1. Evaluation Metric/Visualization

Weighted Fβ-score:

A weighted version of F1-score where β determines the importance of recall relative to precision.

$$F_\beta = (1 + \beta^2)\frac{Precision \cdot Recall}{\beta^2 \cdot Precision + Recall}$$

- If recall is more important, use β > 1 (e.g., β = 2).
- If precision is more important, user β < 1 (e.g., β = 0.5).

Why using Weighted Fβ-score: Recall is key for retrieval models.

- 4-star (or higher) ratings are labeled as "interested" and 3-star (or lower) as "not interested." However, some marginal ratings may still be acceptable.
- Ensuring the retrieval model returns enough relevant items makes recall critical.

F1-score:

When it is difficult to determine the best β for Weighted Fβ-score, standard F1-score can be used by setting thresholds for precision and recall.

Mean Reciprocal Rank (MRR):

- Reciprocal Rank (RR) measures the rank of the fist relevant item in the recommended list:

$$RR = \frac{1}{rank\ of\ the\ first\ relevant\ item}$$

- MRR is the average of reciprocal ranks across all users:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank\ of\ the\ first\ relevant\ item\ for\ user\ i}$$

- Example:
    - User A: Relevant item at rank 3 → Reciprocal Rank = $\frac{1}{3}$
    - User B: Relevant item at rank 1 → Reciprocal Rank = $\frac{1}{1}$
    - User C: No relevant item → Reciprocal Rank = 0
    - $MRR = \frac{1}{3}\left(\frac{1}{3} + 1 + 0\right) = 0.44$

Visualization:

- **Precision-Recall curves.**
- **Bar chart of MRR for different users.**

2. Advance Metric

a. Coverage

- **Definition**: Measures the proportion of the catalog (items or users) that the system can recommend:

$$Item\ Coverage = \frac{Number\ of\ unique\ items\ recommended}{Total\ number\ of\ items\ in\ the\ catalog}$$

- **Importance**: High coverage ensures a variety of items are recommended, not just popular ones.

b. Diversity

- **Definition**: Measures the dissimilarity among items in a single user's recommendation list.
- Use a similarity metric (e.g., cosine similarity) to calculate pairwise distances between items in the list:

$$Diversity = 1 - \frac{\sum_{i \neq j} Similarity(i,j)}{Number\ of\ item\ pairs}$$

- **Why It Matters**: Balances personalization with exploration.

c. Novelty

- **Definition**: Assesses how unexpected or unique recommendations are. Less popular items contribute more to novelty.
- Example Metric: Inverse Popularity Score:

$$Novelty = \frac{1}{N}\sum_{i=1}^{N} -\log_2(popularity\ of\ item\ i)$$

d. Serendipity

- **Definition**: Evaluates how surprising and delightful recommendations are beyond relevance.
- Example: Measure the difference between expected recommendations (e.g., predicted by collaborative filtering) and actual recommendations.

e. User Effort

- Definition: Measures the effort required by users to find relevant items.
- Example Metric: Average position of relevant items in the recommended list.

$$Effort = \frac{\sum_{i=1}^{N} Position\ of\ relevant\ item}{N}$$

f. Fairness

- Definition: Evaluates whether recommendations are equitably distributed among different user groups or item categories.

3. Optimizing Models

a. Improving Item Similarity Metric

1. **Cosine Similarity**: Measure angular similarity (baseline).
2. **Jaccard Index**: Focus on overlap of interaction sets, effective for binary interaction data.
3. **Adjusted Cosine Similarity**: Account for user biases.
4. **Pearson Correlation Coefficient**: Linear correlation for numerical ratings.
5. **Combine Multiple Metrics**: Weight each based on performance contribution.

b. Features Engineering

1. **Enhanced Weighting Schemes**: Assign higher weights to recent interactions or higher ratings.
2. **Incorporate Additional Parameters**: Add user demographics, business categories, or sentiment analysis of reviews. Consider contextual factors like time of interaction, user location, or session data.
3. **Timeliness and Time Decay**:
    - Exclude older reviews or apply decay factors (e.g., exponential decay).
    - Example:

$$Weight = e^{-\lambda \cdot Age\ of\ Interaction}$$

        - λ controls the rate of decay (e.g., 0.01 for gradual decay).

c. Negative Sampling

- Why: Enhances model training by focusing on harder-to-classify items.
- Dynamically generate negative samples during training/testing to address bias.

d. Loss Functions

- Experiment with ranking-specific loss functions like Bayesian Personalized Ranking (BPR) or Listwise Loss.

e. Hyperparameter Tuning

- Optimize parameters like learning rate or embedding dimensions via grid search or Bayesian optimization.

4. Testing Template

<table>
<tr><td>Stage</td><td>Retrieval</td></tr>
<tr><td>Model</td><td>Item Collaborative Filtering</td></tr>
<tr><td>Dataset</td><td>Yelp (sampled)</td></tr>
<tr><td>Train Size</td><td>80% of dataset for normal case</td></tr>
<tr><td>Evaluation Metric</td><td>
<table>
<tr><th>Metric</th><th>Rationale</th></tr>
<tr><td>Accuracy</td><td>Measures the overall correctness</td></tr>
<tr><td>Recall</td><td>Most important metric in retrieval</td></tr>
<tr><td>Precision</td><td>Ensures retrieved items are relevant</td></tr>
<tr><td>F1-score</td><td>Provides a balanced view of recall and precision</td></tr>
<tr><td>Mean Reciprocal Rank</td><td>Measures ranking of relevant items</td></tr>
<tr><td>Weighted Fβ-score</td><td>Adjusts the trade-off for recall and precision dynamically</td></tr>
</table>
</td></tr>
<tr><td>Baseline Model</td><td>Simplest Item Collaborative Filtering Model
- Item Similarity: Cosine Similarity
- Prediction Function:
$$\sum j\ like(user, item_j) \times sim(item_j, item)$$
Features: Total of 1 feature is used
- Rate in Review Table</td></tr>
<tr><td>Evaluation Strategies/Steps</td><td>Strategy 1: K-Nearest Neighbors (KNN)
Set-up:
1. Prepare user and business index tables.
2. Split 80/20 for train/test data.
3. Exclude test data during KNN computation.
4. Compute KNN results for evaluation.
Evaluation:
1. Take a pair of user_id and business_id as input.
2. Compute the list of item similarity with the user recent interacted items (some may be 0, max case will have N similarities).
3. Compute all the interest scores.
4. Compute the evaluation metric.
<table>
<tr><td rowspan="3">Accuracy</td><td>True Positive</td><td>User gave a rate of 4 or higher for business, predicted interest score is in top K results.</td></tr>
<tr><td>True Negative</td><td>User gave a rate of 3 or lower for business, predicted interest score is not in top K results.</td></tr>
<tr><td colspan="2">$$\frac{(True\ Positive + True\ Negative)}{Total}$$</td></tr>
<tr><td>Precision</td><td>False Positive</td><td>User gave a rate of 3 or below for business, predicted interest score are in top K results.</td></tr>
</table>
</td></tr>
</table>

	$\frac{True\ Positive}{True\ Positive + False\ Positive}$	
Recall	$\frac{True\ Positive}{Total\ Positive}$	
F1-Score	$\frac{2 \times precision \times recall}{precision + recall}$	
Mean Reciprocal Rank	**RR**	The respective rank of **True Positive** business in the top K results.
Weighted Fβ-score	**β**	β >1 as recall is important here.
	$\frac{(1+\beta^2) \times precision \times recall}{\beta^2 \times precision \times recall}$	

Strategy 2: Leave-One-Out Cross-Validation (LOOCV)
Set-up:

1. Set up the user and business index tables.
2. For each user, leave one interaction out of the training set and treat it as the test case.
3. Exclude this data from training.
4. Compute KNN results for evaluation.

Evaluation: Same metrics as Strategy 1, ensuring robust testing for individual interactions.