A Comprehensive Understanding of Evaluation Metrics and User Experience in Recommender Systems

Lucas Kim
Loyola Marymount University
Los Angeles, CA, USA

ABSTRACT

Evaluation lies at the core of building effective recommender systems—those that not only predict accurately, but also drive user engagement, business value, and long-term satisfaction. As recommender systems grow central to products, evaluation metrics must evolve beyond static accuracy to reflect real-world goals.

This survey introduces a comprehensive framework for evaluating recommender systems across four dimensions: similarity-based relevance, predictive modeling accuracy, user experience impact, and business or behavioral outcomes. By exploring both offline and online paradigms, as well as behavioral and perceptual indicators, we aim to highlight how thoughtful metric design supports sustainable system development and long-term engagement.

Rather than treating metrics as mere performance indicators, we position them as feedback signals that guide model iteration, product design, and experience optimization. By organizing metrics through this lens, we aim to support practitioners in aligning model development with product impact, ensuring that evaluation becomes a mechanism for system growth, not just assessment.

KEYWORDS

Evaluation metrics, Recommender Systems, Product performance, User experience, A/B testing, Online learning, LLMs, Ranking metrics, Behavioral feedback

1 INTRODUCTION

As intelligent algorithms increasingly shape personalized experiences, recommender systems have become central to digital products ranging from online platforms to services and beyond. Although many systems achieve technical sophistication, their real-world success hinges on how they are evaluated, not just for accuracy, but for their impact on users and product outcomes.

Historically, evaluation in recommender systems has emphasized metrics such as RMSE, precision, or recall, largely reflecting an offline, accuracy-focused mindset. However, these metrics often fail to capture whether the system drives meaningful user engagement, supports business objectives, or adapts effectively over time. As recommender systems become more embedded in user-facing products, evaluation must evolve beyond technical benchmarking.

This survey proposes a holistic framework for understanding evaluation in recommender systems, structured across three interconnected dimensions:

- **Similarity and Predictive Metrics** – capturing how well the model estimates user preferences through pairwise comparison, regression, and ranking.
- **User Experience Metrics** – evaluating how recommendations affect user engagement, satisfaction, and diversity of exposure.
- **Business and Behavioral Metrics** – assessing downstream impact on platform performance, from click-through to retention and fairness.
- **Online Evaluation Metrics** – leveraging real-time user interactions to validate models in production environments through A/B testing, interleaving, and adaptive bandit-based feedback.

By organizing metrics along these axes, we aim to bridge the gap between model-centric and product-centric evaluation. The goal is not only to catalog metrics, but to show how they serve as feedback mechanisms: facilitating model improvement, informing design decisions, and ultimately shaping system impact in real-world settings.

2 SIMILARITY METRICS

2.1 Cosine Similarity

Measures the cosine of the angle between two non-zero vectors, focusing on orientation rather than magnitude. It quantifies the directional similarity between vectors, making it effective for high-dimensional and sparse data representations.

$$\text{Cosine Similarity} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Applications: Used in collaborative filtering and content-based filtering to compute similarity between user or item vectors.
Example: *CosineRecom: A KNN-Based Movie Recommendation System Using Cosine Similarity* [1]

2.2 Euclidean Distance

Computes the straight-line distance between two points in a multi-dimensional space. It assumes all features contribute equally and is sensitive to scale differences across dimensions.

$$\text{Euclidean Distance} = \sqrt{\sum_{i=1}^{n} (A_i - B_i)^2}$$

Applications: Used in user-based collaborative filtering or clustering approaches, because it captures overall user similarity based on absolute feature values, making it suitable for grouping users with similar preferences in continuous-valued spaces.
Example: *Metric Factorization: Recommendation beyond Matrix Factorization* [2]

2.3 Jaccard Index

Measures the similarity between finite sets by comparing intersection over union. It captures how much two sets overlap relative to their combined size, making it effective for binary or implicit feedback data in recommender systems.

$$\text{Jaccard Index} = \frac{|A \cap B|}{|A \cup B|}$$

Applications: Best used in binary preference settings (e.g., implicit feedback), because it evaluates set overlap without relying on rating magnitude, aligning well with click/view-based interactions.
Example: *Combinations of Jaccard with Numerical Measures for Collaborative Filtering Enhancement: Current Work and Future Proposal* [3]

2.4 Hamming Distance

Counts the number of differing positions between two binary strings. It quantifies dissimilarity by measuring how many attributes do not match, making it suitable for comparing binary preference vectors in recommendation tasks.

$$\text{Hamming Distance} = \sum_{i=1}^{n} \delta(A_i, B_i), \quad \delta(x, y) = \begin{cases} 1, & \text{if } x \neq y \\ 0, & \text{if } x = y \end{cases}$$

Applications: Used for binary vector comparison in memory-efficient recommenders, because it enables fast similarity computations with minimal overhead, especially in large-scale systems with sparse, implicit feedback data.
Example: *Deep Pairwise Hashing for Cold-start Recommendation* [4]

2.5 Manhattan Distance

Calculates the sum of the absolute differences across dimensions. It treats each feature independently and is more robust to outliers than Euclidean distance in high-dimensional, sparse data.

$$\text{Manhattan Distance} = \sum_{i=1}^{n} |A_i - B_i|$$

Applications: Preferred when working with high-dimensional sparse data, because it preserves the additive nature of differences and avoids the distortion caused by squaring large values, making it more stable for recommendation scenarios.
Example: *A Manhattan Distance Based Hybrid Recommendation System* [5]

2.6 Chebyshev Distance

Calculates the maximum absolute coordinate difference between two vectors. It focuses on the worst-case deviation, making it useful in systems sensitive to single extreme differences.

$$\text{Chebyshev Distance} = \max_i |A_i - B_i|$$

Applications: Ubecause the distance reflects the maximum discrepancy between any single dimension. This makes it ideal when extreme variation in any feature indicates abnormal or critical behavior.
Example: *Twin Subsequence Search in Time Series* [6]

2.7 Adjusted Cosine Similarity

Extends cosine similarity by removing user bias through mean-centering of ratings. This allows more fair comparisons between items, especially when users have different rating habits.

$$\text{Adjusted Cosine Similarity}_{i,j} = \frac{\sum_{u \in U} (R_{u,i} - \bar{R}_u)(R_{u,j} - \bar{R}_u)}{\sqrt{\sum_{u \in U} (R_{u,i} - \bar{R}_u)^2} \sqrt{\sum_{u \in U} (R_{u,j} - \bar{R}_u)^2}}$$

Applications: Commonly used in item-based collaborative filtering to correct for individual user bias in rating scale, because it normalizes differences in user tendencies and allows more objective comparisons of similarity between items.
Example: *Item-based collaborative filtering recommendation algorithms* [7]

2.8 Pearson Correlation Coefficient

Measures the linear correlation between two variables, indicating how strongly they move together. It is sensitive to scale and central tendency, making it suitable when rating magnitude matters.

$$r = \frac{\sum (A_i - \bar{A})(B_i - \bar{B})}{\sqrt{\sum (A_i - \bar{A})^2} \sqrt{\sum (B_i - \bar{B})^2}}$$

Applications: Used in user-based collaborative filtering to capture similarity in rating behavior trends, since it considers both direction and strength of user rating patterns. This helps identify users who consistently rate items in a similar way.
Example: *A Note on Pearson Correlation Coefficient in Recommender System* [8]

2.9 Spearman Rank Correlation

Measures the strength of a monotonic relationship based on rank rather than raw value. It is ideal when absolute differences are less meaningful than order.

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

Applications: Useful for ranking-based recommenders where users care about preference order, not rating scale, because it captures the consistency of ranked preferences rather than their numerical distance.
Example: *A Modified Spearman's Rank Correlation Coefficient for an Efficient Method of Similarity Calculation in Collaborative Filtering-Based Recommendation* [9]

3 RANKING METRICS

3.1 Precision@K

Measures the proportion of top-K recommended items that are relevant to the user. Useful in applications where showing only a few highly accurate recommendations is important.

$$\text{Precision@K} = \frac{\text{Number of relevant items in top K}}{K}$$

Applications: Common in top-N recommenders and UI-limited systems such as mobile or web apps, where screen space is limited and only the most relevant items can be shown.
Example: *Probabilistic Metric Learning with Adaptive Margin for Top-K Recommendation* [10]

3.2 Recall@K

Measures the proportion of all relevant items that appear in the top-K recommendations. Reflects the model's ability to capture the full set of relevant items.

$$\text{Recall@K} = \frac{\text{Number of relevant items in top K}}{\text{Total number of relevant items}}$$

Applications: Valuable in systems where missing any relevant element could harm the user experience, such as academic search or e-Commerce.
Example: *Knowledge-Aware Meta-Concept Framework for Fine-Grained Recommendation Services* [11]

3.3 F1@K

Harmonic mean of Precision@K and Recall@K. Balances the trade-off between precision and recall in the top-K recommendation.

$$\text{F1@K} = 2 \cdot \frac{\text{Precision@K} \cdot \text{Recall@K}}{\text{Precision@K} + \text{Recall@K}}$$

Applications: Used in scenarios requiring a balanced view of accuracy and completeness, such as fashion or news recommendation.
Example: *Mutual Information Assisted Ensemble Recommender System for Identifying Critical Risk Factors in Healthcare Prognosis* [12]

3.4 Average Precision@K

Computes the average of precision values at each position in the ranked list where a relevant item is found. Provides a finer-grained view of ranking quality.

$$\text{AP@K} = \frac{1}{\text{Number of relevant items}} \sum_{k=1}^{K} P(k) \cdot rel(k)$$

Applications: Useful when relevance varies by position and maintaining quality throughout the ranked list is important.
Example: *BERT4Rec: Sequential Recommendation with Bidirectional Transformers* [11]

3.5 Average Recall@K

Averages recall values across users or sessions. Focuses on retrieving relevant items for each user more consistently.

$$\text{Average Recall@K} = \frac{1}{|U|} \sum_{u \in U} \frac{|\text{RelevantItems}_u \cap \text{RecommendedItems}_u @K|}{|\text{RelevantItems}_u|}$$

Applications: Used in personalized or session-based recommenders where individual consistency matters.
Example: *Debiasing the Cloze Task in Sequential Recommendation with Bidirectional Transformers* [13]

3.6 Mean Average Precision (MAP)

Averages AP@K across all users.
Comprehensive metric for ranking evaluation when relevance scores vary.

$$\text{MAP} = \frac{1}{|U|} \sum_{u \in U} \text{AP}_u$$

Applications: Suitable for datasets with variable relevance where holistic performance matters.
Example: *Personalized Re-ranking for Recommendation* [14]

3.7 nDCG (Normalized Discounted Cumulative Gain)

Accounts for graded relevance and position of relevant items. Heavily penalizes relevant items appearing later in the list.

$$\text{nDCG@K} = \frac{DCG@K}{IDCG@K}, \quad DCG@K = \sum_{i=1}^{K} \frac{rel_i}{\log_2(i+1)}$$

Applications: Ideal for systems like search engines where relevance varies in degree and early ranking is crucial.
Example: *Multi-Resolution Diffusion for Privacy-Sensitive Recommender Systems* [15]

3.8 Hit Ratio@K

Measures whether at least one of the top-K recommended items is relevant to the user.
Unlike precision or ranking-based metrics, this binary measure reflects if the system makes any correct prediction within top-K.

$$\text{HR@K} = \frac{1}{|U|} \sum_{u=1}^{|U|} \mathbb{I}[\text{Hit}_u @K]$$

Applications: Commonly used in top-K recommendation tasks, especially when the presence of a relevant item is more important than its position. It is a fundamental benchmark for evaluating the effectiveness of recall-style.
Example: *Mamba for Scalable and Efficient Personalized Recommendations* [16]

3.9 ARHR@K (Average Reciprocal Hit Rank)

Averages the reciprocal rank of relevant items in the top-K list.
Rewards placing relevant items earlier in the ranked list.

$$\text{ARHR@K} = \sum_{i=1}^{K} \frac{1}{rank_i} \cdot hit(i)$$

Applications: Prioritizes early engagement, important in media or music recommenders where quick discovery matters.

3.10 Mean Reciprocal Rank (MRR)

Average of reciprocal ranks of the first relevant item for each query/user.
Emphasizes early relevance in a ranked list.

$$\text{MRR} = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{rank_i}$$

Applications: Widely used in Q&A systems or first-item retrieval scenarios.
Example: *Mamba for Scalable and Efficient Personalized Recommendations* [16]

4 PREDICTIVE METRICS

4.1 Root Mean Squared Error (RMSE)

Calculates the square root of the average squared differences between the predicted and actual values. It penalizes larger errors more heavily due to squaring, which makes it sensitive to outliers.

$$\text{RMSE} = \sqrt{\frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2}$$

Applications: Commonly used in rating prediction tasks where large errors are particularly undesirable, as it emphasizes large deviations more than small ones.
Example: *A Neural Network Based Explainable Recommender System* [17]

4.2 Mean Absolute Error (MAE)

Computes the average of the absolute differences between the predicted and actual values. Treat all errors equally, regardless of direction or magnitude.

$$\text{MAE} = \frac{1}{n} \sum_{i=1}^{n} |y_i - \hat{y}_i|$$

Applications: Suitable when every unit of error is equally important, such as in general-purpose recommender evaluation, since it is less sensitive to outliers and provides a more interpretable average error.
Example: *TRSM-RS: A Movie Recommender System Based on Users' Gender and New Weighted Similarity Measure* [18]

4.3 Mean Squared Error (MSE)

Calculates the average of the squared differences between predicted and actual values. Often used as an optimization objective for regression-based recommenders.

$$\text{MSE} = \frac{1}{n} \sum_{i=1}^{n} (y_i - \hat{y}_i)^2$$

Applications: Frequently used when training models through gradient descent, as its smooth, differentiable nature facilitates efficient optimization.
Example: *Large Language Models meet Collaborative Filtering: An Efficient All-round LLM-based Recommender System* [19]

4.4 Mean Absolute Percentage Error (MAPE)

Measures prediction accuracy as a percentage by calculating the average absolute percentage difference between predicted and actual values. Expresses error relative to actual value, which aids interpretability across varying scales.

$$\text{MAPE} = \frac{100\%}{n} \sum_{i=1}^{n} \left| \frac{y_i - \hat{y}_i}{y_i} \right|$$

Applications: Often used in forecasting or capacity planning tasks in recommender systems, especially when proportional error matters more than absolute error.
Example: *Price-guided user attention in large-scale E-commerce group recommendation* [20]

4.5 Coefficient of Determination (R^2)

Indicates the proportion of the variance in the dependent variable explained by the model. Higher values indicate better model explanatory power.

$$R^2 = 1 - \frac{\sum_{i=1}^{n} (y_i - \hat{y}_i)^2}{\sum_{i=1}^{n} (y_i - \bar{y})^2}$$

Applications: Used to evaluate how well a model captures the variance in ratings or preferences, particularly useful when modeling continuous-valued outputs.
Example: *Graph Convolutional Neural Networks for Web-Scale Recommender Systems* [21]

4.6 Explained Variance

Measures how much of the variation in the output is captured by the model predictions. Unlike R^2, it is appropriate when intercept terms are missing or for multi-output settings.

$$\text{Explained Variance} = 1 - \frac{\text{Var}(y - \hat{y})}{\text{Var}(y)}$$

Applications: Appropriate when the focus is on how much variation is reduced by the model rather than exact prediction values, especially in noise-sensitive recommendation environments.

4.7 Calibration

Measures how closely the predicted relevance scores align with actual user behavior. A well-calibrated recommender should not only rank relevant items higher, but also provide scores that accurately reflect the likelihood of user interaction.

$$\text{ECE} = \sum_{m=1}^{M} \frac{|B_m|}{n} |\text{acc}(B_m) - \text{conf}(B_m)|$$

Applications: Used to improve trustworthiness of recommendations, especially in explainable or uncertainty-aware systems. Common in probabilistic recommenders where score interpretation matters.
Example: *Calibrated Recommendations* [22]

5 USER EXPERIENCE METRICS

5.1 Novelty

Quantifies how unexpected or unknown the recommended items are to the user. Encourages discovery by promoting less popular items that reduce redundancy in recommendations.

$$\text{Novelty}(R_u) = \frac{1}{|R_u|} \sum_{i \in R_u} (1 - \text{popularity score}(i))$$

Applications: Reduces the filter bubble effect by suggesting lesser-known or long-tail items. Often applied in exploratory recommenders and systems aiming to improve user engagement by introducing unfamiliar content.

5.2 Diversity

Measures the dissimilarity among recommended items in a list. Ensures variety so users are exposed to different item types.

$$\text{ILD}(R) = \frac{2}{n(n-1)} \sum_{i=1}^{n-1} \sum_{j=i+1}^{n} \text{dissimilarity}(r_i, r_j)$$

Applications: Enhances content diversity and avoids recommendation redundancy. Useful in content-based filtering and hybrid recommenders to balance similarity with exploration.

5.3 Serendipity

Captures how many recommended items are both relevant and unexpectedly novel. Balances usefulness and surprise to drive deeper engagement.

$$\text{Serendipity}(u) = \frac{|\{i \in R(u) \cap L(u) \cap D(u)\}|}{|R(u)|}$$

Applications: Used in personalized recommenders to provide unexpected but delightful experiences. Helps avoid monotony in highly personalized systems.

5.4 Catalog Coverage

Measures how much of the entire item catalog is covered by recommendations. Higher coverage implies wider item exposure and reduced popularity bias.

$$\text{Catalog Coverage} = \frac{|\text{Unique Items Recommended}|}{|\text{Total Items in Catalog}|} \times 100\%$$

Applications: Promotes inclusion of long-tail content. Essential in retail and streaming services to surface diverse content beyond top popular items.

5.5 Coverage@K

Measures the proportion of the catalog that appears in the top-K recommendations across users.

$$\text{Coverage@K} = \frac{|\bigcup_{u \in U} R_u^K|}{|\text{Catalog}|}$$

Applications: Used to quantify variety in personalized recommendation at the decision surface. Highlights effectiveness of the model in surfacing a diverse portion of the catalog.

5.6 Distributional Coverage

Quantifies how evenly recommendations are distributed over the catalog using entropy. Aims to avoid dominance by a few popular items.

$$\text{DC} = -\sum_{i=1}^{N} p(i) \log_2 p(i)$$

Applications: Supports fairness and avoids content over-concentration. Applied in multi-stakeholder environments where balance across item providers is crucial.

5.7 Popularity Bias

Measures the tendency of a recommender system to over-represent popular items. Helps assess recommendation equality across the item spectrum.

$$\text{PopularityBias} = \frac{1}{|U|} \sum_{u \in U} \frac{1}{|R_u|} \sum_{i \in R_u} \text{popularity}(i)$$

Applications: Used to ensure long-tail item discovery and reduce over-reliance on top-sellers or trending content. Important in fairness-aware and diversity-boosted recommenders.

6 BUSINESS AND BEHAVIORAL METRICS

6.1 Click-Through Rate (CTR)

Measures how often users click on a recommendation after viewing it. Indicates immediate engagement or surface-level interest.

$$\text{CTR} = \frac{\text{Number of Clicks}}{\text{Number of Impressions}}$$

Applications: Used in online A/B testing to evaluate recommender effectiveness. Helps optimize UI placement, personalization ranking, and short-term engagement.

6.2 Conversion Rate (CVR)

Tracks how many users perform a desired action (e.g., purchase) after a recommendation. Highlights commercial effectiveness and deeper engagement.

$$\text{CVR} = \frac{\text{Number of Conversions}}{\text{Number of Visitors}}$$

Applications: Monitors monetization success and sales effectiveness. Central in e-commerce, ads, and transactional recommender pipelines.

6.3 Bounce Rate

Measures the percentage of users who leave after viewing only one page. High bounce rates suggest poor recommendation targeting.

$$\text{Bounce Rate} = \frac{\text{Single-Page Sessions}}{\text{Total Sessions}}$$

Applications: Used in funnel analysis and homepage optimization. Helps assess initial recommendation impact on session continuation.

6.4 Retention Rate

Indicates how well a platform retains users over a given time window. Used to assess long-term user satisfaction and habit-forming behavior.

$$\text{Retention Rate}_D = \frac{\text{Users on Day D}}{\text{Users on Day 0}}$$

Applications: Core to evaluating user loyalty and the long-term success of personalization strategies. Essential in SaaS and subscription-based products.

6.5 Session Duration

Average time users spend actively engaging with the system per session. Higher duration may indicate better content relevance and flow.

$$\text{Avg. Session Duration} = \frac{\sum \text{Session Lengths}}{\text{Number of Sessions}}$$

Applications: Used to measure user engagement depth in media and entertainment platforms. Correlates with recommendation quality and satisfaction.

6.6 Net Promoter Score (NPS)

User survey-based metric indicating loyalty and likelihood to recommend the service. Balances quantitative data with perceptual user satisfaction.

$$\text{NPS} = \%\text{Promoters} - \%\text{Detractors}$$

Applications: Acts as a user sentiment signal to track satisfaction over time. Helps assess overall recommendation impact and drives product strategy.

6.7 Cumulative Regret

Quantifies the loss from not recommending the optimal item over time. Lower regret suggests better learning and adaptation in online models.

$$\text{Regret}_T = \sum_{t=1}^{T} (r_t^* - r_{a_t})$$

Applications: Foundational in bandit-based and reinforcement recommenders. Enables evaluation of exploration-exploitation tradeoffs in adaptive systems.

6.8 Latency

Time taken to generate and return a recommendation after user input. High latency can degrade real-time user experience.

$$\text{Latency} = \text{Time}_{response} - \text{Time}_{request}$$

Applications: Used to ensure responsive user interaction. Vital for real-time and edge-device deployment scenarios.

6.9 Temporal Stability

Assesses how consistent recommendations are over time, assuming minimal change in user behavior.

$$\text{Stability}(u) = 1 - \frac{|R_u^t \Delta R_u^{t+1}|}{|R_u^t \cup R_u^{t+1}|}$$

Applications: Ensures smoother user experience and trust in recommendation continuity. Critical in session-based and dynamic recommenders.

7 ONLINE EVALUATION METRICS

7.1 A/B Testing

A/B testing splits live user traffic into distinct groups, each exposed to a different version of the recommendation system. The effectiveness of each version is compared using real-time metrics like CTR, CVR, or dwell time. A/B testing enables causal inference by isolating changes in user behavior resulting from model updates or UI changes. This approach is essential for validating new algorithms or interfaces before full-scale deployment. To minimize bias, it often requires large user samples and statistical significance testing over time.

7.2 Interleaving Evaluation

Interleaving displays a mixed list of recommendations from two competing models to the same user in a single session. By tracking which model's items users interact with (e.g., clicks), preference signals are inferred without requiring separate user groups. This allows for faster and more data-efficient evaluation than A/B testing. It's particularly useful for iterative model comparison and micro-optimizations in ranking strategies, where quick feedback is needed without full rollout.

7.3 Bandit-Based Evaluation

In bandit-based evaluation, the system continuously balances exploration (trying new recommendations) and exploitation (serving known successful items) while learning from user responses. Each arm (model variant or recommendation strategy) receives traffic proportionally to its past reward. This setup supports real-time adaptation and evaluation, especially when item popularity or user preferences shift rapidly. It is widely used in personalization, adserving, and news/article recommendations, where engagement must be optimized under uncertainty.

7.4 Cumulative Regret (Online)

Cumulative regret quantifies the loss incurred by not recommending the best possible item at each timestep. In online recommenders using bandits or reinforcement learning, regret is minimized over time as the model learns optimal behaviors. Tracking regret provides insight into how efficiently a system adapts to user preferences and how much it sacrifices short-term performance for long-term learning. This is critical in domains like e-commerce or gaming where rapid adaptation is key.

7.5 Temporal CTR / CVR Trends

Monitoring CTR and CVR over time provides visibility into how user engagement evolves post-deployment. These metrics help identify novelty effects, personalization fatigue, and degradation in relevance. Time-series trends are essential during rollout of new models, ensuring that initial improvements are sustained. Systems may segment users by recency, region, or cohort to detect changes and guide retraining schedules.

7.6 Score Calibration in Online Settings

Score calibration ensures that predicted scores from a recommender reflect actual user behavior probabilities. For example, a system predicting a 90% click likelihood should result in roughly 90 clicks out of 100 impressions. Calibration is performed by comparing predicted scores with observed outcomes and applying corrections (e.g., isotonic regression). In online settings, maintaining calibration improves trust, interpretability, and fairness, particularly in domains where predictions guide user decisions or product visibility.

8 CONCLUSION AND FUTURE PERSPECTIVES

In modern recommender systems, evaluation is not confined to technical optimization alone. A precision gain without improved engagement, or higher CTR with user churn, reveals the risk of narrow metric alignment. Effective metric design must therefore align stakeholders: data scientists, product teams, and end users.

To this end, we advocate for a unified evaluation framework grounded in three principles:

- **Multi-objective alignment:** Models must be evaluated not only for predictive accuracy but for their impact on user satisfaction, retention, and product KPIs.
- **Feedback-driven iteration:** Evaluation is not an endpoint. It fuels adaptation by serving as the feedback signal for model retraining, UI tuning, and product decision-making.
- **Systemic transparency:** Metrics should reveal not only what performs well, but why. Interpretability, calibration, and fairness must be embedded into metric pipelines to enable trustworthy recommendations.

By embedding these principles, practitioners can go beyond "evaluating to measure" and toward "evaluating to improve", a shift that is essential as recommender systems grow more complex, contextual, and user-facing.

REFERENCES

[1] Mandeep Singh, Prafull Mishra, Nitin Aggarwal, Vinay Kumar, Pramod Kumar Sharma, and Ritesh Yadav. Cosinerecom: A knn-based movie recommendation system using cosine similarity. In *2024 4th International Conference on Advancement in Electronics Communication Engineering (AECE)*, pages 190–194, 2024.

[2] Shuai Zhang, Lina Yao, Yi Tay, Xiwei Xu, Xiang Zhang, and Liming Zhu. Metric factorization: Recommendation beyond matrix factorization, 2018.

[3] Ali A. Amer and Loc Nguyen. Combinations of jaccard with numerical measures for collaborative filtering enhancement: Current work and future proposal, 2021.

[4] Yan Zhang, Ivor W. Tsang, Hongzhi Yin, Guowu Yang, Defu Lian, and Jingjing Li. Deep pairwise hashing for cold-start recommendation, 2020.

[5] Begüm Uyanık and Günce Orman. A manhattan distance based hybrid recommendation system. *International Journal of Applied Mathematics Electronics and Computers*, 11:20–29, 03 2023.

[6] Georgios Chatzigeorgakidis, Dimitrios Skoutas, Kostas Patroumpas, Themis Palpanas, Spiros Athanasiou, and Spiros Skiadopoulos. Twin subsequence search in time series, 2021.

[7] Badrul Sarwar, George Karypis, Joseph Konstan, and John Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, page 285–295, New York, NY, USA, 2001. Association for Computing Machinery.

[8] Leily Sheugh and Sasan H. Alizadeh. A note on pearson correlation coefficient as a metric of similarity in recommender system. In *2015 AI Robotics (IRANOPEN)*, pages 1–6, 2015.

[9] Pradeep Singh, Showmik Setta, and Ishwari Rajput. A modified spearman's rank correlation coefficient for an efficient method of similarity calculation in collaborative filtering-based recommendation. *SSRN Electronic Journal*, 01 2019.

[10] Chen Ma, Liheng Ma, Yingxue Zhang, Ruiming Tang, Xue Liu, and Mark Coates. Probabilistic metric learning with adaptive margin for top-k recommendation, 2021.

[11] Xianglin Wu, Haonan Jiang, Jingwei Zhang, Zezheng Wu, Xinghe Cheng, Qing Yang, and Ya Zhou. Kamc: Knowledge-aware meta-concept recommendation, 06 2024.

[12] Abhishek Dey, Debayan Goswami, Rahul Roy, Susmita Ghosh, Yu Shrike Zhang, and Jonathan H. Chan. Mutual information assisted ensemble recommender system for identifying critical risk factors in healthcare prognosis, 2024.

[13] Khalil Damak, Sami Khenissi, and Olfa Nasraoui. Debiasing the cloze task in sequential recommendation with bidirectional transformers. In *Proceedings of the 28th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, KDD '22, page 273–282. ACM, August 2022.

[14] Changhua Pei, Yi Zhang, Yongfeng Zhang, Fei Sun, Xiao Lin, Hanxiao Sun, Jian Wu, Peng Jiang, and Wenwu Ou. Personalized re-ranking for recommendation, 2019.

[15] Derek Lilienthal, Paul Mello, Magdalini Eirinaki, and Stas Tiomkin. Multi-resolution diffusion for privacy-sensitive recommender systems. *IEEE Access*, 12:58275–58287, 2024.

[16] Andrew Starnes and Clayton Webster. Mamba for scalable and efficient personalized recommendations, 2024.

[17] Jionghao Lin and Yiren Liu. A neural network based explainable recommender system, 2018.

[18] Mostafa Khalaji. Trsm-rs: A movie recommender system based on users' gender and new weighted similarity measure, 2020.

[19] Sein Kim, Hongseok Kang, Seungyoon Choi, Donghyun Kim, Minchul Yang, and Chanyoung Park. Large language models meet collaborative filtering: An efficient all-round llm-based recommender system, 2024.

[20] Yang Shi and Young joo Chung. Price-guided user attention in large-scale e-commerce group recommendation, 2024.

[21] Rex Ying, Ruining He, Kaifeng Chen, Pong Eksombatchai, William L. Hamilton, and Jure Leskovec. Graph convolutional neural networks for web-scale recommender systems. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery amp; Data Mining*, KDD '18, page 974–983. ACM, July 2018.

[22] Harald Steck. Calibrated recommendations. In *Proceedings of the 12th ACM Conference on Recommender Systems*, RecSys '18, page 154–162, New York, NY, USA, 2018. Association for Computing Machinery.