing users to poor recommendations.

Two major categories of accuracy metrics:

- **Prediction accuracy metrics**: Focus on how close predicted ratings are to actual ratings (e.g., RMSE, MAE).
- **Ranking accuracy metrics**: Evaluate how well a model ranks relevant items for users.

Measuring the Accuracy of Ratings Prediction

- Many recommender systems predict numerical ratings (e.g., Netflix ratings from 1-5 stars).
- The quality of these predictions is assessed using metrics like RMSE and MAE.

RMSE versus MAE

1. Root Mean Square Error (RMSE)

- RMSE measures how much the predicted ratings deviate from actual ratings.
- It **squares the errors**, meaning **larger errors are penalized more heavily** than smaller ones.
- RMSE is more sensitive to **outliers** (large prediction mistakes).

📌 **Formula:**

$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(r_i - \hat{r}_i)^2}$$

where:

- N = total number of ratings
- r_i = actual rating given by the user
- $\hat{r}_i$ = predicted rating

2. Mean Absolute Error (MAE)

- MAE measures the **absolute** differences between predicted and actual ratings.
- It treats all errors equally (unlike RMSE, which penalizes large errors more).

📌 Formula:

$$MAE = \frac{1}{N}\sum_{i=1}^{N}|r_i - \hat{r}_i|$$

Impact of the Long Tail

What is the Long Tail?

- In recommender systems, a **few popular items** (head) get **most of the engagement**, while **many less popular items** (long tail) have **very few interactions**.
- Example: On Netflix, a few blockbuster movies dominate while thousands of niche movies are rarely watched.

Why is the Long Tail Important?

- Most **recommender systems favour popular items**, which can lead to a **lack of diversity**.
- Users might **never discover** rare but relevant items.

✅ **Evaluation Approach:**

Instead of just RMSE/MAE, use **diversity and novelty metrics** to ensure **long-tail items** are recommended.

Evaluating Ranking via Correlation

- Instead of just predicting ratings, recommendation systems should **rank** items in the **correct order**.

- Ranking correlation measures **how similar** the predicted ranking is to the actual user preferences.

1. Spearman's Rank Correlation (ρ)

📌 Formula:

$$\rho = 1 - \frac{6 \sum d_i^2}{N(N^2 - 1)}$$

where:

- d_i = difference between predicted and actual ranking for item i
- N = total number of items

✅ Higher ρ means a **better ranking match.**

2. Kendall's Tau (τ)

- Measures how often item pairs are ranked **correctly or incorrectly.**
- **If Item A should be ranked above Item B but isn't**, it lowers Kendall's Tau.

✅ **Why Correlation Metrics?**

- Even if RMSE is low, the ranking of items might still be incorrect.
- Correlation metrics **focus on ranking quality**, not just rating prediction accuracy.

Evaluating Ranking via Utility

Utility-based evaluation measures how **valuable** recommendations are to users.

Discounted Cumulative Gain (DCG)

📌 Formula:

$$DCG = \sum_{i=1}^{N} \frac{\text{relevance}_i}{\log_2(i+1)}$$

- Rewards **highly relevant items appearing earlier** in the ranked list.
- $\log_2(i+1)$ **penalizes** items that appear lower in the ranking.

✅ **Better DCG means** the system ranks useful items **higher in the list.**

🔄 **Normalized DCG (NDCG):**

- Adjusts DCG to a **scale of 0 to 1.**
- Helps compare rankings across different datasets.

Evaluating Ranking via Receiver Operating Characteristic (ROC)

ROC evaluation is used when recommendations are binary (e.g., recommend or not).

1. True Positive Rate (TPR)

$$TPR = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

- Measures **how many relevant items were correctly recommended.**

2. False Positive Rate (FPR)

$$FPR = \frac{\text{False Positives}}{\text{False Positives} + \text{True Negatives}}$$

- Measures how **many irrelevant items were wrongly recommended.**

3. Area Under the Curve (AUC)

- **AUC = 1** means perfect recommendations.
- **AUC = 0.5** means random recommendations (bad).

✅ **Use AUC when recommendations are treated as a classification problem** (e.g., "Should we recommend this item or not?").

Which Ranking Measure is Best?

Each metric serves different purposes:

Metric	Use Case
RMSE / MAE	Best for rating **prediction accuracy**
Spearman's ρ	Best for **ranking correlation**
DCG / NDCG	Best for ranking **relevance**
AUC	Best for **binary recommendations** (e.g., recommend or not)

✅ **Choosing the best metric depends on the goal:**

- If **rating accuracy** is the focus → **Use RMSE/MAE.**
- If **ranking quality** matters more → **Use DCG, Spearman's, or Kendall's Tau.**
- If recommendations are **binary decisions** → **Use AUC.**

Conclusion

Offline evaluation is crucial in **fine-tuning** recommender systems before real deployment. No single metric is **perfect**—choosing the right one depends on the **objective** of the recommendation system.