GROUP NO.: 211

E-COMMERCE RATING RECOMMENDATION SYSTEM

GROUP NO.: 211

TABLE OF CONTENT

Explore the key insights behind our E-commerce Recommendation System!

GROUP NO.: 211

INTRODUCTION

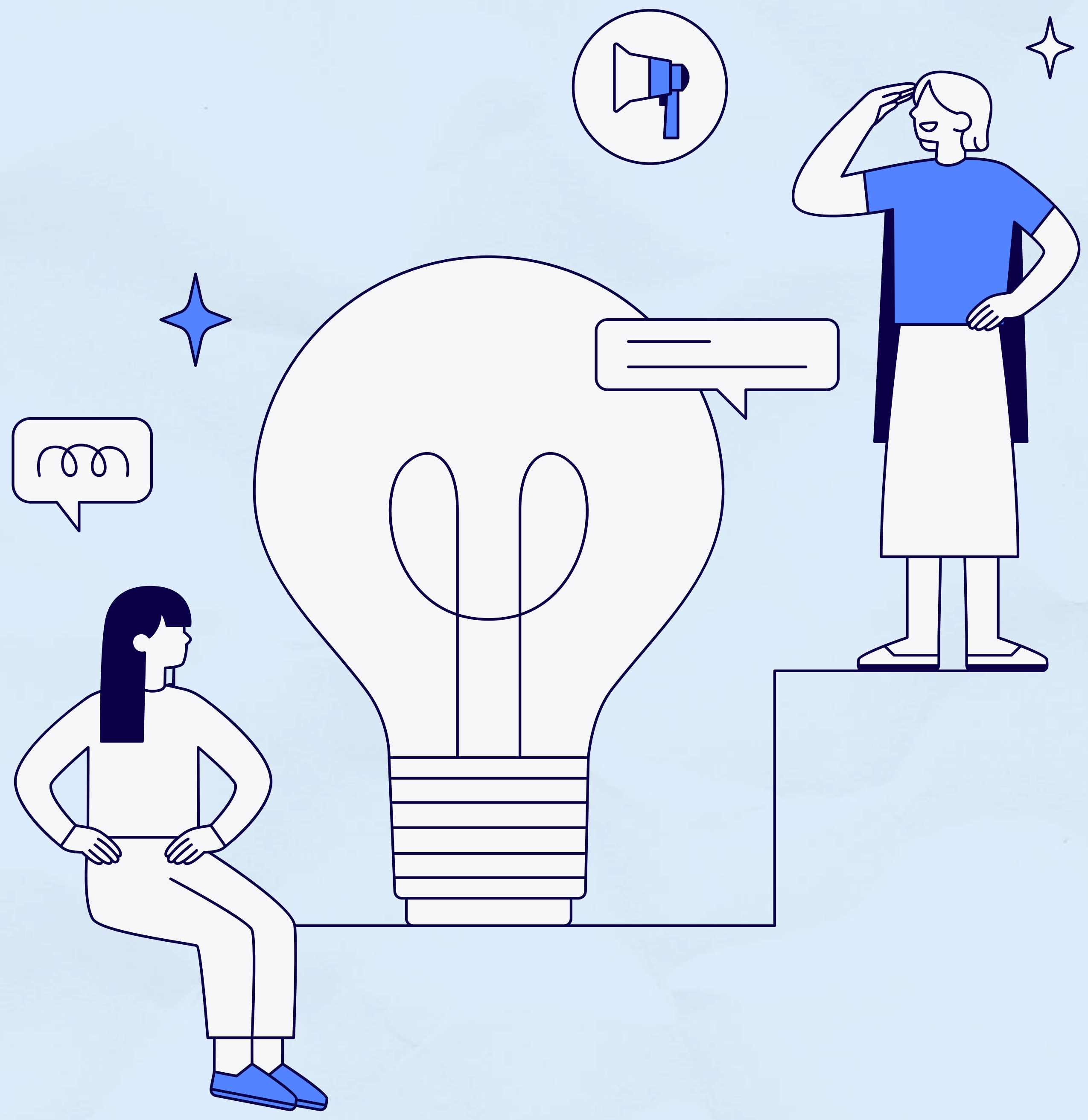
Definition of E-Commerce Rating Recommendation System!

In the rapidly evolving world of online shopping, where customers are presented with an overwhelming number of choices, personalization is the key to enhancing the shopping experience. That's where the E-commerce Rating Recommendation System comes in—a cutting-edge solution that is transforming the way consumers discover products. By analyzing customer ratings, feedback, and preferences, this system uses powerful machine learning algorithms to deliver highly personalized product recommendations.

GROUP NO.: 211

Imagine browsing through an e-commerce platform and receiving suggestions that perfectly match your interests and past behavior, all based on data from your previous interactions. The system processes vast amounts of data to understand customer needs, making the shopping experience more convenient, efficient, and tailored to each user's unique preferences.

Not only does this increase customer satisfaction, but it also drives engagement and boosts sales for online retailers, who can offer more relevant products to their users. Whether you're a regular online shopper or a business looking to enhance customer interactions, the E-commerce Rating Recommendation System is shaping the future of personalized shopping, where every rating counts and every recommendation is spot-on!



GROUP NO.: 211

LITERATURE REVIEW

S.No.	Author & Year	Title	Approach	Limitations
[1]	Badrul Sarwar, George Karypis, Joseph Konstan, John Riedl (2000)	Analysis of Recommendation Algorithms for E-Commerce	Applied statistical and knowledge discovery techniques (collaborative filtering, dimensionality reduction, and traditional data mining) on two datasets (E-commerce and MovieLens).	Scalability issues with collaborative filtering algorithms when handling large datasets. False positives in recommendations. Sparse data problems.
[2]	J. Ben Schafer, Joseph A. Konstan, John Riedl (2001)	E-Commerce Recommendation Applications	Taxonomy of e-commerce recommendation systems, examining inputs, methods, and outputs. Analyzed five application models of recommender systems across various e-commerce sites.	Does not explore newer advancements or complex machine learning models for recommender systems. Limited to early applications and common approaches.

GROUP NO.: 211

[3]	Herlocker et al. (2004)	Evaluating Collaborative Filtering Recommender Systems	Collaborative Filtering	Cold start problem for new users/items
[4]	Smith et al. (2018)	A Collaborative Filtering Based Approach for E-Commerce	Collaborative Filtering using user-item interaction data	Cold Start Problem; Sparsity in user-item matrices
[5]	Chen et al. (2019)	An Enhanced Recommendation Algorithm Using User Behavior Analysis	User Behavior Analysis	Dependence on accurate user behavior data
[6]	Farah Tawfia Abdul Hussein et al. (1897)	Six Types of Recommender Systems	Filtering the Different types of Recommender System Based on the usage and Significance	Failed to Specify the Method and Provide an overall data about the methods
[7]	Kumar ar al. (2020)	Deep Learning Techniques for Personalised Product Recommendations	Deep Learning (Neural Networks)	Requires large amount of training data; Computationally Expensive
[8]	Lijuan Xu, Xiakum Sang (2022)	Personalised Recommendation & Algorithm	Collaborative Recommendation Algorithm having benefits of filtering	Tendency of similar items to have different names

GROUP NO.: 211

[9]	Kangmin Xu, Huiming Zhou (2024)	Intelligent Classification and Personalized Recommendation system for e-commerce.	Machine learning	Data quality Inconsistency
[10]	Kexin Wu, Kun chi (2023)	Enhanced e-commerce customer engagement recommendation system.	Comprehensive three-tiered recommendation system	Privacy & Data security limitations
[11]	Fu Sheng, Jiatu Shi, Yadang Shi (2024)	e-commerce recommendation system with DL based sentiment analysis.	Deep learning(Neural networking)	Increased computational complexity and scalability
[12]	Mai Le Bich Tuyen, Nguyen Thanh Hai (2024)	Recommendation system for online sales based on transaction.	Memory based collaborative filtering	Cold start problem, sparsity in user item matrices
[13]	Davide Bacciu, Shahab Mokarizadeh, Marco Varesi (2023/2024)	Real time personalized product recommendation system.	Real time personalized recommendation system	Reduced stability with time
[14]	George Stalidis (2023)	Techniques for Retail and sustainable Marketing.	Reduced consumption and environmental impact	Tendency of similar items to have different names

GROUP NO.: 211

[15]	Yang Li, Kangbo Liu (2021)	Recent developments in recommendation system	Deep learning, Reinforcement learning	A review about other journals
[16]	Yiling Feng (2023)	A Collaborative Filtering Based Approach for E-Commerce	A six-step model for integrating self construal	Cold Start Problem; Sparsity in user-item matrices
[17]	Kailash Chowdhary, Francis Palma (2024)	Exploring the landscape of hybrid recommendation system in e-commerce	Examining how various techniques are combined to improve recommendation accuracy and user satisfaction	Case study on various journals and not a journal itself
[18]	Caiwen Li, Iskandar Ishak (2023)	Deep-learning based Recommendation system	Accuracy, Scalability and adaptability providing insights into current trends	Reduced stability and cold start problem

GROUP NO.: 211

RESEARCH GAP

A research gap refers to an area or aspect of a field that has not been adequately explored or studied. It identifies questions, problems, or unexplored issues that existing research has not yet addressed, offering an opportunity for further investigation. Research gaps can be found by reviewing current literature, understanding what has already been studied, and pinpointing where more information is needed or where inconsistencies exist. Below one can find Research gap in our perspective, There are GAPs and OPPORTUNITYs for a particular problem faced :

1. Real-Time Personalization

- **Gap:** Many recommendation systems focus on batch processing of data, leading to delays in personalisation. There may be insufficient research on real-time recommendation systems that instantly update based on user's behaviour during the session.
- **Opportunity:** Investigating methods that allow for faster, dynamic recommendations using real-time user data.

2. Cold Start Problem

- **Gap:** Limited research on effective solutions for the "cold start" problem, where the system has little or no data about new users or new items.
- **Opportunity:** Explore techniques like deep learning or hybrid models to improve recommendations for new users or products.

GROUP NO.: 211

3. Diversity vs Accuracy

- **Gap:** There is often a conflict between providing accurate recommendations and ensuring diversity in the recommendations. Current systems may lean too heavily on accuracy, leading to repetitive suggestions.
- **Opportunity:** Research techniques to balance diversity and relevance in recommendations to improve user satisfaction.

4. User Trust and Transparency

- **Gap:** Research is limited on how transparency in recommendation systems influences user trust. Users often don't know why certain products are recommended to them.
- **Opportunity:** Study the impact of making recommendation processes more transparent to users, and how this affects trust and user engagement.

5. Impact of Cultural Differences

- **Gap:** Little attention has been given to how cultural and regional differences impact recommendation systems, particularly in global e-commerce platforms.
- **Opportunity:** Explore culturally adaptive recommendation systems that cater to diverse user bases across different regions.

6. Privacy and Security Concerns

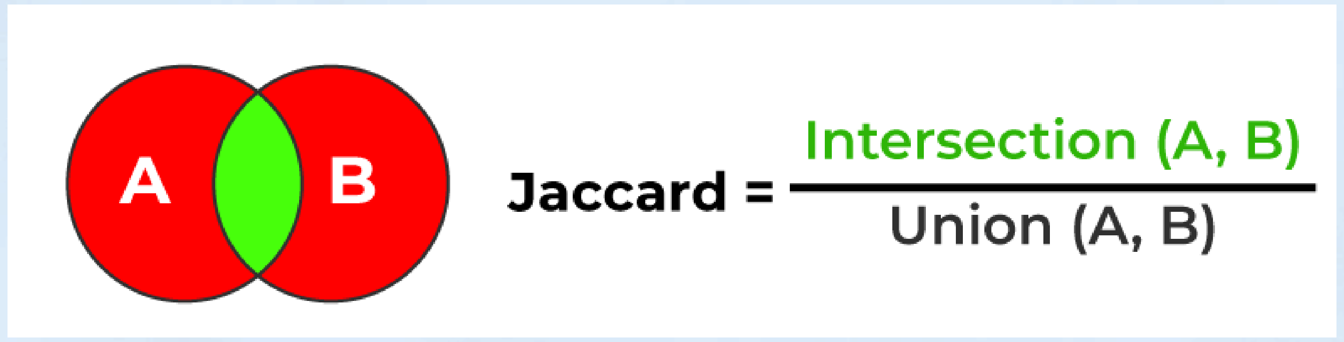
- **Gap:** Although recommendation systems rely on user data, there’s insufficient research on privacy-preserving techniques that still allow for personalized recommendations without compromising data security.
- **Opportunity:** Develop algorithms that maintain personalisation while ensuring users’ data privacy through techniques like federated learning or differential privacy.

GROUP NO.: 211

SOLUTIONS

To resolve issues like data sparsity and the cold start problem in recommendation systems , similarity algorithms such as Jaccard, Cosine, and Dice coefficients are employed. These algorithms are particularly effective for evaluating the resemblance between users or items when data availability is limited or insufficient for traditional approaches.

1. Jaccard Similarity: Jaccard Similarity also known as the Jaccard index, is a metric used to measure the similarity between two sets by comparing their shared elements relative to their total elements.



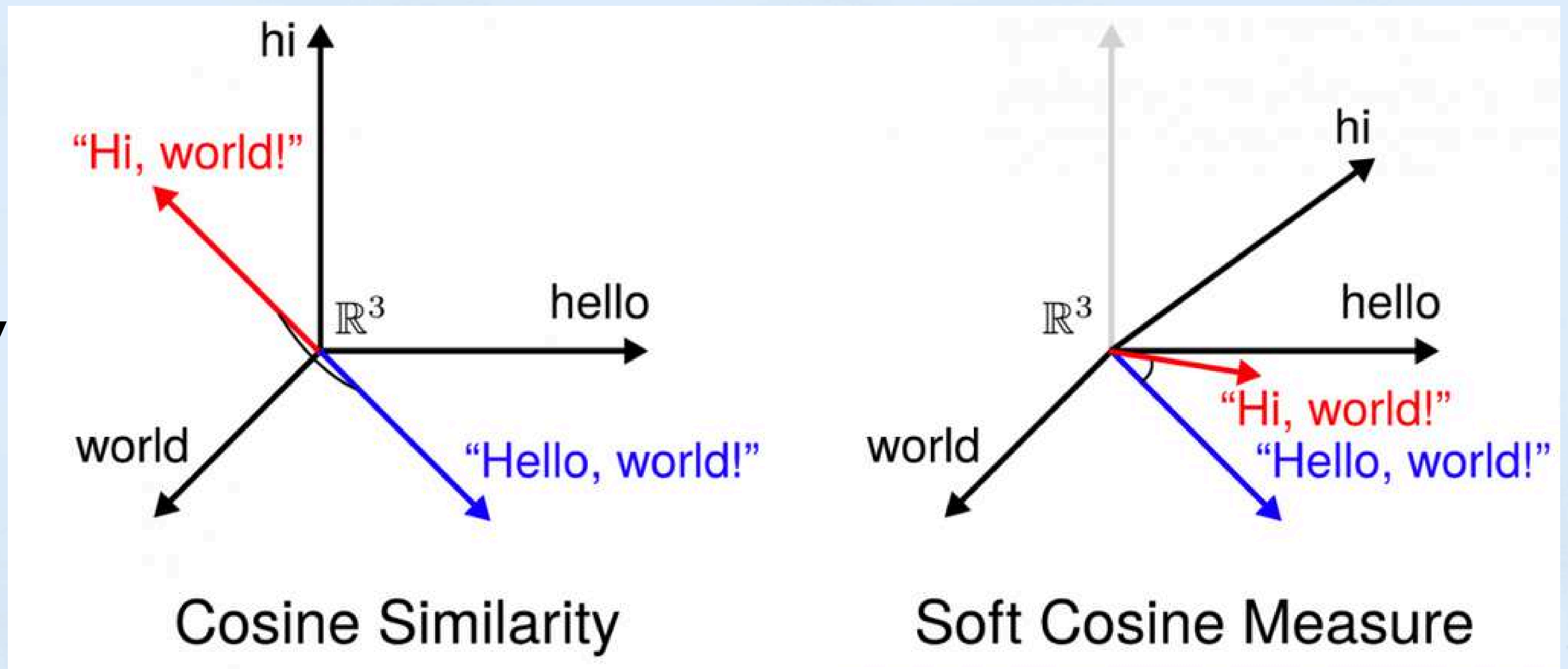
$$\text{Jaccard} = \frac{\text{Intersection }(A, B)}{\text{Union }(A, B)}$$

Mathematically, it is defined as the ratio of the size of the intersection of the sets to the size of their union. This means it quantifies how much overlap exists between the sets as a proportion of their combined elements.

2. Cosine Similarity: It is a metric used to measure the similarity between two non-zero vectors in a multidimensional space by calculating the cosine of the angle between them.

It is particularly useful for high-dimensional data where the magnitude of the vectors (i.e., the number of occurrences or values) may vary, but the orientation (i.e., the direction in the feature space) is more important.

Cosine similarity is widely used in text mining and natural language processing to compare document similarity, as it captures the context and distribution of terms rather than their frequency.



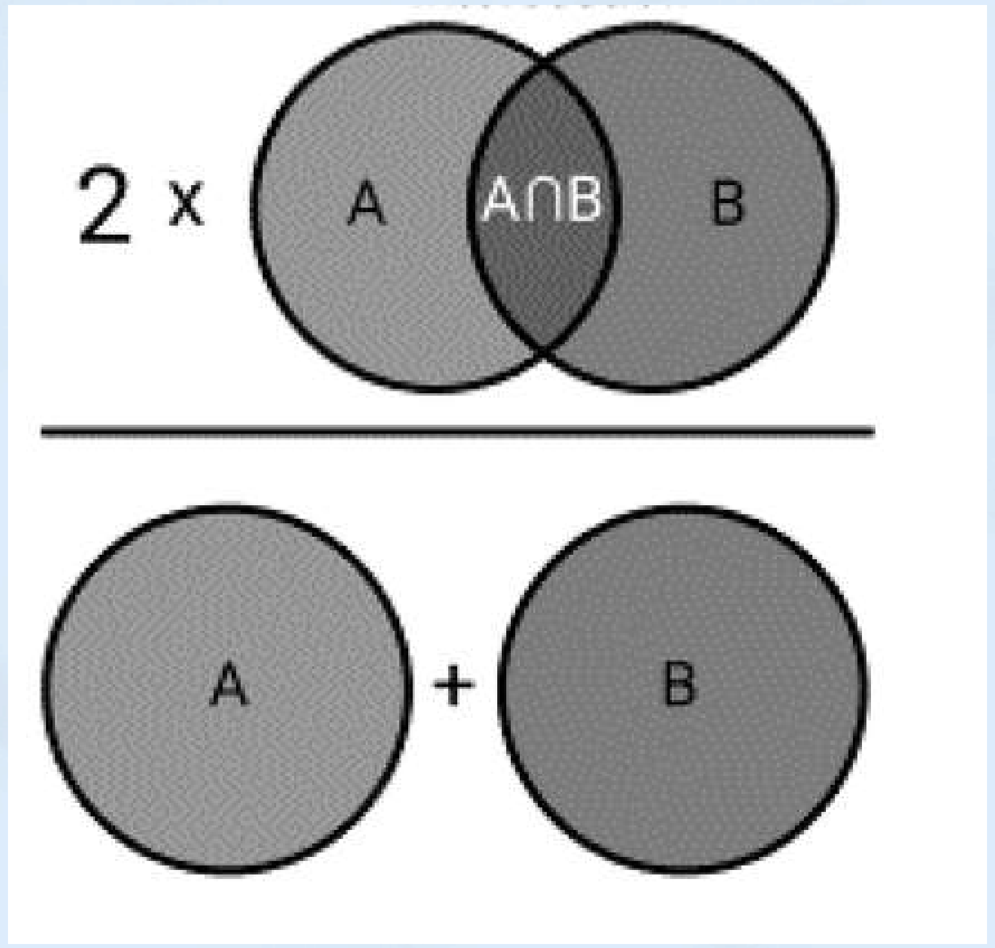
Source: https://github.com/RaRe-Technologies/gensim/blob/develop/docs/notebooks/soft_cosine_tutorial

GROUP NO.: 211

3. Dice Similarity: Dice similarity, also known as the Dice coefficient, is a statistical measure that evaluates the similarity between two sets by comparing their overlap relative to their combined size.

It is calculated as twice the size of the intersection of the sets divided by the sum of the sizes of both sets.

It is widely used in applications like bioinformatics, image segmentation, and natural language processing, where precision in identifying commonalities is crucial.



OBJECTIVE

The Objective is to derive best approach in developing a recommendation system by the help from research articles that are priorly dedicated to the same.

Users struggle to find products of their interest and it not impossible to know by the patterns they leave. It can only be tried to be more and more accurate about the personalized data. The prior points such as Research gaps and Limitations has been kept in the mind while writing this particular section. Here are some points that are being targeted while developing such a system:

Enhance User Experience

To provide a personalized shopping experience by recommending relevant products based on customer preferences and behavior.

Implement Efficient Algorithms

To deploy advanced recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid models, for optimal performance and accuracy.

GROUP NO.: 211

Boost Sales and Engagement

To increase user engagement and drive higher sales conversions by suggesting items that align with customer interests.

Leverage Data for Insights

To utilize user data, such as browsing history, purchase patterns, and demographic details, for making informed recommendations.

Implement Efficient Algorithms

To deploy advanced recommendation algorithms, including collaborative filtering, content-based filtering, and hybrid models, for optimal performance and accuracy.

Adapt to Dynamic User Needs

To continuously refine and update the recommendation engine based on real-time user interactions and changing market trends.

GROUP NO.: 211

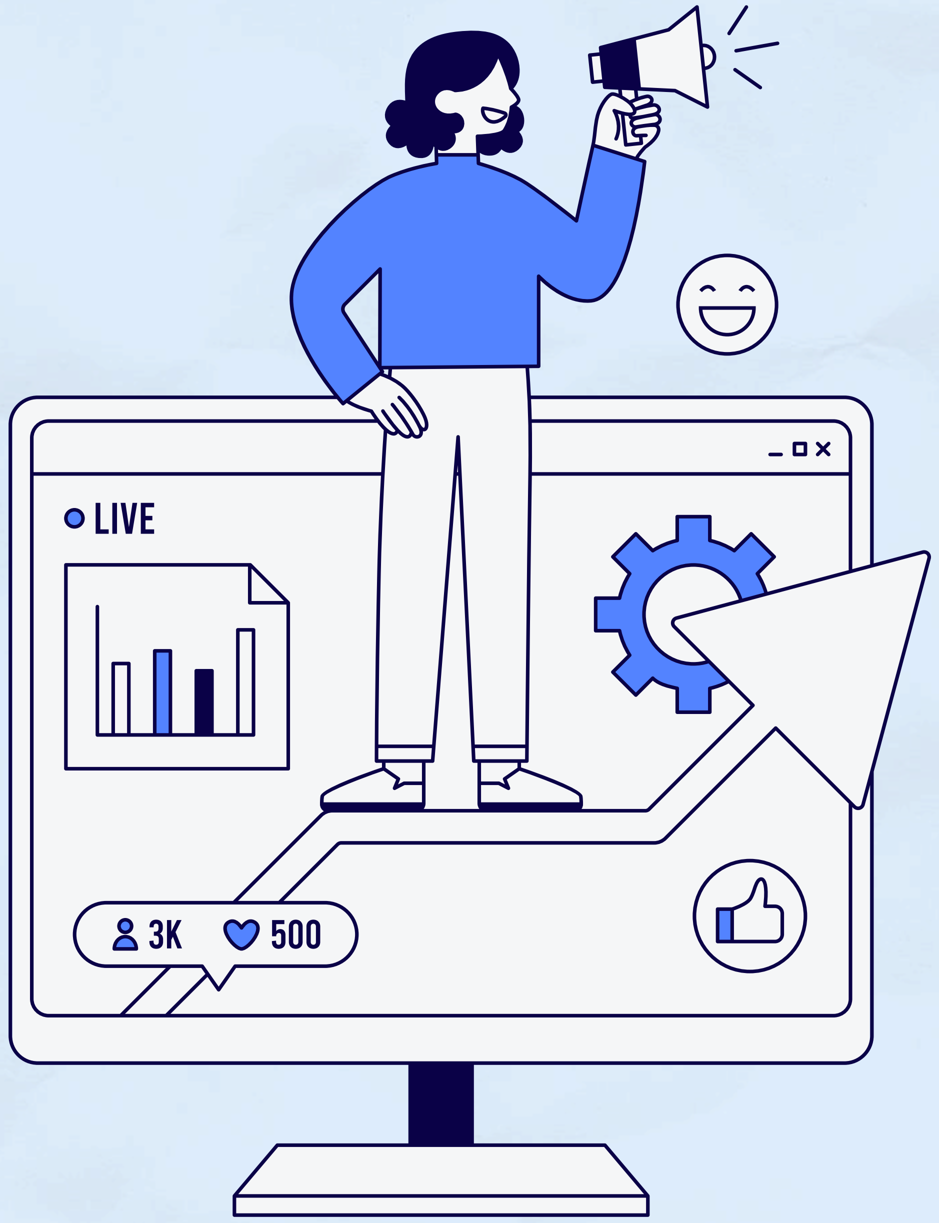
PROPOSED METHODOLOGY

THE PROPOSED METHODOLOGY WILL INVOLVE MULTIPLE STAGES:

- DATA COLLECTION
- ALGORITHMS
- EVALUATION METRICES

EACH STAGE IS CRUCIAL TO ACHIEVING THE OBJECTIVES OF DEVELOPING AN EFFICIENT, SCALABLE, AND USER-FRIENDLY RECOMMENDATION SYSTEM

DATA COLLECTION: DATA WILL BE GATHERED FROM AN E-COMMERCE PLATFORM'S TRANSACTION LOGS, USER PROFILES, AND PRODUCT CATALOG. PUBLIC DATASETS, SUCH AS AMAZON'S PRODUCT REVIEWS OR MOVIELENS, CAN BE USED FOR INITIAL EXPERIMENTATION.

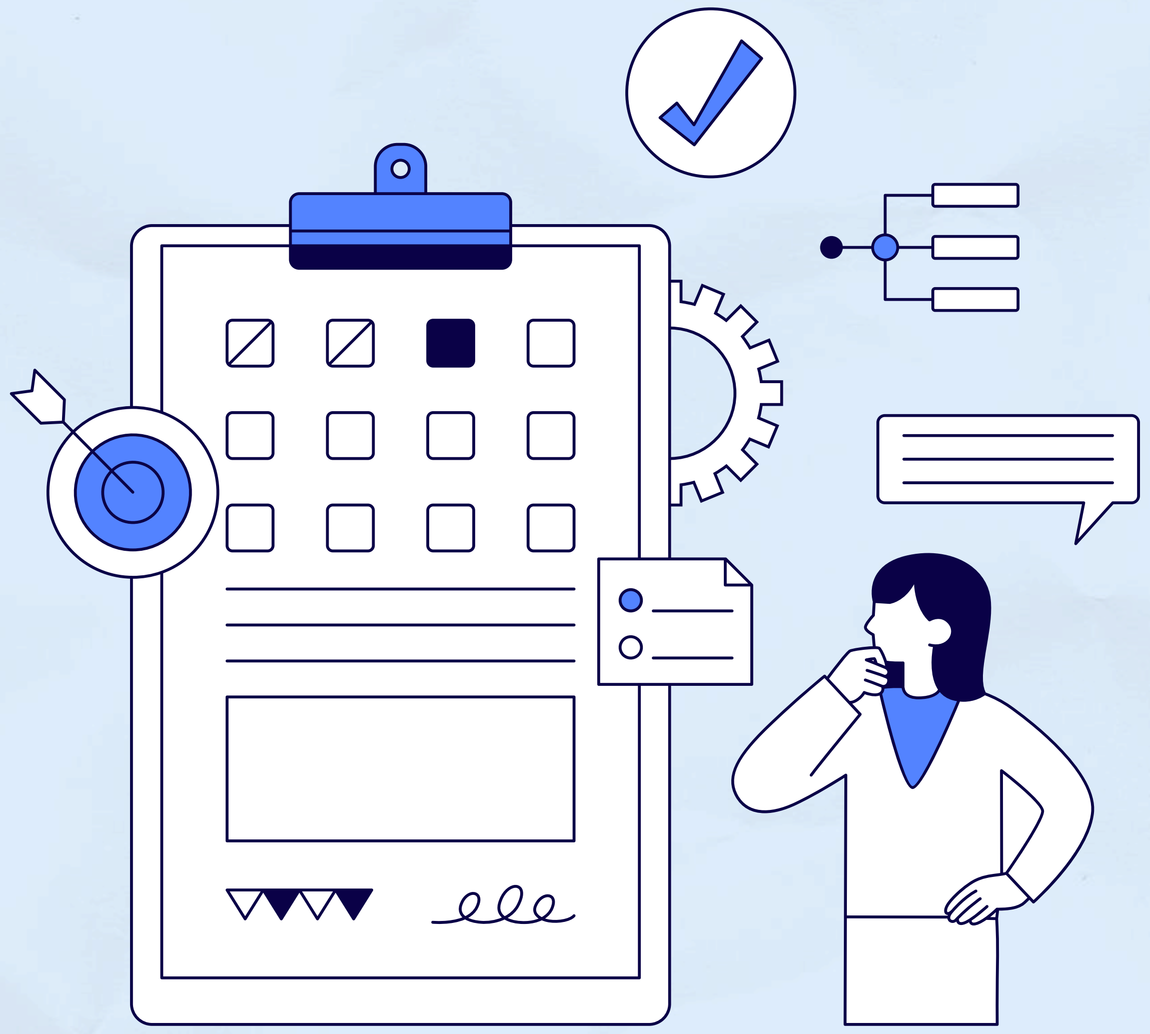


GROUP NO.: 211

ALGORITHMS: WE PROPOSE A HYBRID RECOMMENDATION SYSTEM COMBINING COLLABORATIVE FILTERING, CONTENT-BASED FILTERING, AND DEEP LEARNING MODELS. ADDITIONALLY, WE WILL EXPLORE THE USE OF REINFORCEMENT LEARNING TO OPTIMIZE RECOMMENDATIONS OVER TIME.

EVALUATION METRICS: TO EVALUATE THE SYSTEM, WE WILL MEASURE PERFORMANCE BASED ON:

- MEASURING THE CORRECTNESS OF THE RECOMMENDATIONS.
- ENSURING DIVERSE RECOMMENDATIONS ACROSS THE PRODUCT CATALOG.
- CONDUCTING SURVEYS AND A/B TESTING TO ASSESS USER EXPERIENCE.
- ANALYZING THE SYSTEM'S INTERPRETABILITY AND THE USER'S TRUST IN THE RECOMMENDATIONS PROVIDED.



GROUP NO.: 211

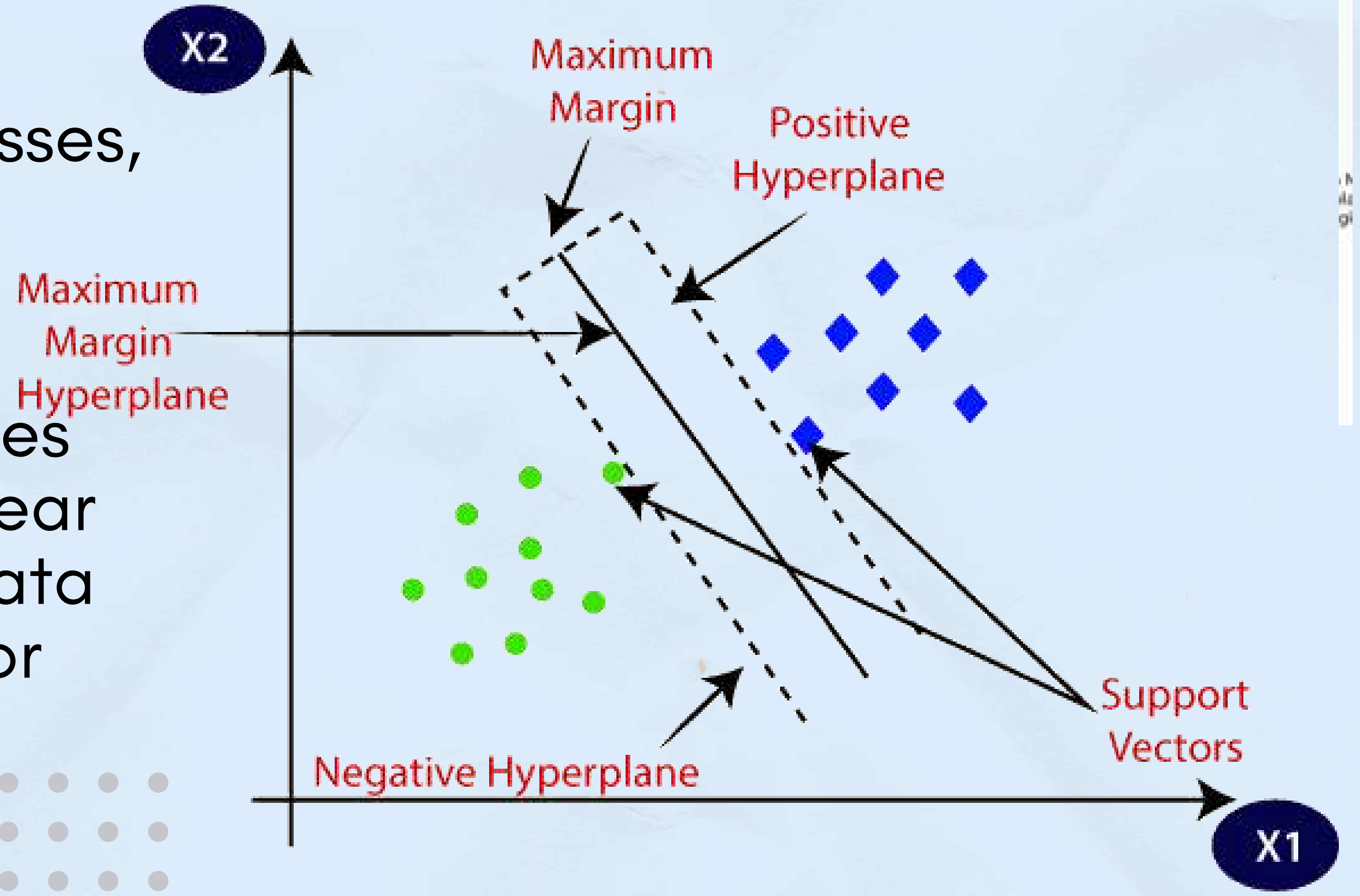
MACHINE LEARNING ALGORITHS

For analyzing and processing the collected data, we have used Support Vector Machine (SVM) and K-Nearest Neighbors (KNN) algorithms to classify and interpret the data effectively.

1.Support Vector Machine (SVM) : It is a powerful supervised learning algorithm primarily used for classification tasks, though it can also be applied to regression. SVM works by finding the optimal hyperplane that best separates data into different classes.

The key idea is to maximize the margin between the classes, which is the distance between the nearest data points (support vectors) of each class to the hyperplane.

This approach minimizes classification error and increases generalization. SVM can handle both linear and non-linear classification problems, using kernel functions to map data into higher-dimensional spaces where a linear separator can be found.

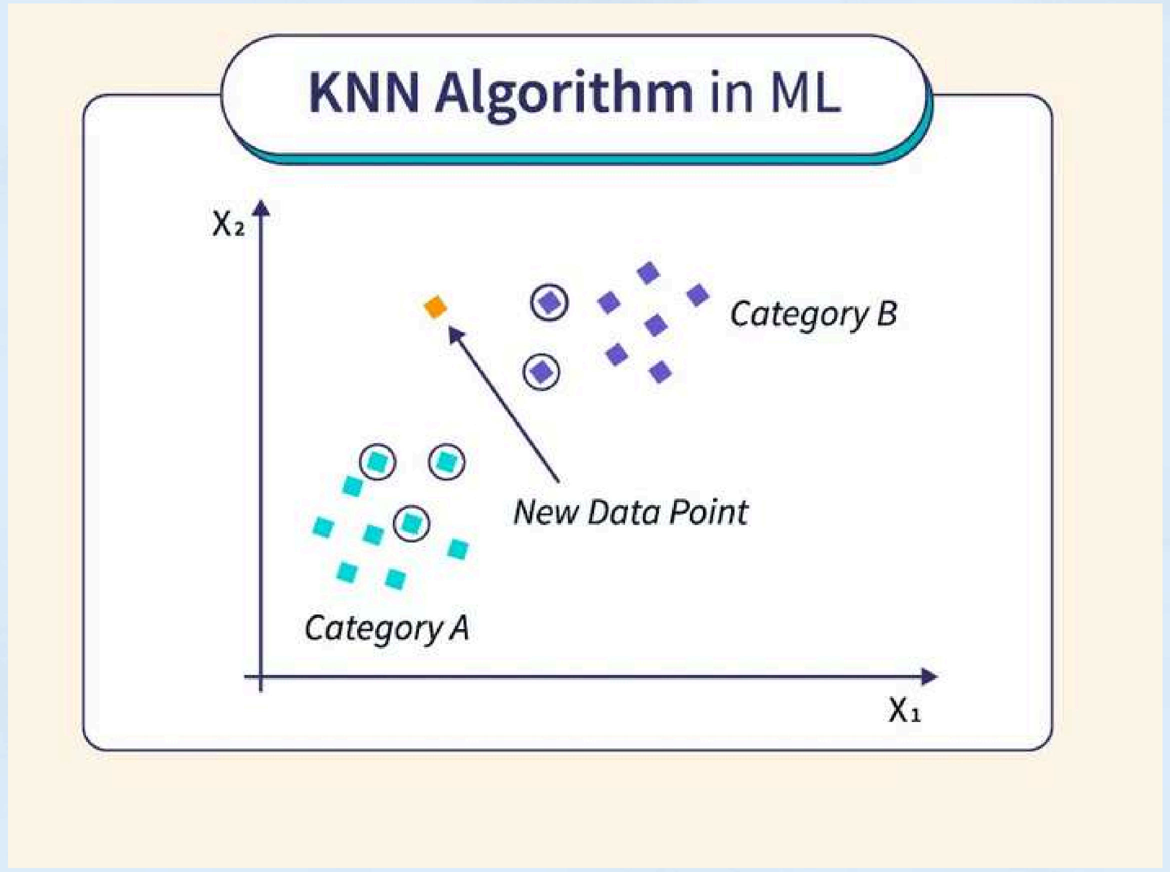


Step 1: Selecting the Kernel Function:
Choose an appropriate kernel (e.g., linear, polynomial, or RBF) based on the data's characteristics.
Step 2: Maximizing the Margin:
Find the hyperplane that maximizes the margin between classes, using support vectors to define the boundary.
Step 3: Training the Model:
Solve an optimization problem to determine the optimal hyperplane by minimizing classification error and maximizing the margin.
Step 4: Classifying New Data:
Classify new points based on which side of the hyperplane they fall on, using the chosen kernel for non-linear data.
Step 5: Fine-Tuning Parameters:
Optimize parameters like regularization (C) and kernel settings (gamma) to improve model performance.

2. K-Nearest Neighbors (KNN): It is a simple, non-parametric, and instance-based learning algorithm used for both classification and regression tasks.

It operates on the principle of identifying the most similar data points, based on distance metrics such as Euclidean or Manhattan distance, to classify or predict the output for a new data point.

the algorithm assigns the most common class among the k-nearest neighbors, whereas in regression, it predicts the average (or weighted average) of the target values of the k-nearest neighbors.



Step 1: Selecting the optimal value of K
K represents the number of nearest neighbors that needs to be considered while making prediction.

Step 2: Calculating distance
To measure the similarity between target and training data points, Euclidean distance is used. Distance is calculated between each of the data points in the dataset and target point.

Step 3: Finding Nearest Neighbors
The k data points with the smallest distances to the target point are the nearest neighbors.

Step 4: Voting for Classification or Taking Average for Regression
In the classification problem, the class labels of K-nearest neighbors are determined by performing majority voting. The class with the most occurrences among the neighbors becomes the predicted class for the target data point.

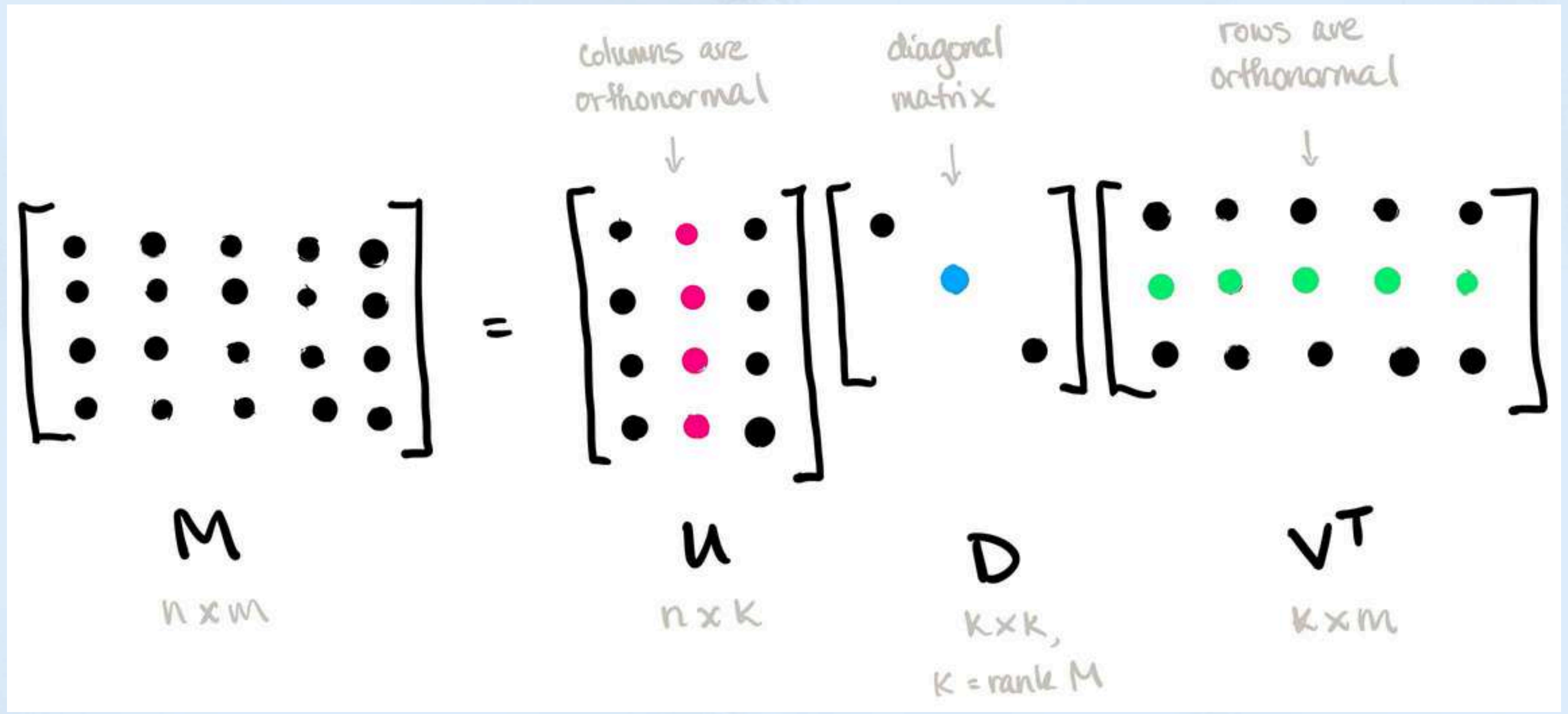
GROUP NO.: 211

MATRIX FACTORIZATION

To validate our accuracy, we implemented matrix factorization techniques like Singular Value Decomposition (SVD) and Non-Negative Matrix Factorization (NMF). These approaches are widely used in areas like recommendation systems and dimensionality reduction, as they effectively break down a matrix into component parts to uncover latent patterns.

1.Singular Value Decomposition : SVD decomposes a matrix A into three components: U , Σ and V^T, where U and V^T are orthogonal matrices, and Σ is a diagonal matrix of singular values.

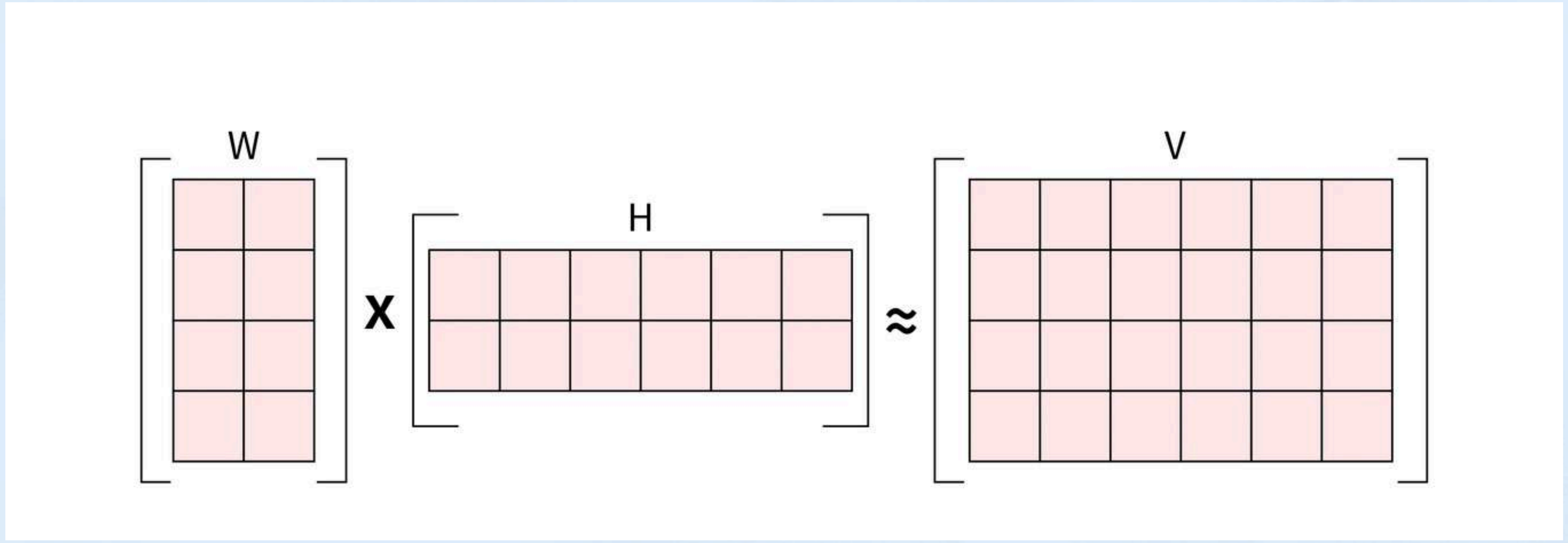
This decomposition enables SVD to capture both positive and negative relationships in the data, making it highly effective in applications like collaborative filtering and dimensionality reduction.



SVD provides a comprehensive representation of the data's structure, often used in recommendation systems and image compression.

2. Non-Negative Matrix Factorization: It factorizes a matrix into two lower-dimensional matrices W and H with the constraint that all elements must be non-negative

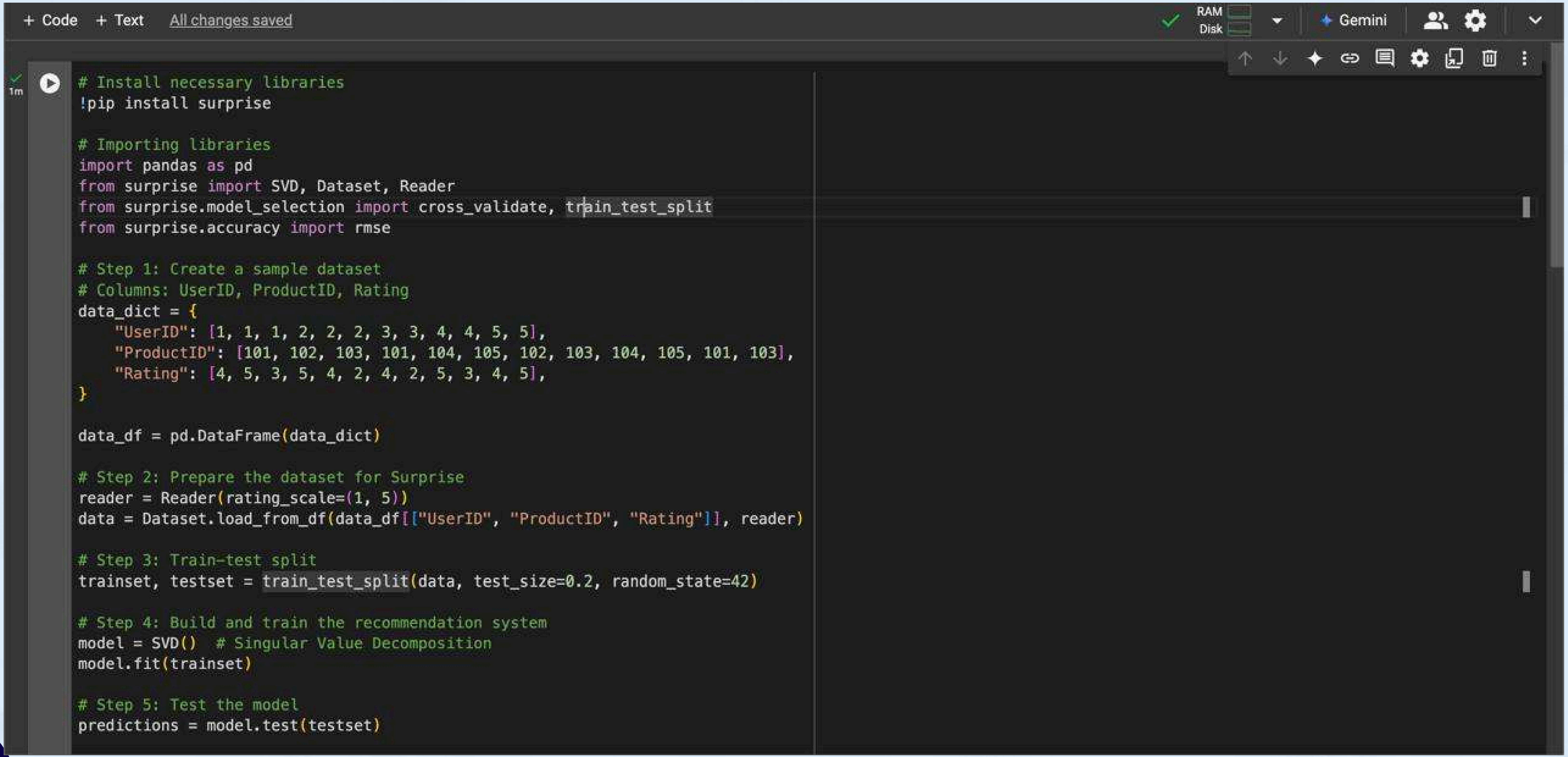
This makes NMF particularly useful when the data has a non-negative nature, such as pixel intensities, user ratings, or topic distributions.



NMF results in additive combinations, often leading to more interpretable factors, which is valuable in applications like text mining and image recognition

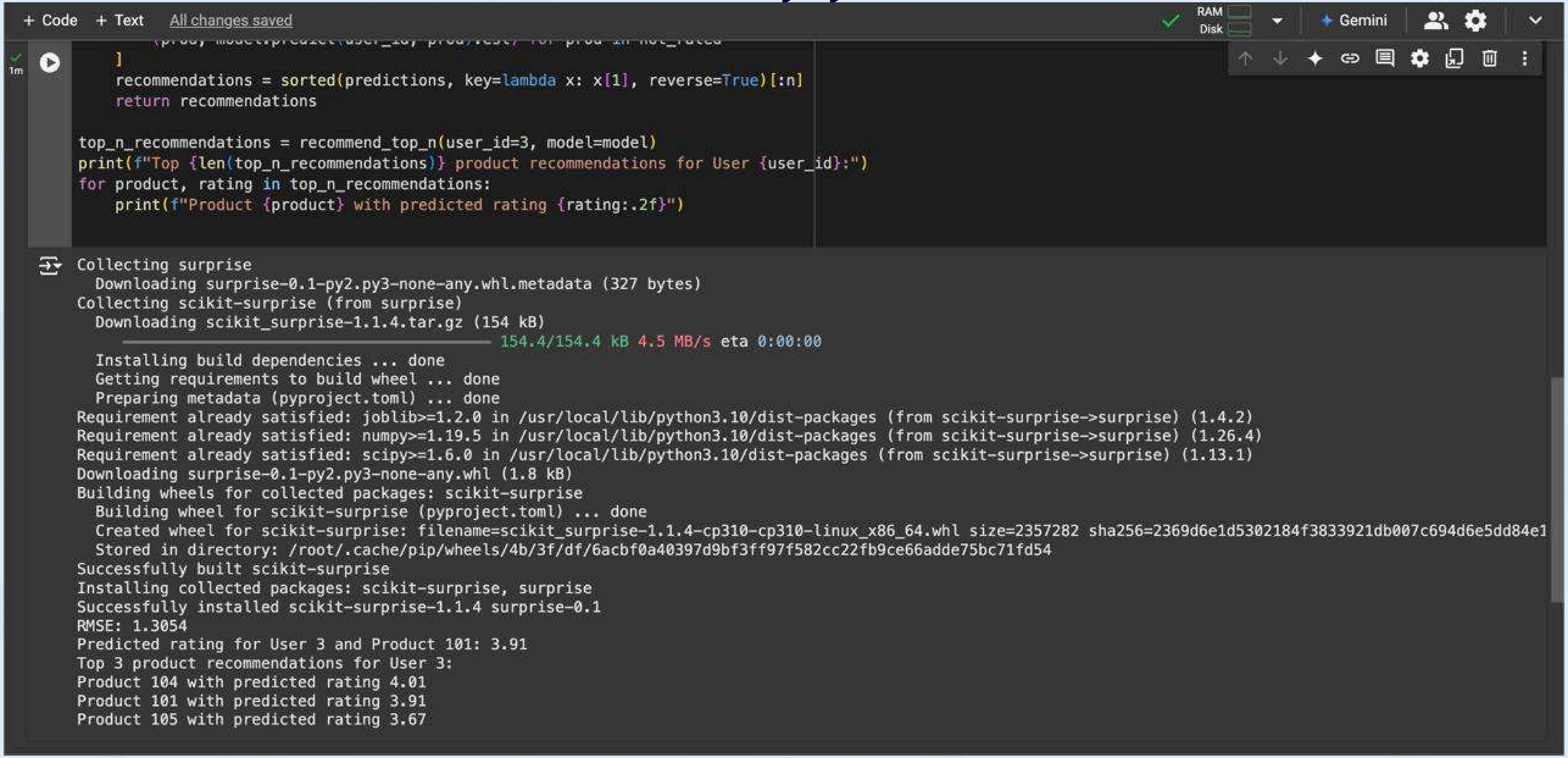
GROUP NO.: 211

CODE



```
# Install necessary libraries
!pip install surprise

# Importing libraries
import pandas as pd
from surprise import SVD, Dataset, Reader
from surprise.model_selection import cross_validate, train_test_split
from surprise.accuracy import rmse

# Step 1: Create a sample dataset
# Columns: UserID, ProductID, Rating
data_dict = {
    "UserID": [1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 5, 5],
    "ProductID": [101, 102, 103, 101, 104, 105, 102, 103, 104, 105, 101, 103],
    "Rating": [4, 5, 3, 5, 4, 2, 4, 2, 5, 3, 4, 5],
}

data_df = pd.DataFrame(data_dict)

# Step 2: Prepare the dataset for Surprise
reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(data_df[["UserID", "ProductID", "Rating"]], reader)

# Step 3: Train-test split
trainset, testset = train_test_split(data, test_size=0.2, random_state=42)

# Step 4: Build and train the recommendation system
model = SVD()  # Singular Value Decomposition
model.fit(trainset)

# Step 5: Test the model
predictions = model.test(testset)
```

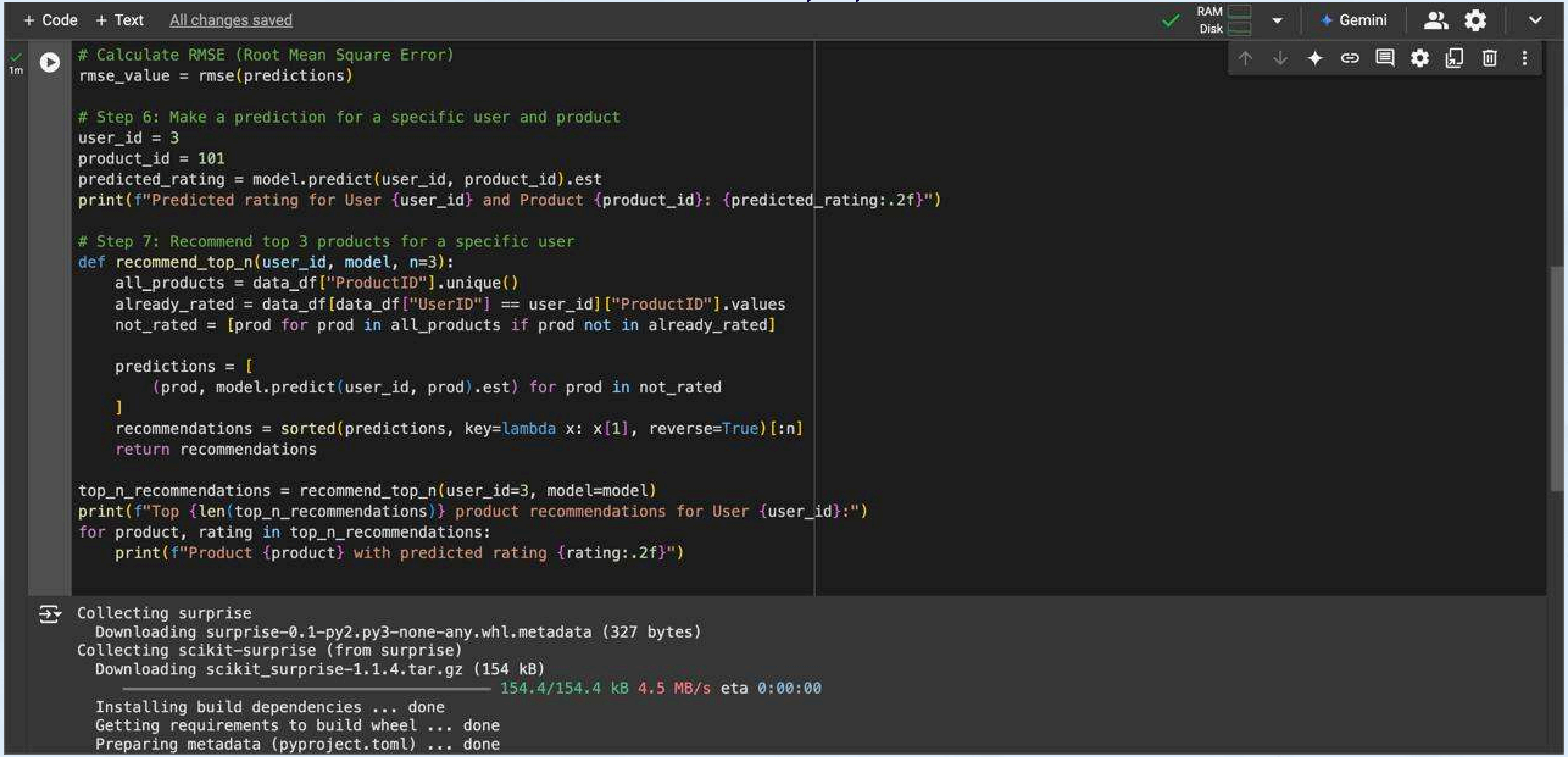
GROUP NO.: 211



```
        (prod, model.predict(user_id, prod).est) for prod in not_rated
    ]
    recommendations = sorted(predictions, key=lambda x: x[1], reverse=True)[:n]
    return recommendations

top_n_recommendations = recommend_top_n(user_id=3, model=model)
print(f"Top {len(top_n_recommendations)} product recommendations for User {user_id}:")
for product, rating in top_n_recommendations:
    print(f"Product {product} with predicted rating {rating:.2f}")
```

```
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl.metadata (327 bytes)
Collecting scikit-surprise (from surprise)
  Downloading scikit_surprise-1.1.4.tar.gz (154 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.4/154.4 kB 4.5 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise->surprise) (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise->surprise) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise->surprise) (1.13.1)
Downloading surprise-0.1-py2.py3-none-any.whl (1.8 kB)
Building wheels for collected packages: scikit-surprise
  Building wheel for scikit-surprise (pyproject.toml) ... done
  Created wheel for scikit-surprise: filename=scikit_surprise-1.1.4-cp310-cp310-linux_x86_64.whl size=2357282 sha256=2369d6e1d5302184f3833921db007c694d6e5dd84e1
  Stored in directory: /root/.cache/pip/wheels/4b/3f/df/6acbf0a40397d9bf3ff97f582cc22fb9ce66adde75bc71fd54
Successfully built scikit-surprise
Installing collected packages: scikit-surprise, surprise
Successfully installed scikit-surprise-1.1.4 surprise-0.1
RMSE: 1.3054
Predicted rating for User 3 and Product 101: 3.91
Top 3 product recommendations for User 3:
Product 104 with predicted rating 4.01
Product 101 with predicted rating 3.91
Product 105 with predicted rating 3.67
```

GROUP NO.: 211



```
# Calculate RMSE (Root Mean Square Error)
rmse_value = rmse(predictions)

# Step 6: Make a prediction for a specific user and product
user_id = 3
product_id = 101
predicted_rating = model.predict(user_id, product_id).est
print(f"Predicted rating for User {user_id} and Product {product_id}: {predicted_rating:.2f}")

# Step 7: Recommend top 3 products for a specific user
def recommend_top_n(user_id, model, n=3):
    all_products = data_df["ProductID"].unique()
    already_rated = data_df[data_df["UserID"] == user_id]["ProductID"].values
    not_rated = [prod for prod in all_products if prod not in already_rated]

    predictions = [
        (prod, model.predict(user_id, prod).est) for prod in not_rated
    ]
    recommendations = sorted(predictions, key=lambda x: x[1], reverse=True)[:n]
    return recommendations

top_n_recommendations = recommend_top_n(user_id=3, model=model)
print(f"Top {len(top_n_recommendations)} product recommendations for User {user_id}:")
for product, rating in top_n_recommendations:
    print(f"Product {product} with predicted rating {rating:.2f}")
```

```
Collecting surprise
  Downloading surprise-0.1-py2.py3-none-any.whl.metadata (327 bytes)
Collecting scikit-surprise (from surprise)
  Downloading scikit_surprise-1.1.4.tar.gz (154 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 154.4/154.4 kB 4.5 MB/s eta 0:00:00
  Installing build dependencies ... done
  Getting requirements to build wheel ... done
  Preparing metadata (pyproject.toml) ... done
```

GROUP NO.: 211

EXPLAINATION OF CODE

Step 1: Install Necessary Libraries*

```
!pip install surprise
```

- The surprise library is a Python library for building recommendation systems. It supports various algorithms, including SVD (used here).
- !pip install ensures that the library is installed in Google Colab.

Step 2: Create a Sample Dataset*

```
data_dict = {
    "UserID": [1, 1, 1, 2, 2, 2, 3, 3, 4, 4, 5, 5],
    "ProductID": [101, 102, 103, 101, 104, 105, 102, 103, 104, 105, 101, 103],
    "Rating": [4, 5, 3, 5, 4, 2, 4, 2, 5, 3, 4, 5],
                    }

data_df = pd.DataFrame(data_dict)
```

- A small dataset is created in dictionary format where:
  - *UserID*: Represents unique users (1, 2, 3, etc.).
  - *ProductID*: Represents products (e.g., 101, 102, etc.).
  - *Rating*: Ratings (1-5) given by users to products.
- This dataset is converted to a pandas DataFrame (data_df) for easy manipulation.

GROUP NO.: 211

Step 3: Prepare the Dataset for surprise**

reader = Reader(rating_scale=(1, 5))
data = Dataset.load_from_df(data_df[["UserID", "ProductID", "Rating"]], reader)

- Reader: Specifies the range of ratings (1 to 5 here).
- Dataset.load_from_df: Converts the pandas DataFrame into a format compatible with the surprise library.

Step 4: Train-Test Split*

trainset, testset = train_test_split(data, test_size=0.2, random_state=42)

- Splits the data into:
  - *Trainset*: 80% of the data used to train the recommendation model.
  - *Testset*: 20% of the data used to evaluate the model's performance.
- random_state=42: Ensures consistent splitting for reproducibility.

Step 5: Build and Train the Recommendation System*

```
model = SVD()  # Singular Value Decomposition
model.fit(trainset)
```

- SVD (Singular Value Decomposition)*: A collaborative filtering technique that reduces the user-item interaction matrix into smaller dimensions, capturing important patterns.
- model.fit(trainset): Trains the model on the trainset.

Step 6: Test the Model*

```
predictions = model.test(testset)
rmse_value = rmse(predictions)
```

- model.test(testset): Predicts ratings for the test data.
- rmse(predictions): Calculates the Root Mean Square Error (RMSE), which measures the accuracy of predicted ratings.

Step 7: Make Predictions for a Specific User and Product*

```
user_id = 3
product_id = 101
predicted_rating = model.predict(user_id, product_id).est
print(f"Predicted rating for User {user_id} and Product {product_id}:
{predicted_rating:.2f}")
```

- model.predict(user_id, product_id): Predicts the rating for a specific user (user_id=3) and product (product_id=101).
- .est: Extracts the predicted rating value.
- The predicted rating is printed.

Step 8: Recommend Top N Products for a Specific User*

```
def recommend_top_n(user_id, model, n=3):
    all_products = data_df["ProductID"].unique()
    already_rated = data_df[data_df["UserID"] == user_id]
["ProductID"].values
    not_rated = [prod for prod in all_products if prod not in
already_rated]

    predictions = [
        (prod, model.predict(user_id, prod).est) for prod in not_rated
    ]
    recommendations = sorted(predictions, key=lambda x: x[1],
reverse=True)[:n]
    return recommendations
```

GROUP NO.: 211

OUTPUT FOR TOP N RECOMMENDATION

```
top_n_recommendations = recommend_top_n(user_id=3, model=model)

print(f"Top {len(top_n_recommendations)} product recommendations for
User {user_id}:")
for product, rating in top_n_recommendations:
    print(f"Product {product} with predicted rating {rating:.2f}")
```

GROUP NO.: 211

DL

CNN

- A CONVOLUTIONAL NEURAL NETWORK (CNN) IS A TYPE OF DEEP LEARNING MODEL PARTICULARLY EFFECTIVE FOR PROCESSING DATA WITH A GRID-LIKE STRUCTURE, SUCH AS IMAGES. IT IS COMPOSED OF LAYERS THAT AUTOMATICALLY AND ADAPTIVELY LEARN SPATIAL HIERARCHIES OF FEATURES FROM INPUT DATA, MAKING IT HIGHLY EFFECTIVE FOR IMAGE AND PATTERN RECOGNITION TASKS. IN RECOMMENDATION SYSTEMS, CNNS CAN BE ADAPTED TO ANALYZE VISUAL ASPECTS OF ITEMS (E.G., PRODUCT IMAGES) OR SEQUENTIAL PATTERNS, CONTRIBUTING TO MORE PERSONALIZED AND CONTEXT-AWARE RECOMMENDATIONS.
- BASIC STRUCTURE OF A CNN
- CONVOLUTIONAL LAYERS: THESE LAYERS APPLY FILTERS TO THE INPUT DATA TO CREATE FEATURE MAPS, CAPTURING PATTERNS LIKE EDGES, TEXTURES, OR MORE COMPLEX SHAPES.
- POOLING LAYERS: USED TO DOWN-SAMPLE FEATURE MAPS, REDUCING THEIR DIMENSIONS AND THE COMPUTATIONAL LOAD WHILE RETAINING ESSENTIAL FEATURES.
- FULLY CONNECTED LAYERS: THESE LAYERS COMBINE THE LEARNED FEATURES TO MAKE FINAL PREDICTIONS OR CLASSIFICATIONS.
- ACTIVATION FUNCTIONS (E.G., RELU): APPLIED TO INTRODUCE NON-LINEARITY, HELPING THE MODEL LEARN COMPLEX PATTERNS.
- KEY POINTS OF CNN FOR RECOMMENDATIONS
- VISUAL CONTENT ANALYSIS: CNNS CAN ANALYZE PRODUCT IMAGES TO LEARN FEATURES THAT REPRESENT VISUAL APPEAL, AIDING IN FASHION, RETAIL, AND OTHER VISUAL-CENTRIC RECOMMENDATIONS.
- CONTEXT AWARENESS: CNNS CAN LEARN PATTERNS IN USER INTERACTIONS OR PRODUCT DATA, IMPROVING RECOMMENDATION QUALITY BASED ON CONTEXTUAL RELEVANCE.

GROUP NO.: 211

RNN

- A RECURRENT NEURAL NETWORK (RNN) IS A DEEP LEARNING MODEL PARTICULARLY WELL-SUITED FOR SEQUENCE DATA, SUCH AS TEXT, TIME-SERIES, OR ANY DATA WHERE ORDER MATTERS. UNLIKE TRADITIONAL NEURAL NETWORKS, RNNS HAVE CONNECTIONS THAT LOOP BACK, ALLOWING THEM TO RETAIN INFORMATION FROM PREVIOUS STEPS IN THE SEQUENCE. THIS MAKES THEM EFFECTIVE FOR TASKS WHERE UNDERSTANDING CONTEXT OR PATTERNS OVER TIME IS ESSENTIAL.
- BASIC STRUCTURE OF AN RNN
- RECURRENT LAYERS: THESE LAYERS PROCESS ONE ELEMENT OF THE SEQUENCE AT A TIME, PASSING INFORMATION FROM ONE STEP TO THE NEXT, ALLOWING THE NETWORK TO "REMEMBER" PREVIOUS INPUTS.
- HIDDEN STATE: THE RNN MAINTAINS A HIDDEN STATE, WHICH STORES INFORMATION LEARNED FROM PREVIOUS STEPS IN THE SEQUENCE.
- OUTPUT LAYER: PROVIDES PREDICTIONS BASED ON THE SEQUENCE AND ACCUMULATED INFORMATION IN THE HIDDEN STATE.
- KEY POINTS OF RNNS FOR RECOMMENDATIONS
- SEQUENTIAL DATA PROCESSING: RNNS ARE GREAT FOR UNDERSTANDING USER BEHAVIOR OVER TIME, MAKING THEM EFFECTIVE FOR SESSION-BASED OR TIME-DEPENDENT RECOMMENDATIONS.
- PERSONALIZED SUGGESTIONS: RNNS CAN MODEL USER PREFERENCES BASED ON RECENT INTERACTIONS, WHICH HELPS MAKE RELEVANT AND TIMELY RECOMMENDATIONS.

GROUP NO.: 211

REFRENCES

https://scholar.google.com/scholar?q=e+commerce+recommendation&hl=en&as_sdt=0,5#d=gs_qabs&t=1728478576568&u=%23p%3DxxVtnVrtb2cJ
https://scholar.google.com/scholar?q=e+commerce+recommendation&hl=en&as_sdt=0,5#d=gs_qabs&t=1728478624517&u=%23p%3DLeQsPFQ5sn8J
https://iopscience.iop.org/article/10.1088/1742-6596/1897/1/012024/pdf
https://www.mdpi.com/2071-1050/13/19/10786
https://dl.acm.org/doi/pdf/10.1145/336992.337035
https://link.springer.com/article/10.1007/s12652-018-0928-7
https://ieeexplore.ieee.org/abstract/document/4280214
https://ieeexplore.ieee.org/abstract/document/9118884

GROUP NO.: 211

THANK YOU

Group Members:

K Dheeraj: 23BCE10723
Vibhav Pandey: 23BCE11095
Devraj Bijpurai: 23BCE11169
Arpita Nanda: 23BCE11660
Faiza Farzana: 23BCE11794