# Project Phase 2

# E-commerce Product Recommendation System

The implementation of the E-commerce Product Recommendation System in Phase 2 builds upon the foundational structures designed in Phase 1. Important activities such as recording user sessions, accessing and updating user-product interactions, and selecting the closest product suggestions based on feature vectors are made possible by modifications to the KDTree, Collaborative Filtering Matrix, and User Session Map classes made during this phase. In addition, it shows how these features work in a recommendation system by introducing the necessary code snippets and the results they produce.

Users' interactions with items are modeled in the Collaborative Filtering Matrix class, which stores ratings in a 2D matrix. The class was limited to updating the interaction matrix in Phase 1. In the second stage, we include new techniques to get all product ratings or all user ratings. For instance, the matrix records the interaction when a user assigns a score to a product. These exchanges may be retrieved by the system later on for use in making recommendations. Picture this: User 0 gives Product 0 a perfect score of 5, while User 1 gives Product 2 a very respectable score of 3. In order to make these ratings retrievable, the system saves them in a matrix. The capability is shown in the sample script by outputting the rows of the matrix that correspond to User 0 and the column for Product 2. The findings validate the accessibility and accuracy of the stored data. Collaborative filtering algorithms rely on this capacity to examine user-product interactions to provide suggestions based on common user-product preferences.

With the addition of a mechanism to get all items that a user has interacted with during a session, the User Session Map class has been improved in Phase 2. Adding interactions to a user's session was limited to a linked list structure in Phase 1. At this point, the system may get the sequence of goods that a user interacted with by traversing their linked list. For instance, the system keeps track of these interactions in linked lists if User 0 interacts with Product 101, then Product 102, and finally User 1 interacts with Product 103. As the script shows, the modified class effectively obtains this series of interactions. User 0's session is shown as [102, 101] while User 1's session is printed as [103]. These results validate the accuracy of the session map in tracking the sequence of interactions. For session-based recommendations—which improve real-time customisation by drawing on the user's recent browsing history—this feature is invaluable.

There have been major updates to the KDTree class, which uses feature vectors to describe product similarity. Adding nodes to the tree was the only functionality available in Phase 1. In the second phase, the system is introduced to the nearest\_neighbor approach, which allows it to find the product that is most similar to a given feature vector. This improvement paves the way for suggestions based on content by locating goods that have characteristics with the input. A product identifier (product\_id) is now stored with the feature vector at each KDTree node, connecting the structure of the tree to the real product catalog. A user searches for the product closest to the feature vector [2.0, 3.0], and the nearest\_neighbor method finds Product ID 201 as the closest match based on distances calculated using Euclidean metrics. The script inserts products with feature vectors [1.0, 2.0], [3.0, 4.0], and [5.0, 6.0] and associates them with Product IDs 202, 203, and 201, respectively. The outcome proves that the KDTree works as intended and shows how useful it is for making efficient and accurate product attribute-based recommendations.

To ensure the system works as intended, the example script integrates all of these features. To begin, we add five people and five goods to the Collaborative Filtering Matrix. Product 0 gets a 5 from User 0, whereas Product 2 gets a 3 from User 1. It is clear that User 0 has only rated Product 0 because the script returns [5, 0, 0, 0, 0] when it obtains their ratings. Also, when we look for Product 2 ratings, we get [0, 3, 0, 0, 0], which means that only User 1 has given this product a rating. The validity of the matrix operations is validated by these outcomes. After then, the User Session Map keeps track of what User 0 and User 1 have been up to. Product 103 is the focus of User 1, while Products 101 and 102 are handled by User 0. The linked list structure causes the retrieval of sessions for User 0 to return [102, 101], which represents the opposite sequence of interactions. As an example, the output from User 1's session is [103]. At last, three product feature vectors are added to the KDTree and it is started. Since Product ID 201 is the one that is geographically closest to the target in feature space, the script returns it after searching for the product that is closest to the feature vector [2.0, 3.0].

The components save, retrieve, and process data appropriately, as shown by the results. Collaborative Filtering Matrix models user-product interactions effectively, User Session Map monitors and retrieves engagement, and KDTree finds the most related goods to give content-based recommendations. When put together, these elements provide the framework for a strong recommendation system.

**Collaborative Filtering Matrix**

cf\_matrix = CollaborativeFilteringMatrix(num\_users=5, num\_products=5)

cf\_matrix.update\_interaction(0, 0, 5)

cf\_matrix.update\_interaction(1, 2, 3)

print("User 0 Ratings:", cf\_matrix.get\_user\_ratings(0))

print("Product 2 Ratings:", cf\_matrix.get\_product\_ratings(2))

**Output**

User 0 Ratings: [5, 0, 0, 0, 0]

Product 2 Ratings: [0, 3, 0, 0, 0]

**User Session Map**

user\_sessions = UserSessionMap()

user\_sessions.add\_interaction(0, 101)

user\_sessions.add\_interaction(0, 102)

user\_sessions.add\_interaction(1, 103)

print("User 0 Session:", user\_sessions.get\_user\_session(0))

print("User 1 Session:", user\_sessions.get\_user\_session(1))

**Output**

User 0 Session: [102, 101]

User 1 Session: [103]

**KDTree**

kdtree = KDTree()

kdtree.add\_product([1.0, 2.0], product\_id=201)

kdtree.add\_product([3.0, 4.0], product\_id=202)

kdtree.add\_product([5.0, 6.0], product\_id=203)

target\_point = [2.0, 3.0]

nearest = kdtree.nearest\_neighbor(kdtree.root, target\_point)

print("Nearest Product to", target\_point, "is Product ID:", nearest.product\_id)

**Output**

Nearest Product to [2.0, 3.0] is Product ID: 201

**Conclusion**

In conclusion, the Phase 2 implementation enhances the functionality of the E-commerce Product Recommendation System by adding essential operations to the Collaborative Filtering Matrix, User Session Map, and KDTree classes. These enhancements enable the system to store and retrieve user-product interactions, track engagement during sessions, and recommend products based on similarity. The demonstration script validates the correctness of each component, and the results confirm that the system can effectively manage data and provide recommendations. This foundation sets the stage for further development, such as integrating machine learning algorithms for predictive recommendations and scaling the system to handle real-world e-commerce datasets.

# References

Arya, S., & Mount, D. M. (2018). Approximate nearest neighbor queries in fixed dimensions.Proceedings of the ACM-SIAM Symposium on Discrete Algorithms (SODA), 271-280.

Hidasi, B., Karatzoglou, A., Baltrunas, L., & Tikk, D. (2016). Session-based recommendations with recurrent neural networks. Proceedings of the International Conference on Learning Representations (ICLR).

Koren, Y., Bell, R., & Volinsky, C. (2009). Matrix factorization techniques for recommender systems. Computer, 42(8), 30-37.