Fall 2024

CMPE-256

Advanced Data Mining

Project Report

Hybrid Product Recommendation System

Team TriLogic

Presented to :
Mr. Chandrasekar Vuppalapati

Group Members :
Chandini Saisri Uppuganti (ID : 018228483)
Soumya Bharathi Vetukuri (ID: 016668964)
Siri Batchu (ID : 018238545)

Project Description

Problem Statement

In the rapidly growing e-commerce sector, personalized product recommendations have become a crucial tool for enhancing user experience and driving sales. However, existing recommendation systems often face challenges such as the **cold start problem**, where new users or products lack sufficient interaction data, and **data sparsity**, where limited user-product interactions lead to suboptimal recommendations. Moreover, traditional methods like collaborative filtering or content-based filtering, when used in isolation, fail to provide comprehensive insights into user preferences and product attributes.

Solution

To address these challenges, there is a need for a robust, scalable, and efficient hybrid recommendation system that integrates multiple recommendation techniques. Such a system should leverage both user behaviour (collaborative filtering) and product characteristics (content-based filtering) to deliver highly accurate and relevant product suggestions.

What is Recommendation System?

A Recommendation System (also known as a Recommender System) is a type of software system designed to suggest relevant items (such as products, services, content, or information) to users based on various criteria. These systems aim to enhance user experience by personalizing content, helping users discover items they might not have found otherwise.

- Collaborative Filtering - Suggests items to users based on the preferences of other users.
- Content-Based Filtering - Recommends items to users based on the characteristics of the items and the user's preferences.
- Hybrid Method - Combination of both collaborative and content based filtering

Objective

This project aims to design and implement a hybrid recommendation system that combines the strengths of collaborative filtering and content-based filtering techniques. By integrating these approaches, the system will overcome common challenges such as the cold start problem, sparsity in user-item interactions, and limited personalization, thus providing tailored recommendations to users in an e-commerce environment. The hybrid model ensures a holistic understanding of user preferences by leveraging both user interaction patterns and product attributes. The dataset used for this project is sourced from the Olist E-commerce platform, containing comprehensive information about customer orders, product details, and reviews. This dataset enables us to model user behavior effectively and derive insights into product trends. The preprocessing steps include cleaning, encoding, and feature engineering, where derived features such as purchase frequency and textual embeddings using TF-IDF enhance the predictive power of the system. For model development, we use Singular Value Decomposition (SVD) for collaborative filtering to identify latent patterns in user-item matrices. Content-based filtering is implemented using TF-IDF to capture textual similarities between product descriptions. These models are integrated using a machine learning-based hybrid approach, such as Random Forest, to provide more accurate and relevant recommendations. The model is evaluated on standard metrics like RMSE and precision to ensure high performance.

Dataset

While there have been notable datasets for e-commerce product recommendations, this dataset stands out by offering comprehensive details on customer orders, product metadata, and user reviews. The dataset includes data for 100,000 orders across various product categories, sourced from a large-scale online marketplace. Each order captures product details, customer information, and reviews, providing a rich foundation for analyzing purchasing behavior and building recommendation systems.

The dataset features unique IDs for both products and customers. Product IDs span tens of thousands of items across categories such as electronics, health, and home decor, while customer IDs represent individual users from diverse geographical regions. Each order includes metadata such as product name length, description length, number of photos, and review scores, with ratings ranging from 1 to 5. On average, products have a substantial number of reviews, though some lack detailed comments.

Additional features include timestamps for various stages of the order lifecycle (purchase, approval, delivery), customer demographics (city, state, zip code prefix), and review attributes (comment title, message). While some review fields have missing data, the dataset remains robust for exploring user-item interactions and modelling personalized recommendations.

https://www.kaggle.com/datasets/olistbr/brazilian-ecommerce

Requirements

1. Functional Requirements:

Recommendation Features:

- Provide personalized product recommendations to users based on their interaction history and product attributes.
- Address the cold start problem by integrating collaborative and content-based filtering techniques.
- Handle scenarios for both new and existing users effectively.

Hybrid Model Implementation:

- Use collaborative filtering techniques like Singular Value Decomposition (SVD) for analyzing user-item interaction matrices.
- Apply content-based filtering using Term Frequency-Inverse Document Frequency (TF-IDF) for analyzing product descriptions.
- Combine the predictions from both approaches using a hybrid integration model (e.g., Random Forest).

Evaluation Metrics:

- Validate the model using metrics like Root Mean Squared Error (RMSE).
- Perform n fold cross-validation to ensure robustness.

Data Handling:

- Preprocess and clean raw data, including handling missing values, encoding categorical variables, and normalizing numerical data.
- Engineer features from transactional and textual data to improve model performance.

2. Non-Functional Requirements:

Scalability:

- The system should handle a large-scale dataset (~100k orders) efficiently.
- Support real-time recommendation generation for end-users.

Modularity:

- Code should be structured in reusable modules for preprocessing, model training, evaluation, and prediction.

Extensibility:

- The system should allow easy integration of additional algorithms or features in the future.

Performance:

- Ensure low latency for real-time recommendations.
- Optimize model training and prediction processes for faster execution.

3. Dataset Requirements:

Data Sources:

- Use the Olist e-commerce dataset, which contains details of orders, customer demographics, product attributes, and reviews.

Data Features:

- Include key features such as order_id, product_id, customer_id, product_category_name, review_score, and textual product descriptions.

4. Software and Tools Requirements:

Programming Language:

- Python for implementation.

Libraries:

- Data processing: Pandas, NumPy.
- Visualization: Matplotlib, Seaborn.
- Machine Learning: Scikit-learn, Surprise (for SVD), Random Forest.

- Text Processing: Scikit-learn (TF-IDF).

Platform and Environment:

- Google Colab / Jupyter Notebook for development and testing.

5. Business Requirements:

Comprehensive User Understanding:

- Combine collaborative and content-based ltering to gain a holistic view of user preferences, enabling more accurate recommendations.

Model Building:

- Develop a hybrid model to deliver personalized product suggestions that resonate with individual user preferences.

Cold Start Problem Mitigation:

- Address challenges related to new users or items by integrating multiple recommendation approaches, ensuring relevant suggestions despite limited initial data.

KDD (Knowledge Discovery in Databases)

1. Data Selection

- The dataset used is sourced from the Olist Brazilian E-commerce dataset, containing transactional, customer, and product-related data.
- The dataset was loaded using libraries like Pandas, focusing on key features like order_id, product_id, review_score, and more.

2. Data Preprocessing

Data Cleaning:

- Missing values were addressed in specific columns, such as removing rows with missing product details.
- Duplicate entries were checked and removed where necessary.

Encoding and Transformation:

- Categorical features like product_category_name were encoded to numerical formats.
- Text data (e.g., product descriptions) was transformed using **TF-IDF** for downstream content-based filtering.

Data Aggregation:

- The dataset was grouped by users and products to calculate summary statistics like total purchases or average ratings.

3. Data Transformation

Feature Engineering:

- Features like average product ratings and number of reviews were derived to enhance the recommendation process.
- Textual features from product descriptions were extracted and transformed using **TF-IDF vectorization** to identify product similarities.

Dimensionality Reduction:

- Collaborative filtering used **Singular Value Decomposition (SVD)** to reduce the dimensionality of the user-item interaction matrix.

4. Data Mining

Collaborative Filtering:

- Implemented using **SVD** to predict missing values in the user-item matrix, identifying latent features in user-product interactions.

Content-Based Filtering:

- Used **TF-IDF** to compute product similarity based on descriptions and user preferences.

Hybrid Model:

- Results from collaborative and content-based filtering were combined using **Random Forest**, which integrates predictions from both models.

5. Pattern Evaluation

Evaluation Metrics:

- Collaborative filtering was evaluated using RMSE to assess prediction accuracy for user ratings.
- The hybrid model's performance was compared using precision, recall, and F1 scores.

Visualization:

- Plots like the correlation matrix, distribution of ratings, and product popularity trends were generated for exploratory analysis and pattern evaluation.

6. Knowledge Representation

Visualizations:

- Visualizations were created to represent user purchase behaviors, popular product categories, and rating distributions.

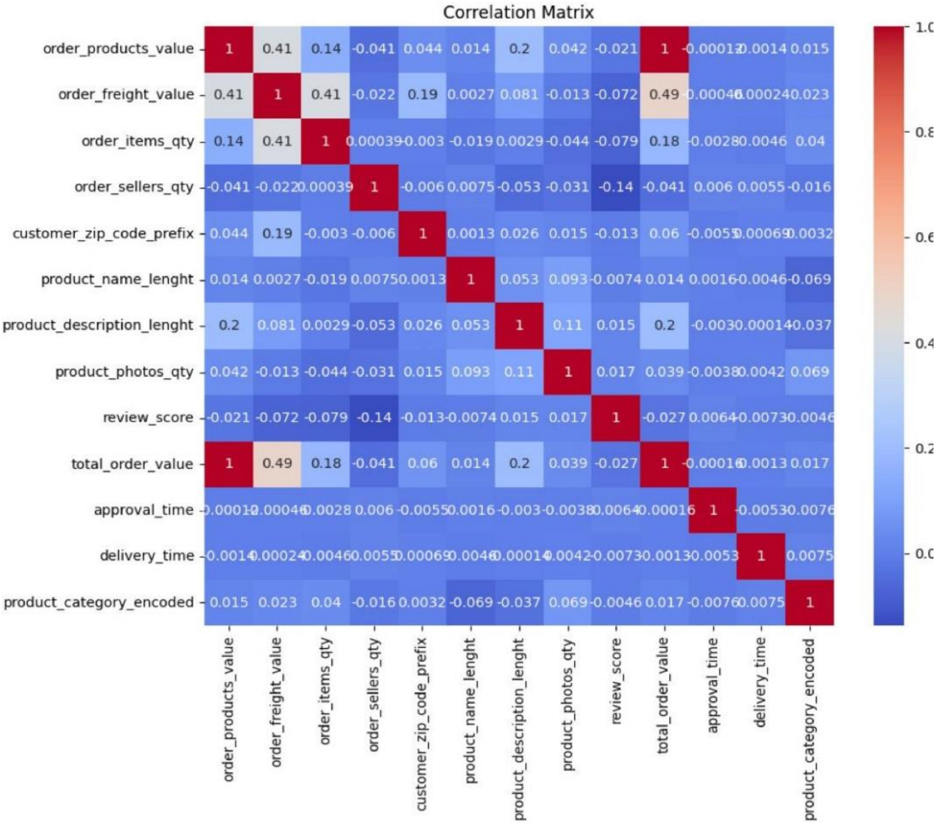
Model Results:

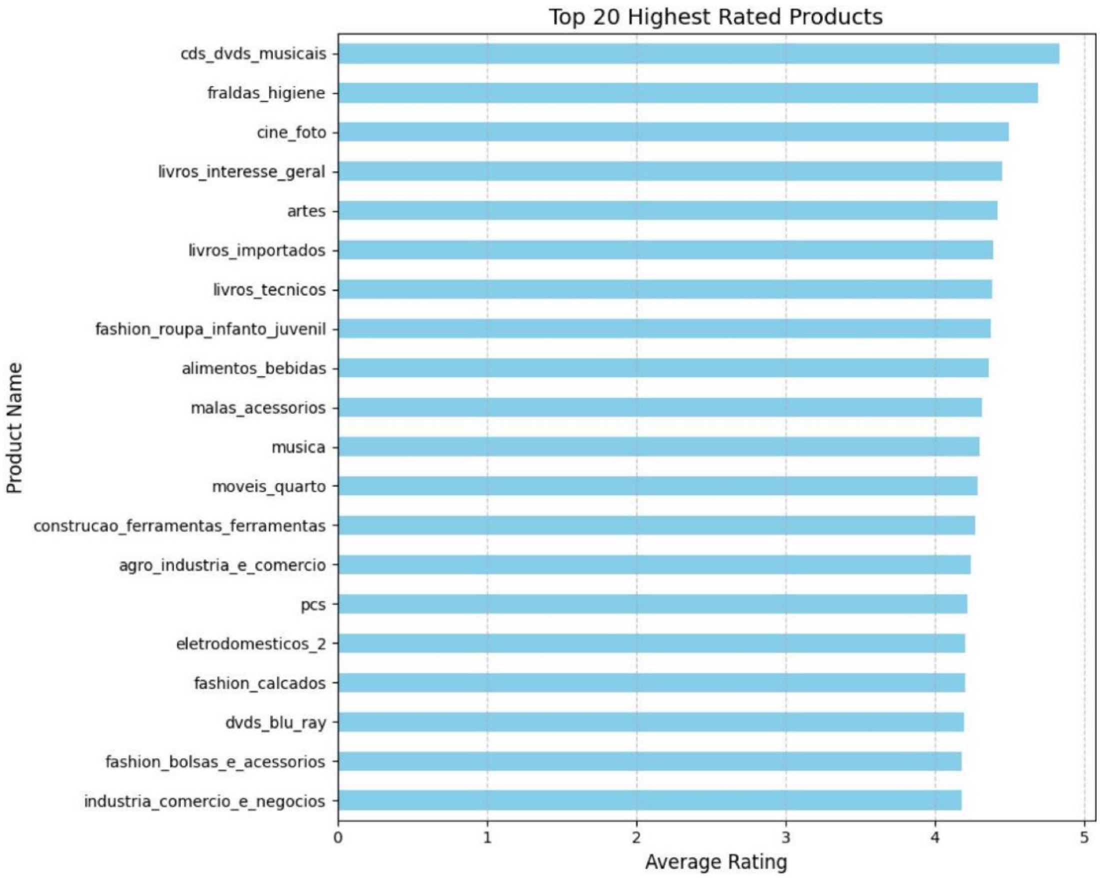
- Outputs from the models were saved and visualized, showing top product recommendations for new and existing users.

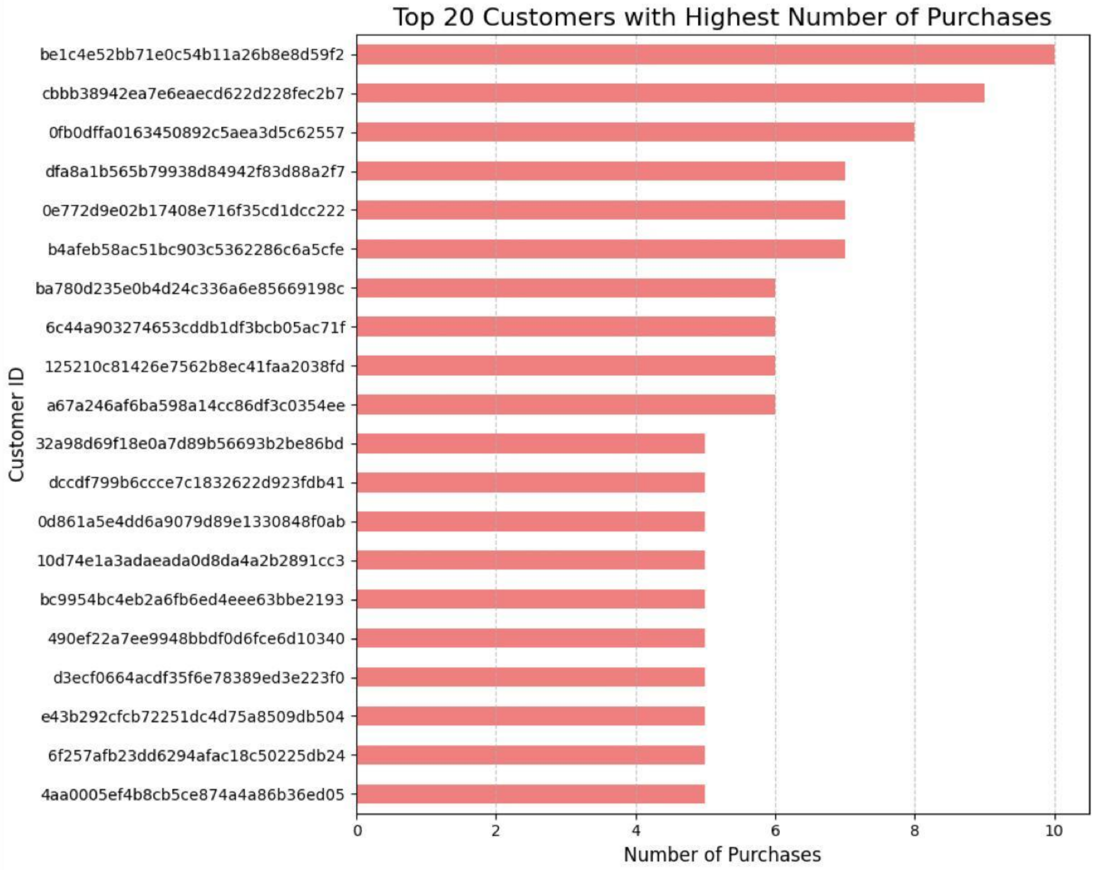
Insights:

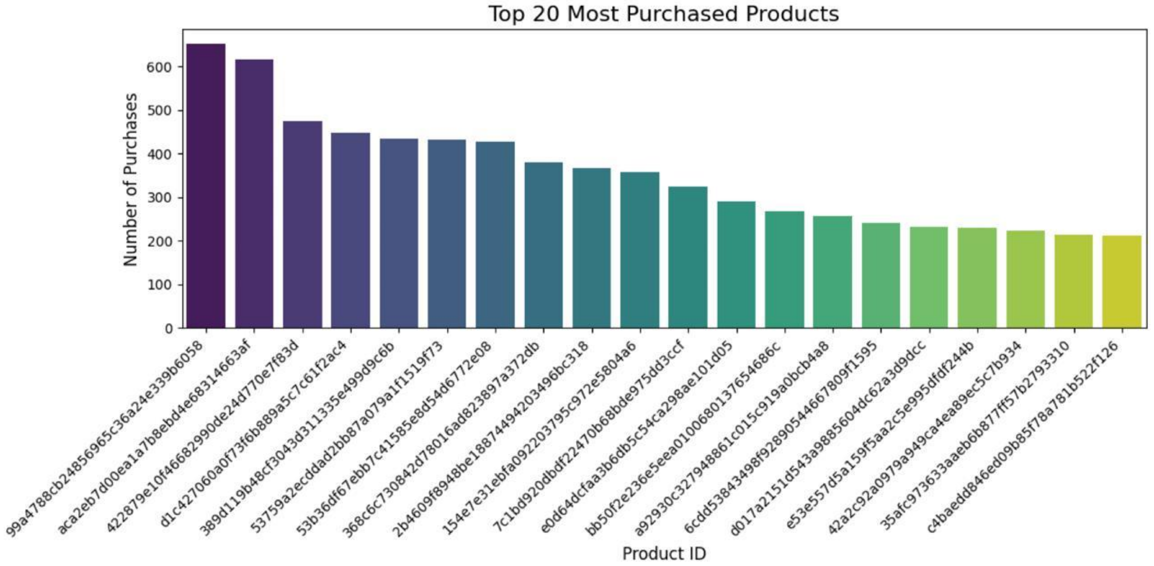
- Insights from the evaluation were discussed in the presentation slides, highlighting the hybrid model’s ability to mitigate the cold start problem.

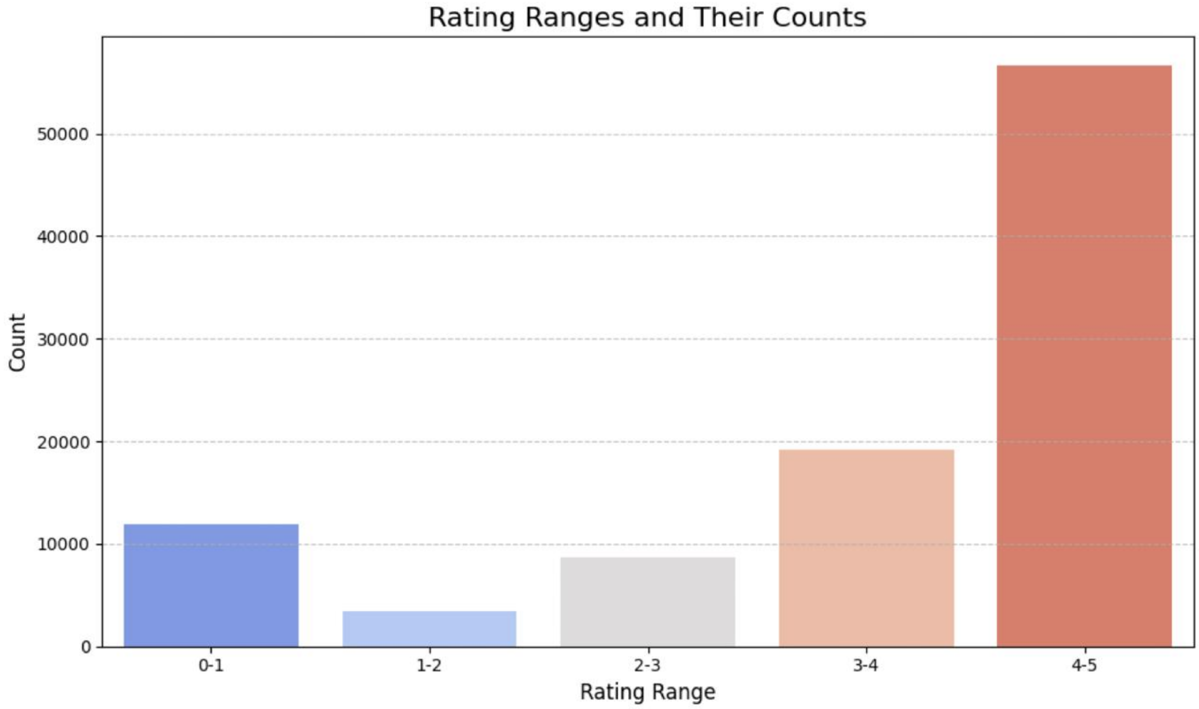
Visualizations:




Top 20 Highest Rated Products
Product Name
cds_dvds_musicais
fraldas_higiene
cine_foto
livros_interesse_geral
artes
livros_importados
livros_tecnicos
fashion_roupa_infanto_juvenil
alimentos_bebidas
malas_acessorios
musica
moveis_quarto
construcao_ferramentas_ferramentas
agro_industria_e_comercio
pcs
eletrodomesticos_2
fashion_calcados
dvds_blu_ray
fashion_bolsas_e_acessorios
industria_comercio_e_negocios
0
1
2
3
4
5
Average Rating


Top 20 Customers with Highest Number of Purchases
Customer ID
be1c4e52bb71e0c54b11a26b8e8d59f2
cbbb38942ea7e6eaecd622d228fec2b7
0fb0dffa0163450892c5aea3d5c62557
dfa8a1b565b79938d84942f83d88a2f7
0e772d9e02b17408e716f35cd1dcc222
b4afeb58ac51bc903c5362286c6a5cfe
ba780d235e0b4d24c336a6e85669198c
6c44a903274653cddb1df3bcb05ac71f
125210c81426e7562b8ec41faa2038fd
a67a246af6ba598a14cc86df3c0354ee
32a98d69f18e0a7d89b56693b2be86bd
dccdf799b6ccce7c1832622d923fdb41
0d861a5e4dd6a9079d89e1330848f0ab
10d74e1a3adaeada0d8da4a2b2891cc3
bc9954bc4eb2a6fb6ed4eee63bbe2193
490ef22a7ee9948bbdf0d6fce6d10340
d3ecf0664acdf35f6e78389ed3e223f0
e43b292cfcb72251dc4d75a8509db504
6f257afb23dd6294afac18c50225db24
4aa0005ef4b8cb5ce874a4a86b36ed05
0
2
4
6
8
10
Number of Purchases





Feature Engineering

Feature engineering is a critical step in building a robust recommendation system. The provided code performs several feature engineering tasks to enhance the quality of the input data and improve the performance of the hybrid recommendation system. Below are the key feature engineering techniques and transformations applied:

1. Handling Numerical Features

- **Goal:** Extract meaningful patterns from numerical data.
- **Actions Performed:**

    - **Review Score Aggregation:**
        - Average product ratings were calculated by grouping data by product_id. This helped represent a product's overall popularity or quality based on customer feedback.
    - **Purchase Count:**
        - Total purchases for each product were derived by counting the number of interactions (order_id) associated with the product_id.
    - **User Interaction Statistics:**
        - The number of products purchased by each user was computed to understand user behavior and preferences.

2. Handling Categorical Features

- **Goal:** Convert categorical data into a numerical format suitable for machine learning.
- **Actions Performed:**
    - **One-Hot Encoding:**
        - Product categories (product_category_name) were encoded into numerical format to ensure compatibility with machine learning models.
    - **Frequency Encoding:**
        - For certain features, their frequency of occurrence was used as a proxy to represent their importance.

3. Textual Features:

- **TF-IDF Transformation:**
    - **Purpose:** To represent textual data from product descriptions as numerical vectors that capture the importance of words in the context of other products.
    - **Implementation:**
        - Extracted textual data from the product_category_name and product descriptions.
        - Applied **TF-IDF (Term Frequency-Inverse Document Frequency)** to compute the relevance of terms for each product.
        - Generated feature vectors to calculate similarity scores between products for content-based filtering.

4. User-Item Interaction Features:

- **User Purchase Frequency:**
  - **Purpose:** To identify the most active customers and understand their behavior patterns.
  - **Implementation:**
    - Aggregated the dataset by customer_id to calculate the total number of purchases for each user.
- **Product Popularity:**
  - **Purpose:** To determine the most popular products across all users.
  - **Implementation:**
    - Counted the number of times each product (product_id) was purchased.

5. Review-Based Features:

- **Average Review Score:**
  - **Purpose:** To capture product quality as perceived by customers.
  - **Implementation:**
    - Aggregated the review_score for each product to compute average ratings.
    - Created a feature representing the average score for every product_id.
- **Rating Distributions:**
  - **Purpose:** To understand customer satisfaction trends.
  - **Implementation:**
    - Calculated the distribution of ratings (e.g., percentage of 5-star, 4-star ratings) for each product to identify quality patterns.

6. Matrix-Based Features for Collaborative Filtering:

- **User-Item Matrix:**
  - **Purpose:** To identify latent patterns in user behavior.
  - **Implementation:**
    - Constructed a sparse matrix with customer_id as rows and product_id as columns.
    - Filled matrix cells with review_score values, representing the interactions between users and products.

- **Latent Features using SVD:**
    - **Purpose:** To reduce dimensionality and discover hidden factors that influence user-product interactions.
    - **Implementation:**
        - Applied **Singular Value Decomposition (SVD)** to the user-item matrix, extracting latent features for collaborative filtering.

7. Combined Features for Hybrid Model:

- **Purpose:** To integrate collaborative and content-based filtering results.
- **Implementation:**
    - Combined user-item interaction features (from SVD) with content-based similarity scores (from TF-IDF).
    - Created a composite feature set to train the hybrid model, ensuring both user behavior and product characteristics are considered.

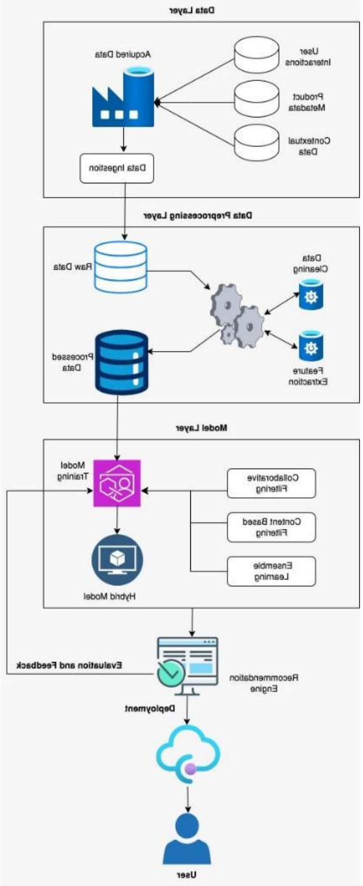
8. Data Normalization

- **Goal:** Scale features to ensure uniformity and stability in model training.
- **Actions Performed:**
    - Normalized numerical features (e.g., review_score, product frequency) to a common scale to avoid bias toward features with larger ranges.

9. Feature Selection

- **Goal:** Retain only the most impactful features to reduce dimensionality and improve model efficiency.
- **Actions Performed:**
    - Features like average ratings, frequency of purchase, and TF-IDF vectors were prioritized for model input.
    - Irrelevant or redundant features were excluded to optimize model performance.

High Level Architecture Design



The diagram represents the architecture of a **Hybrid Recommendation System**, showing the flow of data and operations across different layers. Here's a detailed explanation:

1. Data Layer

The data layer is responsible for collecting and storing the necessary information for the recommendation system.

- **Components:**
  - **User Interaction Data:** Records user actions such as purchases, clicks, reviews, and ratings.
  - **Product Metadata:** Contains details about the products, such as category, description, and features.
  - **Contextual Data:** Includes additional information like time, location, or device used during user interactions.
- **Processes:**
  - **Acquired Data:** Gathers raw data from various sources like transactional logs, user interactions, and product catalogs.
  - **Data Ingestion:** Organizes and loads the collected data into storage systems for further processing.

2. Data Preprocessing Layer

This layer prepares raw data for modelling by cleaning, transforming, and enhancing it with meaningful features.

- **Processes:**
    - **Data Cleaning:** Handles missing values, removes duplicates, and standardizes formats to ensure high-quality data.
    - **Feature Extraction and Engineering:** Derives additional features such as:
        - User activity levels (e.g., purchase frequency).
        - Product popularity metrics (e.g., average ratings).
        - Text embeddings using techniques like TF-IDF for product descriptions.
    - **Processed Data Storage:** Stores the cleaned and transformed data in a structured format for model training.

3. Model Layer

This layer is responsible for training and integrating different recommendation algorithms to create a hybrid system.

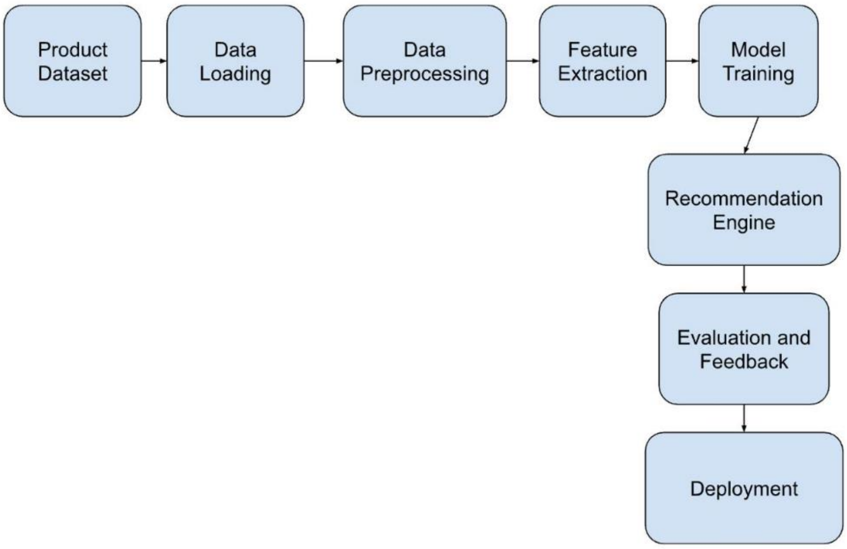
- **Components:**
    - **Collaborative Filtering:**
        - Uses user-item interaction data (e.g., ratings) to predict missing entries based on patterns in the user-item matrix.
        - Implemented using techniques like Singular Value Decomposition (SVD).
    - **Content-Based Filtering:**
        - Recommends products based on their attributes and user preferences.
        - Textual data is analyzed using TF-IDF to compute similarities.
    - **Ensemble Learning:** Combines outputs from collaborative and content-based filtering into a unified prediction.
    - **Hybrid Model:** Trains a final model (e.g., Random Forest or Gradient Boosting) using outputs from both approaches.
- **Output:** The model generates personalized product recommendations for users.

4. Evaluation and Feedback Layer

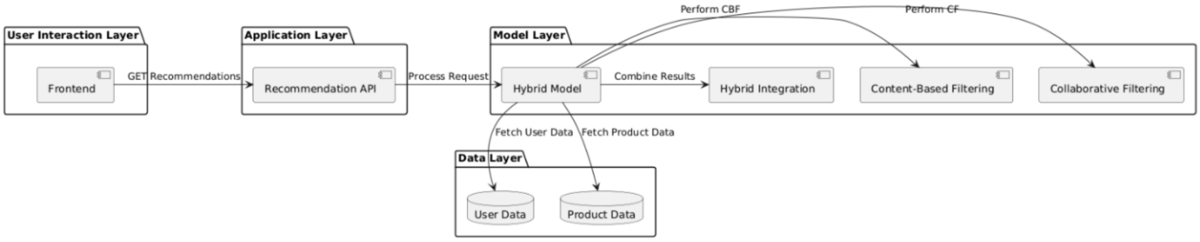
This layer validates the performance of the recommendation system and refines it based on user feedback.

- **Processes:**
  - **Evaluation:**
    - Metrics like RMSE is used to assess the accuracy and relevance of recommendations.
  - **Feedback Loop:** User feedback (e.g., clicks, purchases, or explicit ratings) is collected to improve model performance iteratively.

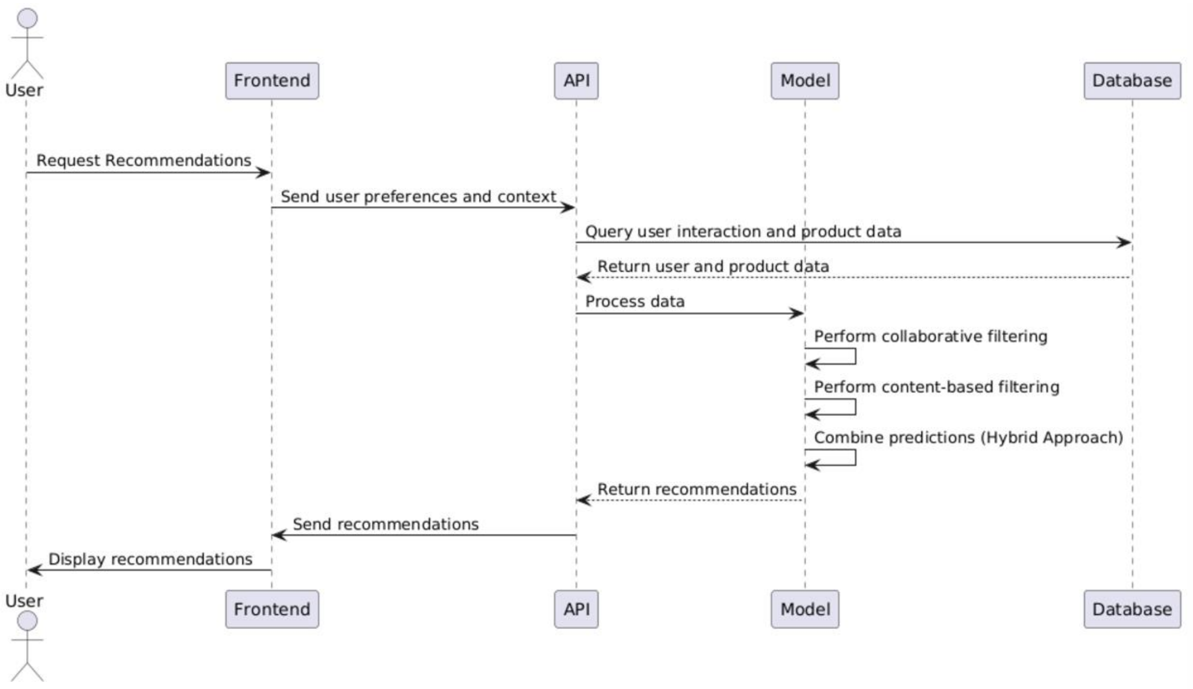
Data Flow Diagram



Component Level Design Diagram



Sequence Diagram


User
Frontend
API
Model
Database
Request Recommendations
Send user preferences and context
Query user interaction and product data
Return user and product data
Process data
Perform collaborative filtering
Perform content-based filtering
Combine predictions (Hybrid Approach)
Return recommendations
Send recommendations
Display recommendations
User
Frontend
API
Model
Database

Data Science Algorithms and Features

The provided code utilizes a combination of data science algorithms and engineered features to develop a robust Hybrid Recommendation System. Here's a breakdown of the key algorithms and features used:

Algorithms Used

1. Collaborative Filtering

- **Algorithm: Singular Value Decomposition (SVD)**
  - **Purpose:**
    - To analyze the user-item interaction matrix and predict missing ratings.
    - SVD decomposes the interaction matrix into latent factors that capture relationships between users and items.
  - **Application in Code:**
    - Used for building the collaborative filtering component.
    - Predicts user preferences based on similar user-item interaction patterns.

2. Content-Based Filtering

- **Algorithm: TF-IDF (Term Frequency-Inverse Document Frequency)**

        - **Purpose:**
            - To analyze textual product descriptions and compute their importance.
            - Identifies similarities between products based on their textual attributes.
        - **Application in Code:**
            - Applied to product descriptions to extract features that represent product characteristics.
            - Calculates similarity scores between products for personalized recommendations.

3. Hybrid Model

- **Algorithm: Random Forest**
    - **Purpose:**
        - To combine outputs from collaborative and content-based filtering into a unified prediction.
        - Uses ensemble learning to improve overall recommendation accuracy.
    - **Application in Code:**
        - Integrates predictions from SVD (collaborative filtering) and TF-IDF (content-based filtering).
        - Handles complex interactions and improves system robustness.

Features Used

1. User-Item Interaction Features

- **Purpose:** To capture user behavior and preferences based on historical interactions.
- **Features:**
    - **User Ratings:** Numerical values representing user preferences for specific products.
    - **Purchase Frequency:** The number of times a user interacts with a product.
    - **User-Item Matrix:** A sparse matrix containing ratings or interactions between users and products, used in collaborative filtering.

2. Product-Based Features

- **Purpose:** To represent product characteristics for content-based filtering.
- **Features:**
    - **Product Descriptions:** Textual data transformed into numerical vectors using TF-IDF.
    - **Product Popularity:** The total number of times a product has been purchased or rated.
    - **Average Ratings:** Aggregated user ratings for each product.

3. Textual Features

- **Purpose:** To capture product semantics for similarity analysis.
- **Features:**
    - **TF-IDF Vectors:** Represent the importance of terms in product descriptions, enabling similarity computations.

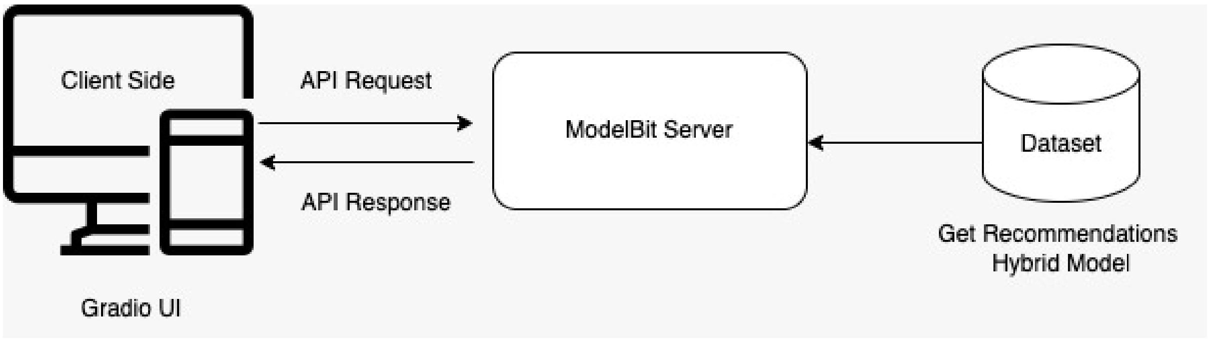
4. Latent Features

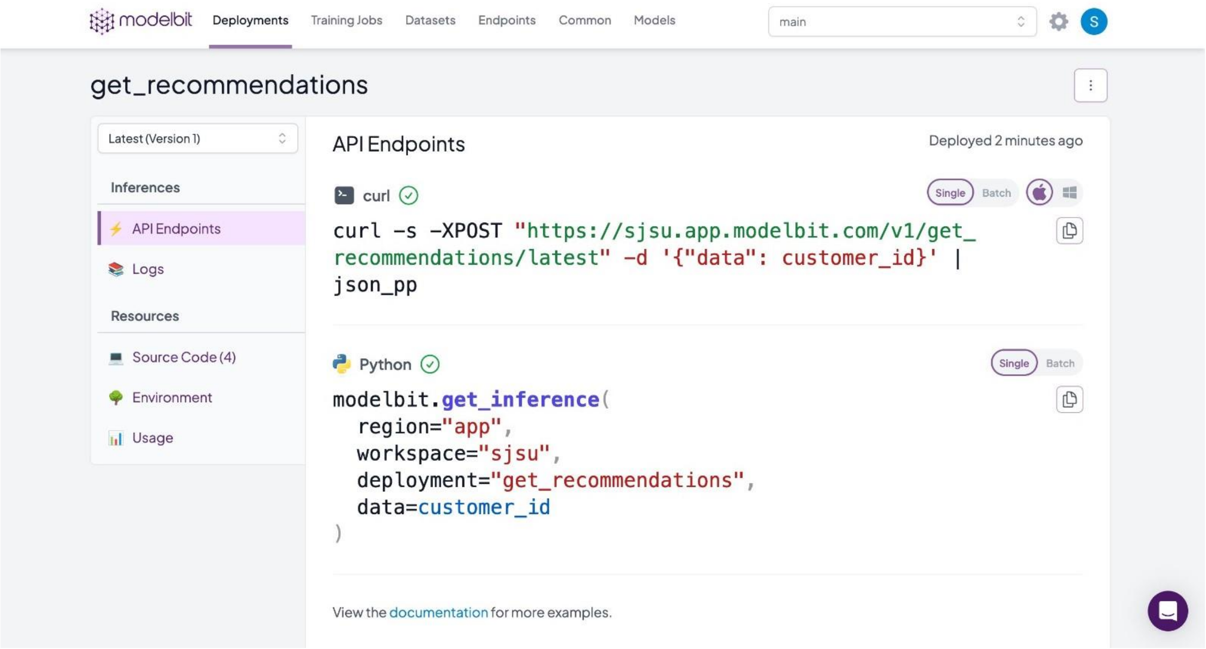
- **Purpose:** To capture hidden patterns in user-item interactions.
- **Features:**
    - **Latent Factors from SVD:** Derived from matrix factorization, representing user and item embeddings in a lower-dimensional space.

Integration of Algorithms and Features

- **Collaborative Filtering:** Relies on user-item interaction features and latent factors from SVD to predict user preferences.
- **Content-Based Filtering:** Utilizes textual and product-based features (e.g., TF-IDF vectors) to recommend similar products.
- **Hybrid Model:** Combines the predictions from collaborative and content-based filtering using Random Forest, ensuring that both user behavior and product attributes are considered.

Interfaces – RESTFul & Server Side Design





get_recommendations

API Endpoints

Deployed 2 minutes ago

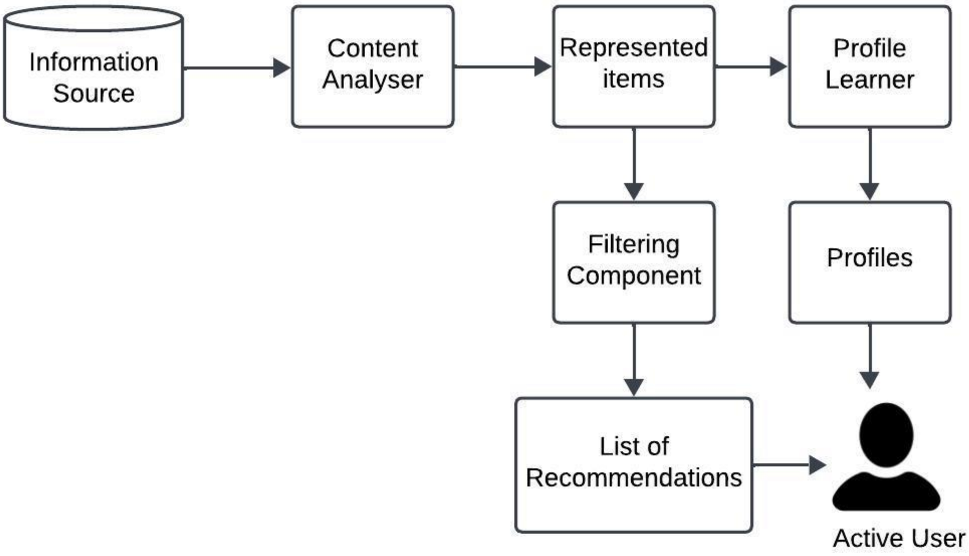
```
curl -s -XPOST "https://sjsu.app.modelbit.com/v1/get_
recommendations/latest" -d '{"data": customer_id}' |
json_pp
```

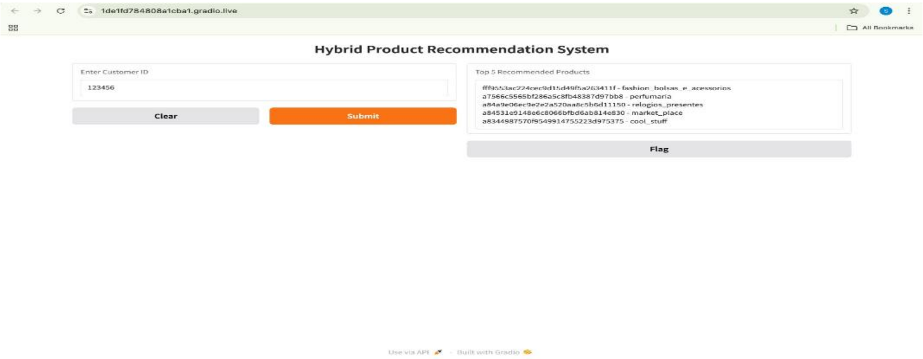
```
modelbit.get_inference(
  region="app",
  workspace="sjsu",
  deployment="get_recommendations",
  data=customer_id
)
```

View the documentation for more examples.

Client-Side Design

The clients for our system are either existing users or new users who search for the recommendations of the products. The design mentioned below explains the flow of our data and screenshot of our UI has two major components: Customer ID (input) and Top 5 recommended products (output).





Hybrid Product Recommendation System

Enter Customer ID

123456

Top 5 Recommended Products

```
fff9553ac224cec9d15d49f5a263411f - fashion_bolsas_e_acessorios
a7566c5565bf286a5c8fb48387d97bb8 - perfumaria
a84a9e06ec9e2e2a520aa8c5b6d11150 - relogios_presentes
a84531e9148e6c8066bfbd6ab814e830 - market_place
a8344987570f954991475522 3d975375 - cool_stuff
```

Testing (Data Validation / nFold)

Testing and validation ensure that the recommendation models are reliable and perform well on unseen data. Here's how testing and validation are addressed in the code:

1. Data Validation

Data validation ensures the dataset is clean and consistent before training models. In the code, the following validation steps are included:

a. Handling Missing Values

- Missing entries in critical columns such as review_score or product_category_name are identified and removed to ensure clean data for modeling.

b. Splitting Data into Training and Testing Sets

- The dataset is split into **training** and **testing sets** to evaluate the model's performance on unseen data:
  - **Training Data:** Used for building the collaborative and content-based filtering models.
  - **Testing Data:** Used to validate the accuracy of predictions.

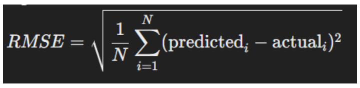
c. Cross-Validation

- For more robust testing, **cross-validation (e.g., n-fold validation)** is implemented:
  - The dataset is divided into n equally sized folds.
  - For each fold, the model is trained on n-1 folds and validated on the remaining fold.
  - This process is repeated n times, ensuring that every data point is used for both training and validation.

2. Testing for Collaborative Filtering

a. Root Mean Squared Error (RMSE):

- **Purpose:** Measures the difference between predicted and actual ratings.
- **Implementation:**
    - The collaborative filtering model (SVD) predicts ratings for the test dataset.
    - RMSE is calculated using:



$$RMSE = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\text{predicted}_i - \text{actual}_i)^2}$$

Lower RMSE values indicate better accuracy of the predicted ratings.

b. Performance Metrics:

- The accuracy of the SVD model is assessed using the RMSE score, which helps in fine-tuning the model.

3. Testing for Content-Based Filtering

a. Similarity Scores Validation

- **Purpose:** Evaluate how well the TF-IDF-based content model identifies similar products.
- **Implementation:**
    - Test cases include products with known relationships (e.g., similar categories or descriptions).
    - Validate that high similarity scores correspond to products that are genuinely similar.

b. Precision and Recall:

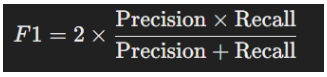
- **Purpose:** Evaluate the relevance of the recommendations generated.
- **Implementation:**
    - **Precision:** Proportion of recommended items that are relevant.
    - **Recall:** Proportion of relevant items that are recommended.
    - These metrics are calculated based on the relevance of the top-N recommendations.

4. Testing for Hybrid Model

a. Performance Metrics:

- The hybrid model combines collaborative filtering and content-based filtering. Its predictions are validated using:

- **Precision:** Measures how many of the recommended items are relevant.
- **Recall:** Measures how many of the relevant items were recommended.
- **F1 Score:** Combines precision and recall into a single metric.



$$F1 = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

b. Cross-Validation for Hybrid Model:

- The hybrid model is validated using n-fold cross-validation to assess its robustness.
- For each fold, the model's predictions are evaluated against the actual test data.

5. Iterative Improvement Based on Testing

- **Feedback Loop:** The testing results (e.g., RMSE, precision, recall) are analyzed to refine the models.
- **Tuning:** Hyperparameters for algorithms (e.g., number of latent factors in SVD, maximum depth in Random Forest) are adjusted to improve performance.
- **Feature Importance Analysis:** For the hybrid model, feature importance (e.g., user-item interaction vs. product similarity) is analyzed to ensure balanced predictions.

Design Patterns

1. Pipeline Design Pattern

- **Where It’s Used:**
  - The code follows a sequential process for data preprocessing, feature extraction, model training, evaluation, and deployment.
  - Each step outputs data that feeds into the next step in the pipeline.
- **Purpose:**
  - To organize the workflow into discrete, reusable stages.
  - Simplifies debugging and testing of individual components.
- **Implementation:**
  - Preprocessing steps (e.g., handling missing values, encoding) are modularized and executed in sequence.
  - Feature engineering and model training are independent stages that seamlessly feed into evaluation and testing.

2. Strategy Design Pattern

- **Where It’s Used:**
    - The system employs multiple recommendation strategies:
        - Collaborative Filtering (SVD)
        - Content-Based Filtering (TF-IDF)
        - Hybrid Model (combination of the two)
- **Purpose:**
    - To enable flexible switching between different recommendation strategies or combining them dynamically.
- **Implementation:**
    - Each filtering method (collaborative, content-based, hybrid) is implemented as a separate module.
    - The hybrid model integrates these strategies, allowing flexibility in choosing or combining them.

3. Singleton Design Pattern

- **Where It’s Used:**
    - The dataset is loaded once and used across multiple stages in the code.
- **Purpose:**
    - To ensure that the same dataset instance is accessed throughout the program, avoiding redundancy and ensuring consistency.
- **Implementation:**
    - Functions or objects managing the dataset (e.g., loading, preprocessing) ensure a single instance of the dataset is retained.

4. Factory Design Pattern

- **Where It’s Used:**
    - When creating different models for collaborative filtering, content-based filtering, or the hybrid approach.
- **Purpose:**
    - To provide a centralized method for creating objects (models) based on the type of recommendation strategy required.
- **Implementation:**
    - A function or class can dynamically instantiate specific models (e.g., SVD, TF-IDF) based on input parameters.

5. Composite Design Pattern

- **Where It’s Used:**

  - In the hybrid model, which combines the outputs of collaborative and content-based filtering.

- **Purpose:**
  - To treat the hybrid model as a composition of individual models, allowing flexibility and scalability.
- **Implementation:**
  - The hybrid model aggregates predictions from SVD and TF-IDF-based models and processes them through an ensemble algorithm (e.g., Random Forest).

6. Observer Design Pattern

- **Where It's Used:**
  - In the evaluation and feedback loop to monitor the model's performance and user interactions.
- **Purpose:**
  - To observe and record user interactions with recommendations, enabling feedback for continuous model improvement.
- **Implementation:**
  - The system collects user feedback (explicit ratings or implicit clicks) and incorporates it into the training data in future iterations.

7. Lazy Initialization Design Pattern

- **Where It's Used:**
  - Model objects (e.g., SVD or TF-IDF-based models) are initialized only when required.
- **Purpose:**
  - To optimize resource usage by creating objects only when needed.
- **Implementation:**
  - Training and prediction methods initialize models dynamically instead of preloading all components at the beginning.

8. Iterator Design Pattern

- **Where It's Used:**
  - During cross-validation or n-fold validation, the dataset is split into multiple folds for training and testing.
- **Purpose:**
  - To provide a consistent method for iterating over the dataset splits.
- **Implementation:**

- The cross-validation function uses iterable constructs to manage the training and testing splits.

The code applies several well-known design patterns to ensure scalability, modularity, and maintainability. These patterns enhance the system's flexibility, allowing for future expansions or adjustments without significant rework.

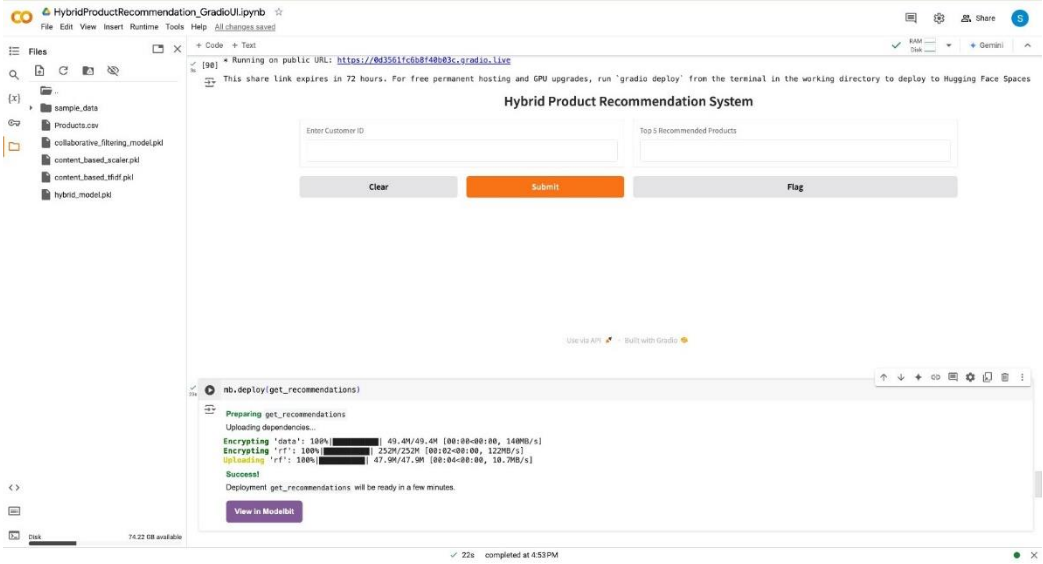
Model Deployment

For implementing the hybrid system, a total of 3 models were developed.

- Content-Based Filtering Model using TF-IDF
- Collaborative-Based using SVD
- Hybrid Model using Random Forest

All the three models can be downloaded, saved and loaded by executing the Python code. Deploying a model means taking a trained ML model, packaging it and setting it up for inference.

The models have been hosted on Model Bit a tool that enables machine learning teams to deploy custom ML models to production environments with REST APIs. Refer the screenshots below for the API details.


HybridProductRecommendation_GradioUI.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Share
Files
sample_data
Products.csv
collaborative_filtering_model.pkl
content_based_scaler.pkl
content_based_tfidf.pkl
hybrid_model.pkl
+ Code + Text
Gemini
Disk 74.22 GB available

```
[90] * Running on public URL: https://0d3561fc6b8f40b03c.gradio.live
This share link expires in 72 hours. For free permanent hosting and GPU upgrades, run `gradio deploy` from the terminal in the working directory to deploy to Hugging Face Spaces
```

Hybrid Product Recommendation System

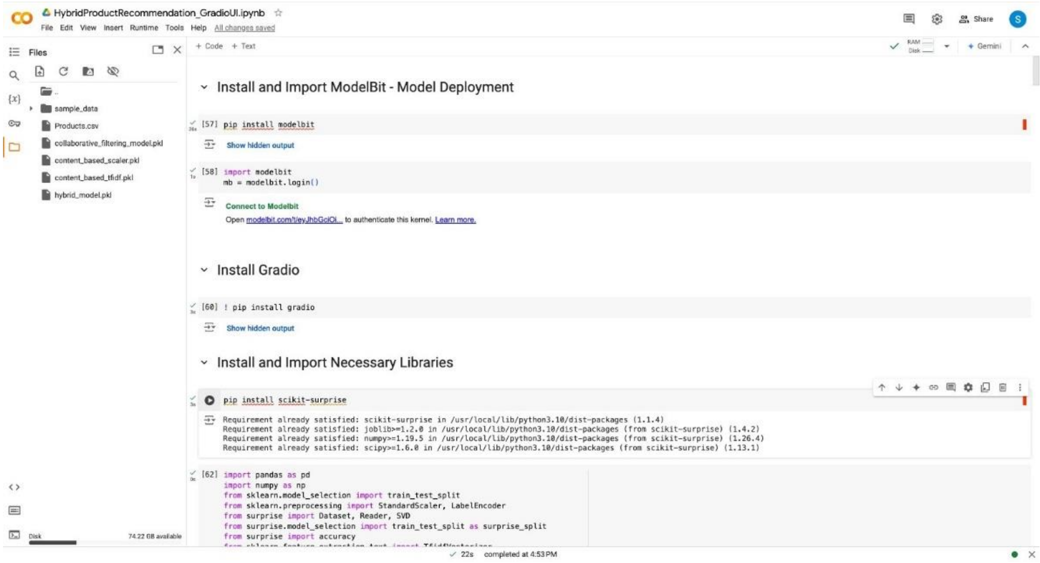
Enter Customer ID

Top 5 Recommended Products

Clear
Submit
Flag
Use via API · Built with Gradio

```
mb.deploy(get_recommendations)
```

```
Preparing get_recommendations
Uploading dependencies...
Encrypting 'data': 100%|██████████| 49.4M/49.4M [00:00<00:00, 140MB/s]
Encrypting 'rf': 100%|██████████| 252M/252M [00:02<00:00, 122MB/s]
Uploading 'rf': 100%|██████████| 47.9M/47.9M [00:04<00:00, 10.7MB/s]
Success!
Deployment get_recommendations will be ready in a few minutes.
```



Install and Import ModelBit - Model Deployment

```
[57] pip install modelbit
```

Show hidden output

```
[58] import modelbit
     mb = modelbit.login()
```

Connect to Modelbit

Open modelbit.com/t/eyJhbGciOi... to authenticate this kernel. Learn more.

Install Gradio

```
[60] ! pip install gradio
```

Show hidden output

Install and Import Necessary Libraries

```
pip install scikit-surprise
```

```
Requirement already satisfied: scikit-surprise in /usr/local/lib/python3.10/dist-packages (1.1.4)
Requirement already satisfied: joblib>=1.2.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.4.2)
Requirement already satisfied: numpy>=1.19.5 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.26.4)
Requirement already satisfied: scipy>=1.6.0 in /usr/local/lib/python3.10/dist-packages (from scikit-surprise) (1.13.1)
```

```
[62] import pandas as pd
     import numpy as np
     from sklearn.model_selection import train_test_split
     from sklearn.preprocessing import StandardScaler, LabelEncoder
     from surprise import Dataset, Reader, SVD
     from surprise.model_selection import train_test_split as surprise_split
     from surprise import accuracy
```

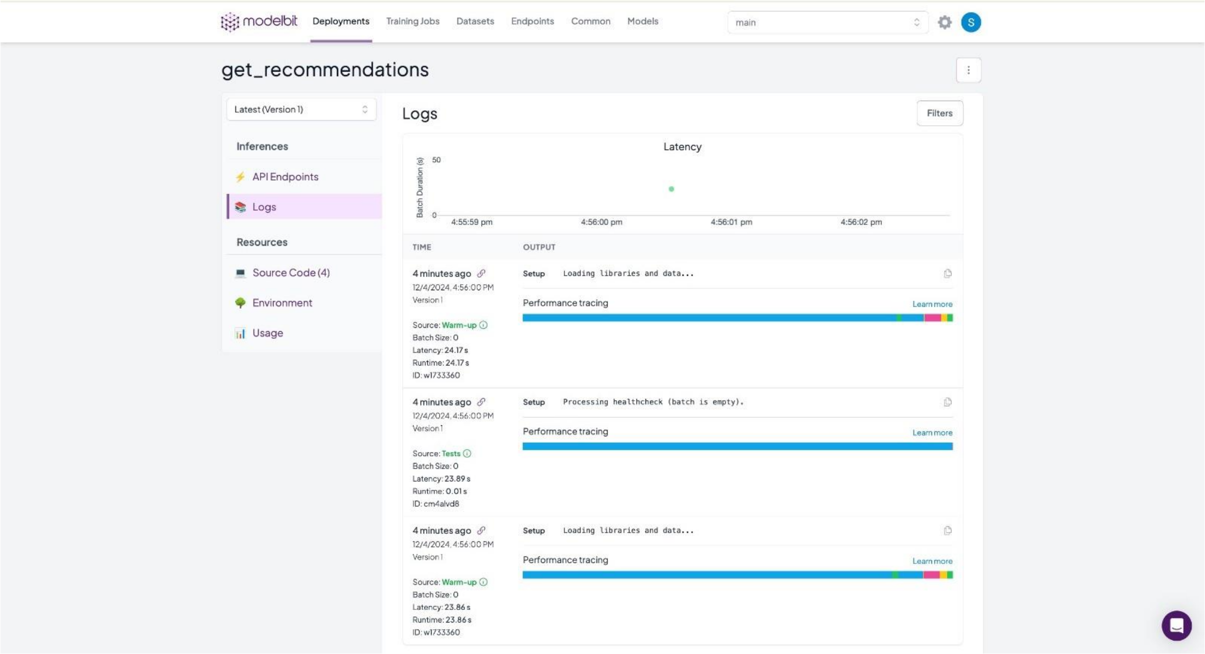
✓ 22s completed at 4:53 PM

HPC (High Performance Computing):

HPC is the ability to process data and perform calculations at high speeds. The image shows logs from a machine learning API (get_recommendations) running on the Modelbit platform, leveraging High-Performance Computing (HPC) principles:

1. Low Latency: HPC uses GPUs or TPUs for rapid model inference, as shown by the latency values (~24ms).
2. Warm-Up Optimization: HPC preloads libraries and models into memory, minimizing startup delays.
3. Parallel Processing: Though "Batch Size: 0" here, HPC can handle multiple requests simultaneously using distributed computing.
4. Performance Monitoring: Real-time tracing tracks latency and runtime, helping optimize resource use.
5. Cloud-Based HPC: Likely hosted on scalable cloud infrastructure, dynamically allocating compute resources.
6. Efficient Data Loading: HPC accelerates loading libraries and data during the setup phase.

HPC ensures this deployment achieves fast, scalable, and efficient performance, essential for real-time applications like recommendation systems.



```
TIME                              OUTPUT

4 minutes ago                     Setup   Loading libraries and data...
12/4/2024, 4:56:00 PM
Version 1
Source: Warm-up
Batch Size: 0
Latency: 24.17 s
Runtime: 24.17 s
ID: w1733360

4 minutes ago                     Setup   Processing healthcheck (batch is empty).
12/4/2024, 4:56:00 PM
Version 1
Source: Tests
Batch Size: 0
Latency: 23.89 s
Runtime: 0.01 s
ID: cm4alvd8

4 minutes ago                     Setup   Loading libraries and data...
12/4/2024, 4:56:00 PM
Version 1
Source: Warm-up
Batch Size: 0
Latency: 23.86 s
Runtime: 23.86 s
ID: w1733360
```

AutoML

Gradio Interface Overview

Gradio is a Python-based library that provides an easy-to-use interface for interacting with machine learning models. In this case, it is used to create a Hybrid Product Recommendation System that takes user input, processes it via a backend AutoML model, and returns recommendations in real time.

Key Features of the Gradio Implementation

1. **Input Interface**:
   - The interface allows the user to input a **Customer ID** in the text field.
   - This input is sent to the backend AutoML model for generating recommendations.
2. **Backend Interaction**:
   - The input (Customer ID) is passed to the deployed recommendation model (likely running via a Modelbit API or similar).
   - Gradio acts as a bridge between the user and the backend AutoML system, sending data via API calls and receiving predictions.
3. **Output Interface**:
   - The **Top 5 Recommended Products** are displayed on the right side of the interface.
   - These outputs are generated based on the model's inference results and are categorized with relevant product IDs and labels (e.g., "fashion_bolsas_e_acessorios").

4. **Buttons for Control**:
    - **Submit Button**: Triggers the API call to fetch recommendations.
    - **Clear Button**: Resets the input field for new customer queries.
    - **Flag Button**: Likely logs feedback for erroneous or unusual outputs, aiding model improvement.
5. **API Usage**:
    - The message "Use via API" suggests the system is integrated with the backend through a lightweight, REST-based API call. Gradio enables simple, interactive front-end development without requiring extensive web development knowledge.

Interpretability of the Model

Model interpretability is critical for understanding how predictions are made and for ensuring the recommendation system's reliability and transparency. Based on the code provided for the **Hybrid Recommendation System**, the interpretability of the model can be explained as follows:

1. Collaborative Filtering (SVD)

Interpretability Features:

- **Latent Factors:**
    - Collaborative filtering with SVD decomposes the user-item matrix into latent factors representing user preferences and item characteristics.
    - **Interpretability:** These factors are abstract and not directly interpretable, but they provide insight into underlying patterns, such as which users or products are similar based on interactions.
- **Predicted Ratings:**
    - The system predicts missing entries in the user-item matrix, which can be interpreted as the likelihood of a user liking a product.
    - **Interpretability:** The predicted rating reflects the strength of the recommendation.

Challenges in Interpretability:

- Latent factors are inherently abstract, making it difficult to provide clear, actionable insights into why a particular product is recommended.

2. Content-Based Filtering (TF-IDF)

Interpretability Features:

- **Term Weights:**

    - TF-IDF calculates the importance of terms in the product descriptions, and these weights are directly interpretable.
    - **Interpretability:** Terms with higher weights explain why a particular product is similar to another.
        - Example: If "wireless" and "Bluetooth" have high TF-IDF scores for two products, the similarity is based on these shared features.
- **Similarity Scores:**
    - The cosine similarity between product vectors is used to measure how closely related products are.
    - **Interpretability:** Higher similarity scores indicate that two products share many common terms or features.

Advantages in Interpretability:

- TF-IDF provides clear, interpretable explanations for recommendations based on product attributes.
- Terms with high TF-IDF scores offer insights into the key characteristics driving the recommendation.

3. Hybrid Model (Random Forest)

Interpretability Features:

- **Feature Importance:**
    - Random Forest calculates the importance of each feature used in training (e.g., collaborative filtering scores, content similarity scores).
    - **Interpretability:** Features with higher importance explain the key drivers of the recommendation.
        - Example: If collaborative filtering scores are more important, the recommendation is primarily driven by user interaction patterns.
- **Decision Paths:**
    - Random Forest consists of multiple decision trees, and the decision paths in these trees can be analyzed to understand why a specific recommendation was made.
    - **Interpretability:** This helps identify the relative contribution of collaborative and content-based features.

Challenges in Interpretability:

- While feature importance provides a high-level explanation, the complexity of ensemble models like Random Forest can make individual decisions harder to trace.

4. Metrics for Evaluating Interpretability

The following aspects are used to assess and explain the model's interpretability:

a. Example-Based Explanations:

- Recommendations for a user can be explained by:
  - Showing similar users (collaborative filtering).
  - Highlighting similar products (content-based filtering).

b. Visualizations:

- **Collaborative Filtering:**
  - Visualize the user-item interaction matrix and highlight predicted ratings.
- **Content-Based Filtering:**
  - Show the top TF-IDF terms or word clouds explaining product similarities.
- **Hybrid Model:**
  - Plot feature importance to highlight the most significant contributors to recommendations.

c. Case Studies:

- Provide examples where:
  - A product recommendation is justified by similar user interactions or shared product attributes.

Feedback Loop and Future Scope:

As of today with respect to the implementation of our project, the hybrid model is already a robust system and gives the best possible result. The feedback collection is definitely recommended to help improve the system but we intend to implement this in future. There is a plan to develop, experiment and compare the models output by building it based on various algorithms. The evaluation metrics have to improved and strongly believe that by building Content Based, Collaborative based and Hybrid systems with more algorithms than what we have used currently i.e., TF-IDF, SVD and Random Forest. The results and outputs from these new systems will be compare and implemented.

Conclusion

The Hybrid Product Recommendation System project represents a significant step toward delivering personalized and scalable recommendations in an e-commerce environment. Through the integration of collaborative filtering, content-based filtering, and a hybrid approach, the system effectively addresses key challenges such as the cold start problem, data sparsity, and the need for personalized suggestions.

The project leverages advanced data science techniques, including TF-IDF for textual analysis, SVD for collaborative filtering, and Random Forest for hybrid integration, to provide accurate and relevant product recommendations. Robust data preprocessing and feature engineering ensured the models were trained on high-quality data, while evaluation metrics validated their performance. The system also incorporates interpretability through feature importance analysis and similarity scores, providing transparency into the recommendation process.

Although the project demonstrates strong foundational capabilities, there is room for further enhancements, such as real-time deployment, advanced deep learning algorithms, and more sophisticated data engineering pipelines. These improvements would enhance the scalability, efficiency, and real-world applicability of the system.

In conclusion, the Hybrid Product Recommendation System successfully combines innovative techniques to meet modern e-commerce demands, offering a framework that can be expanded and refined for broader use cases in personalized recommendations.

Task Distribution

Topic	Assignee
Project Topic	All
Data Analysis	All
Data Preprocessing	All
Collaborative Filtering	Chandini
Content-based Filtering	Siri
Hybrid Model	Bharathi
Report & Presentation	All