# Hybrid Product Recommendation System

*Presented By: TriLogic*
*Chandini Saisri Uppuganti*
*Soumya Bharathi Vetukuri*
*Siri Batchu*

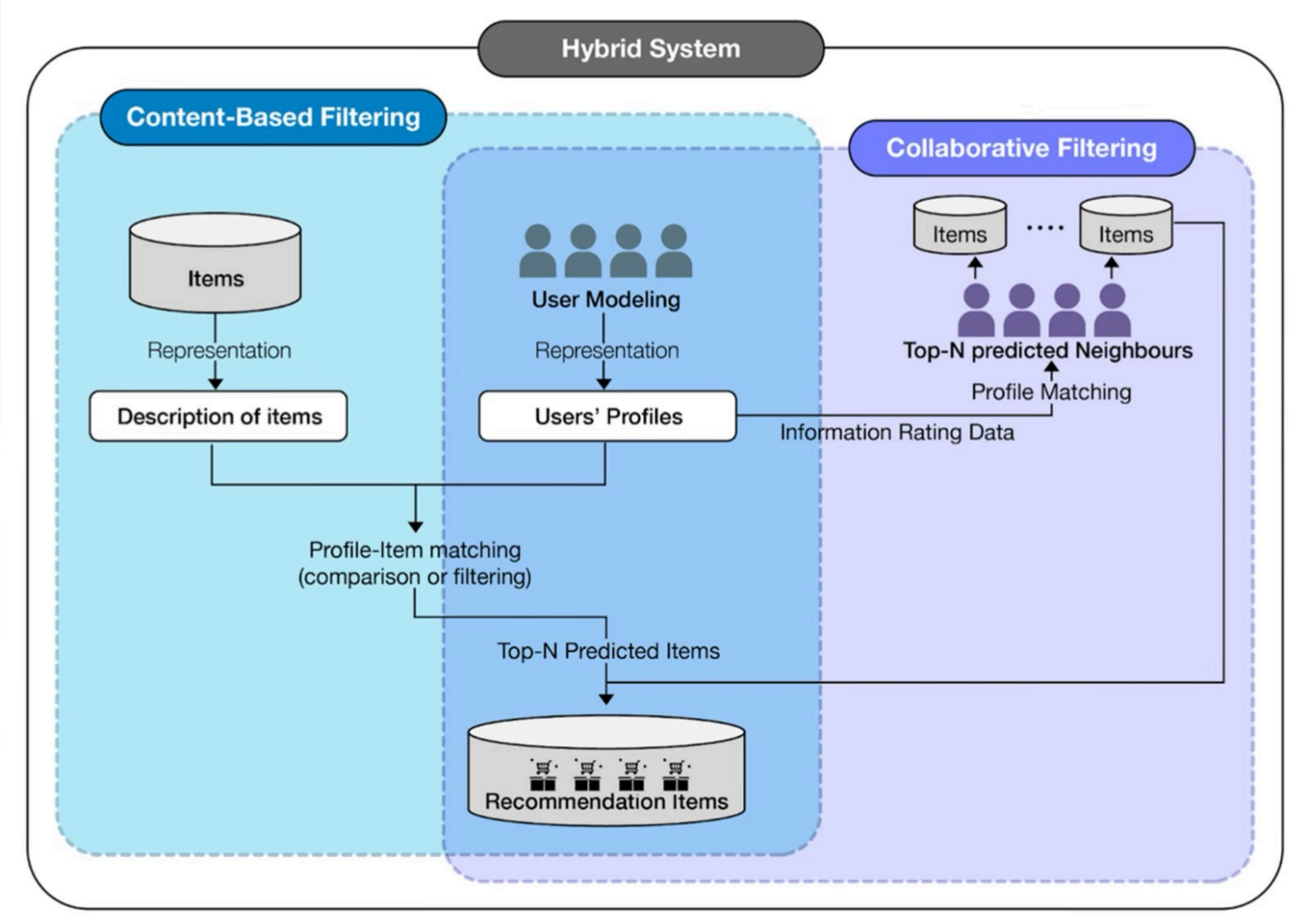
# Introduction

## 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

# Recommendation System



# KDD (Knowledge Discovery in Databases)

KDD refers to the process of identifying valuable, actionable patterns, trends, and insights from large datasets.

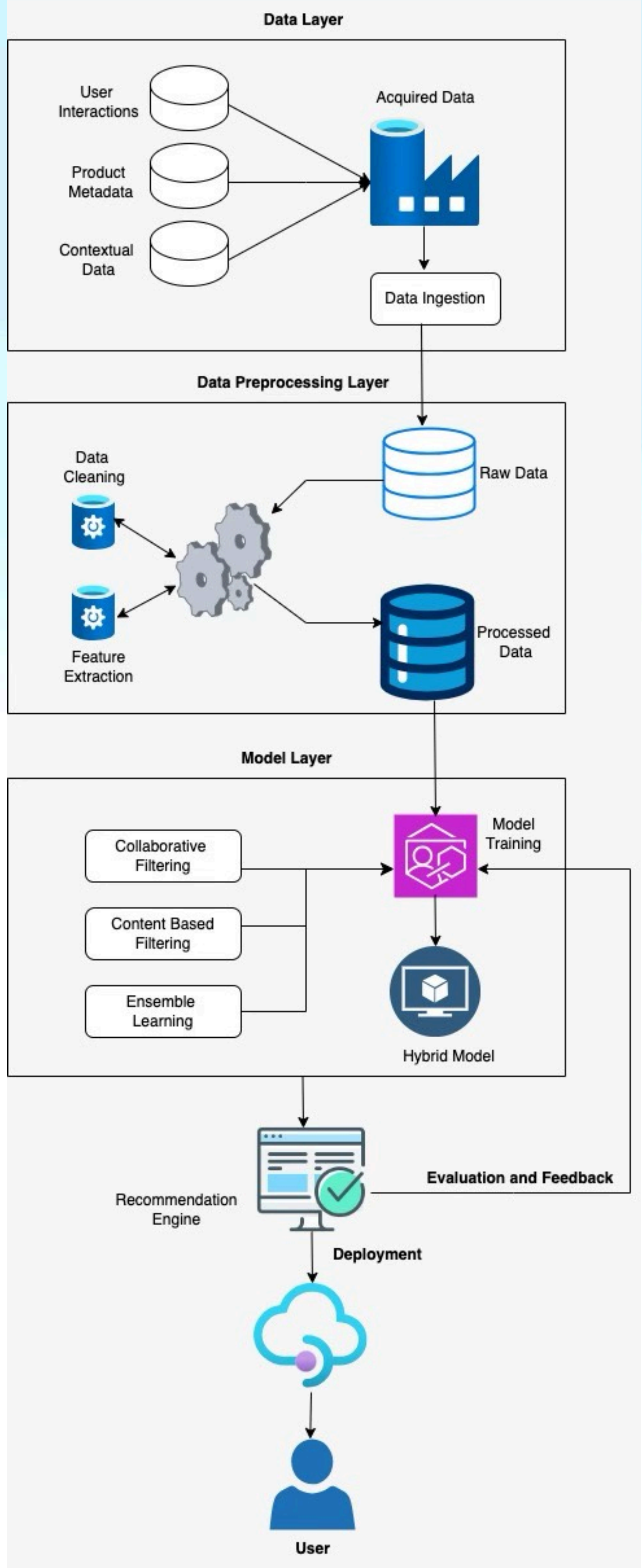
1. **Data Selection:** Collecting transaction data from customers, including product purchases, timestamps and customer demographics.
2. **Data Preprocessing:** Cleaning the data to remove missing or duplicate entries.
3. **Data Transformation:** Aggregating data by customer or product to derive useful features like total spending or average purchase frequency.
4. **Data Mining:** Using clustering to group similar customers based on purchasing behavior, or applying association rule mining to discover which products are often purchased together.
5. **Pattern Evaluation:** Evaluating the discovered patterns to determine the most meaningful customer segments or product pairings.
6. **Knowledge Representation:** Presenting the findings through charts or dashboards to inform marketing strategies.

# Business Requirements

Key business requirements include:

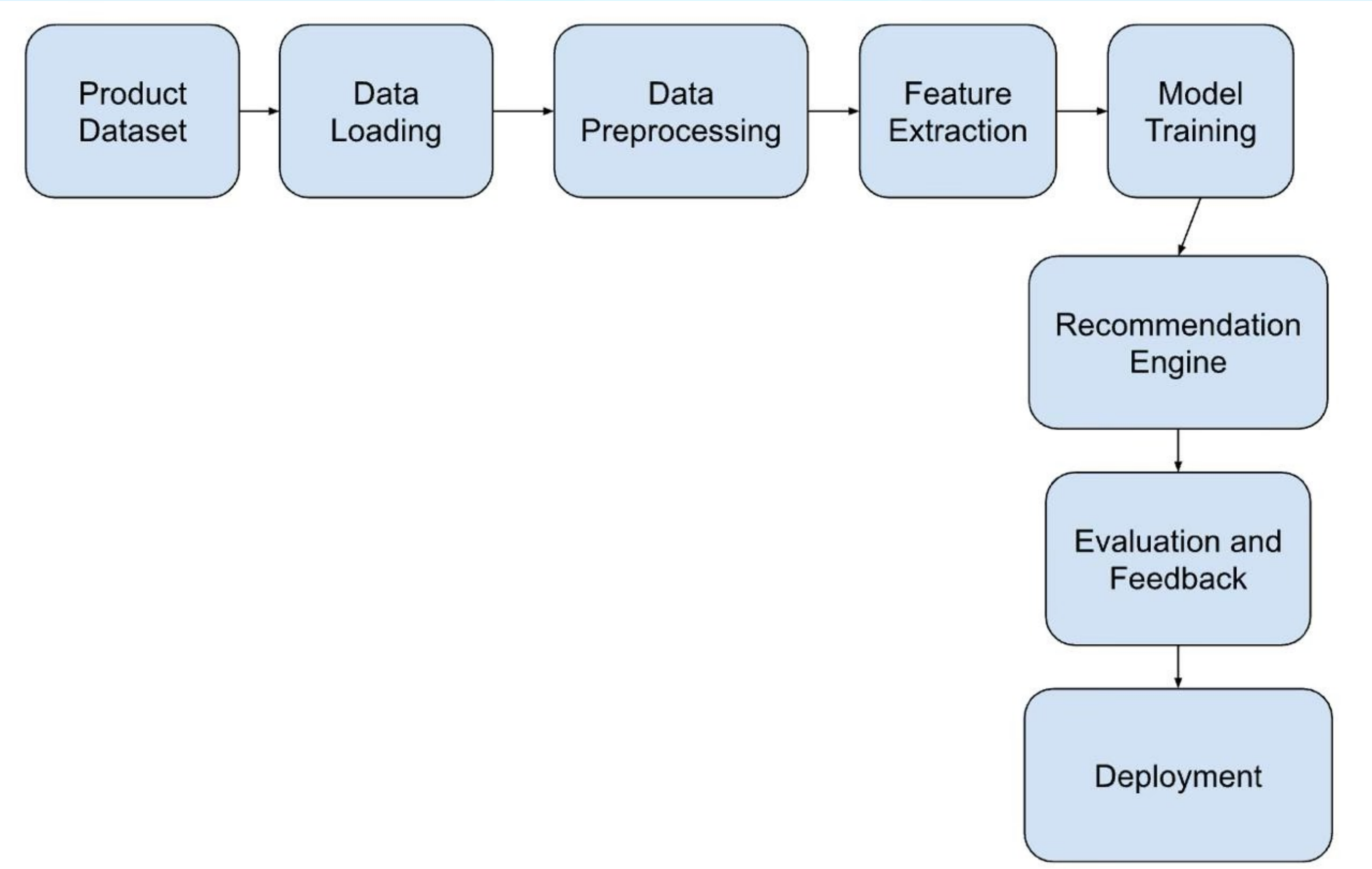
1. **Comprehensive User Understanding**: Combine collaborative and content-based filtering to gain a holistic view of user preferences, enabling more accurate recommendations.
2. **Model Building:** Develop a hybrid model to deliver personalized product suggestions that resonate with individual user preferences.
3. **Cold Start Problem Mitigation**: Address challenges related to new users or items by integrating multiple recommendation approaches, ensuring relevant suggestions despite limited initial data.
4. **Scalability and Performance**: Ensure the system focuses on minimizing errors during training to improve the accuracy of predictions.

# High Level Architecture




•

# Dataflow Diagram



# Dataset Information

1. This dataset provides a comprehensive view of the e-commerce operations, encompassing order details, customer demographics, product information, and customer feedback.
2. The dataset is sourced from Olist, an e-commerce platform and includes orders made between 2016 and 2018 across various marketplaces in Brazil.
3. The dataset contains 100k orders information with 25 features (order_id, product_id, product_category_name, review_score, customer_id, reviews).
4. The features in this dataset are absolutely perfect to study customer's interests and develop predictions based on their past behavior.

# 1. Data Preprocessing

Prepare and clean data for building hybrid recommendation.

**Steps involved:**

1. Import necessary libraries
2. Data loading
3. Feature Engineering
4. Data Cleaning
5. Saved the cleaned dataset

## Handling Missing Values

```
print("\nData types and missing values:")
print(data.info())
```

Show hidden output

```
print("\nSummary of missing values:")
print(data.isnull().sum())
```

Show hidden output

```
# Fill missing dates with placeholders or logical defaults
data['order_aproved_at'].fillna(data['order_purchase_timestamp'], inplace=True)
data['order_delivered_customer_date'].fillna('Not Delivered', inplace=True)
```

Show hidden output

```
print("\nSummary of missing values:")
print(data.isnull().sum())
```

Show hidden output

```
# Fill missing review fields with placeholders
data['review_comment_title'].fillna('No Title', inplace=True)
data['review_comment_message'].fillna('No Comment', inplace=True)
```

Show hidden output

```
print("\nSummary of missing values:")
print(data.isnull().sum())
```

Show hidden output

## Feature Engineering

```
# Convert timestamps to datetime
data['order_purchase_timestamp'] = pd.to_datetime(data['order_purchase_timestamp'], errors='coerce')
data['order_aproved_at'] = pd.to_datetime(data['order_aproved_at'], errors='coerce')
data['order_delivered_customer_date'] = pd.to_datetime(data['order_delivered_customer_date'], errors='coerce')
```

Show hidden output

```
# Feature Engineering
data['total_order_value'] = data['order_products_value'] + data['order_freight_value']
data['approval_time'] = (data['order_aproved_at'] - data['order_purchase_timestamp']).dt.seconds / 3600
data['delivery_time'] = (data['order_delivered_customer_date'] - data['order_purchase_timestamp']).dt.days
```

```
# Fill missing values in new columns
data['approval_time'].fillna(data['approval_time'].median(), inplace=True)
data['delivery_time'].fillna(data['delivery_time'].median(), inplace=True)
```

Show hidden output

## Normalizing Numerical Data

```
import seaborn as sns
import matplotlib.pyplot as plt
```

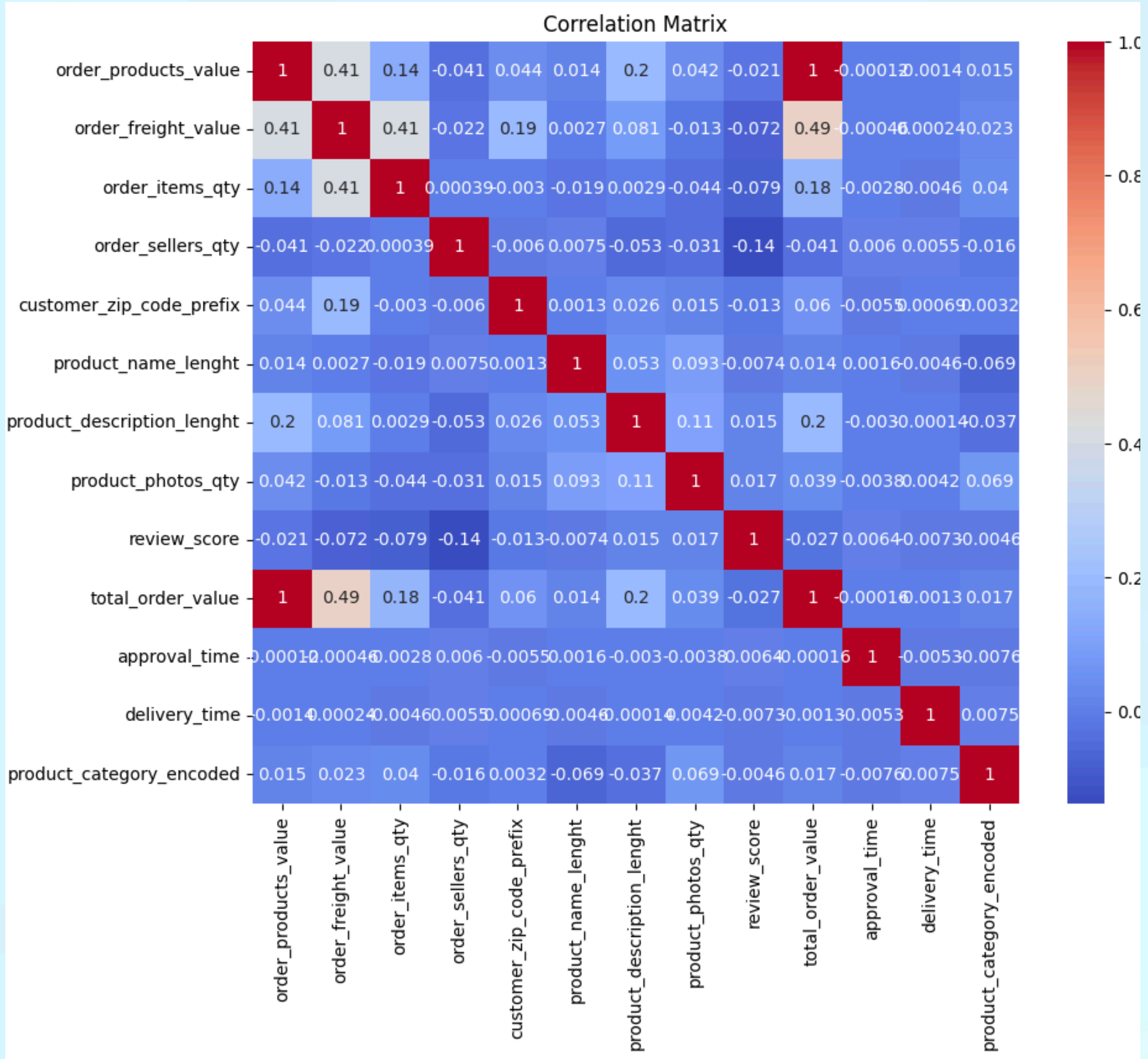
```
# Normalize numeric features
scaler = StandardScaler()
data[['order_products_value', 'order_freight_value', 'total_order_value']] = scaler.fit_transform(
    data[['order_products_value', 'order_freight_value', 'total_order_value']]
)
```

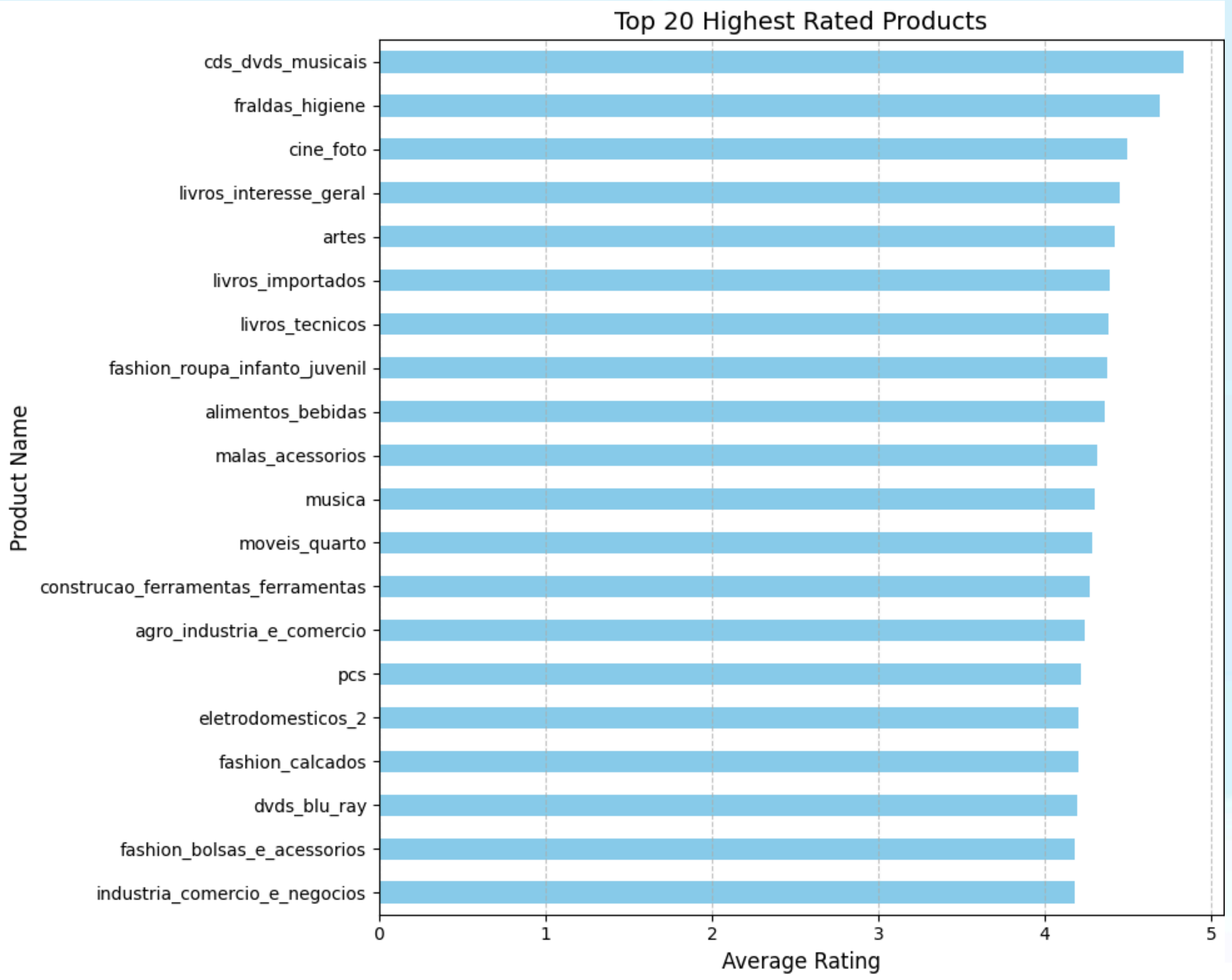
## Normalizing Categorical Columns

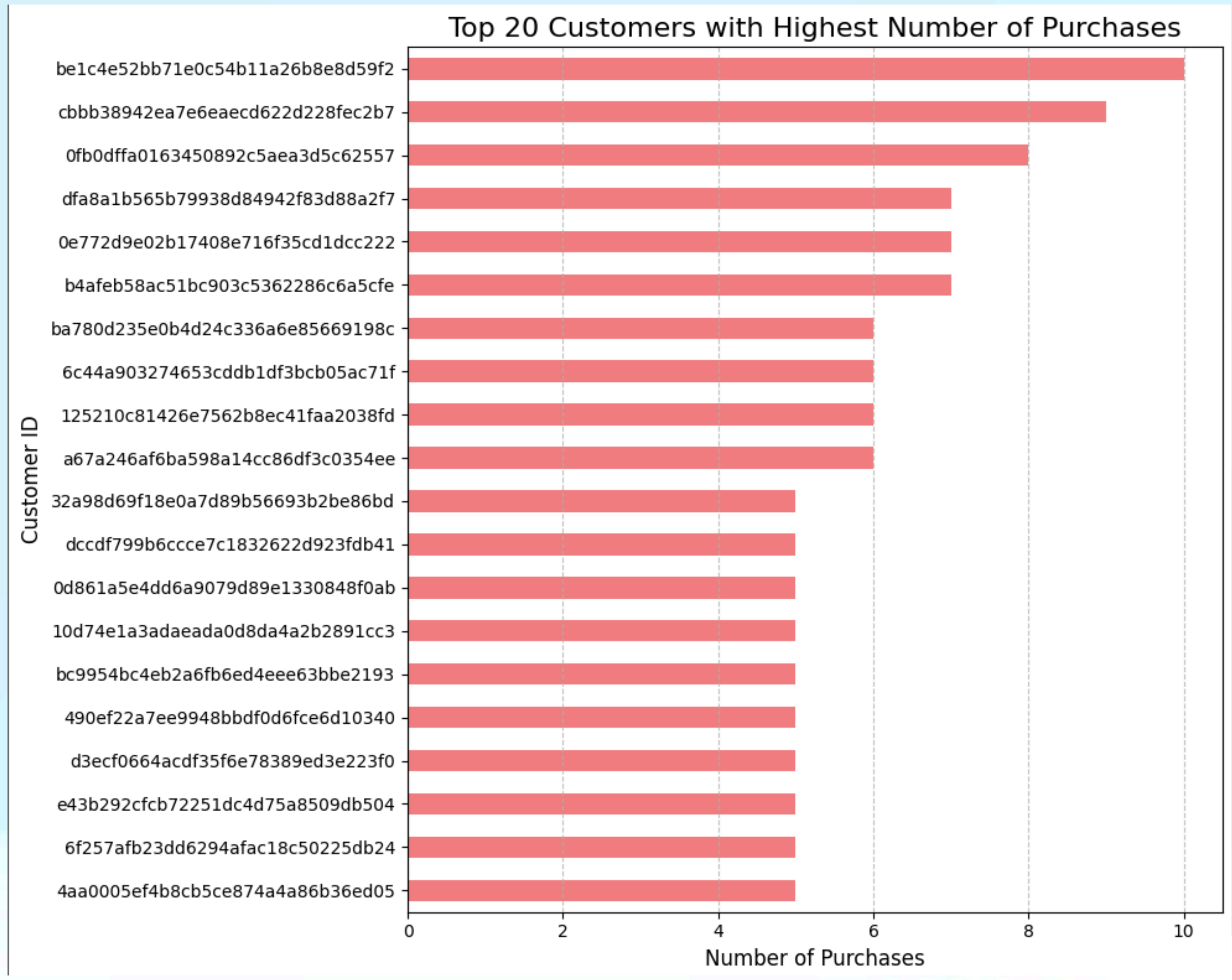
```
# Encode categorical columns
encoder = LabelEncoder()
data['product_category_encoded'] = encoder.fit_transform(data['product_category_name'])
```

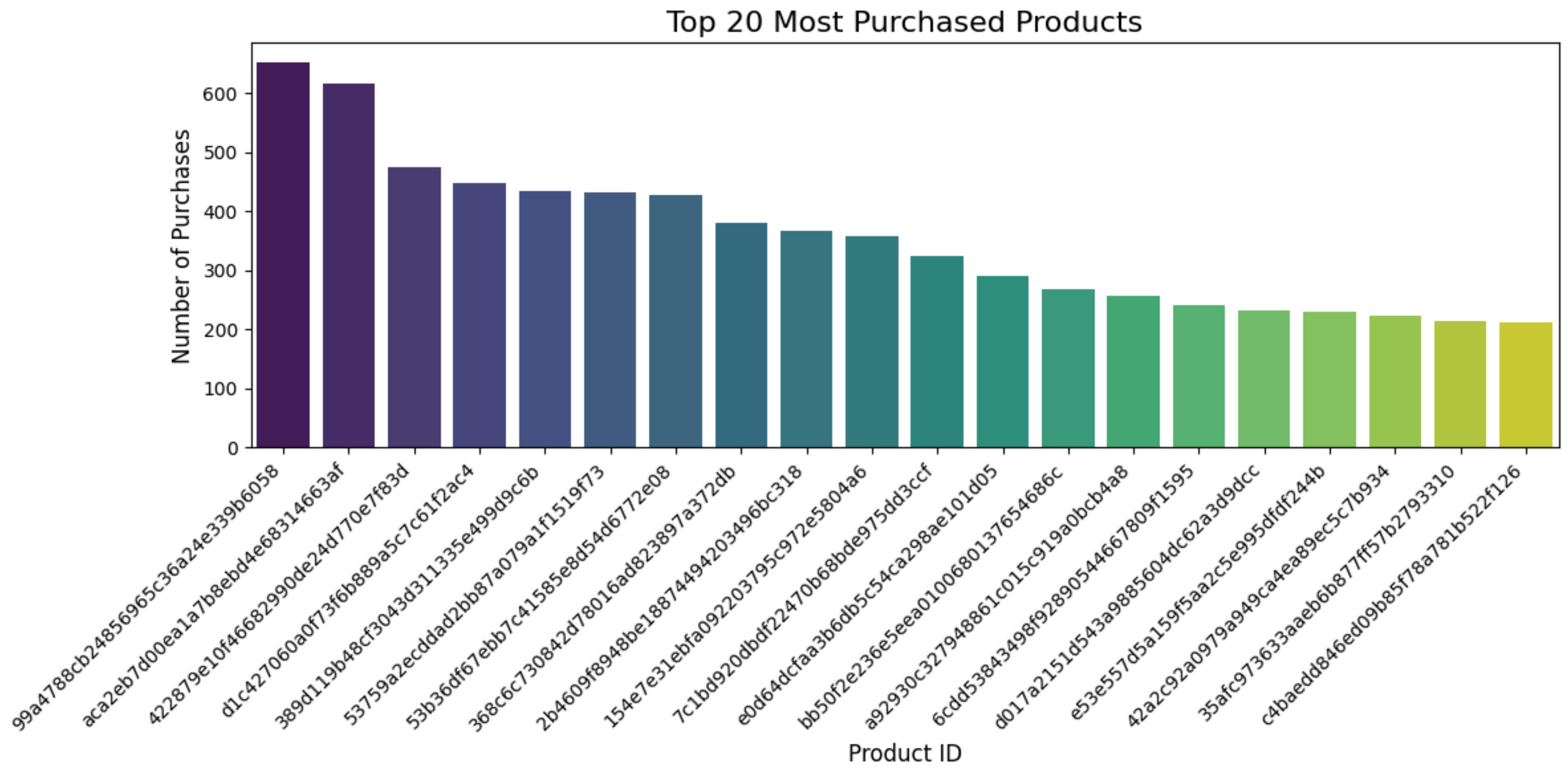
# 2. Exploratory Data Analysis

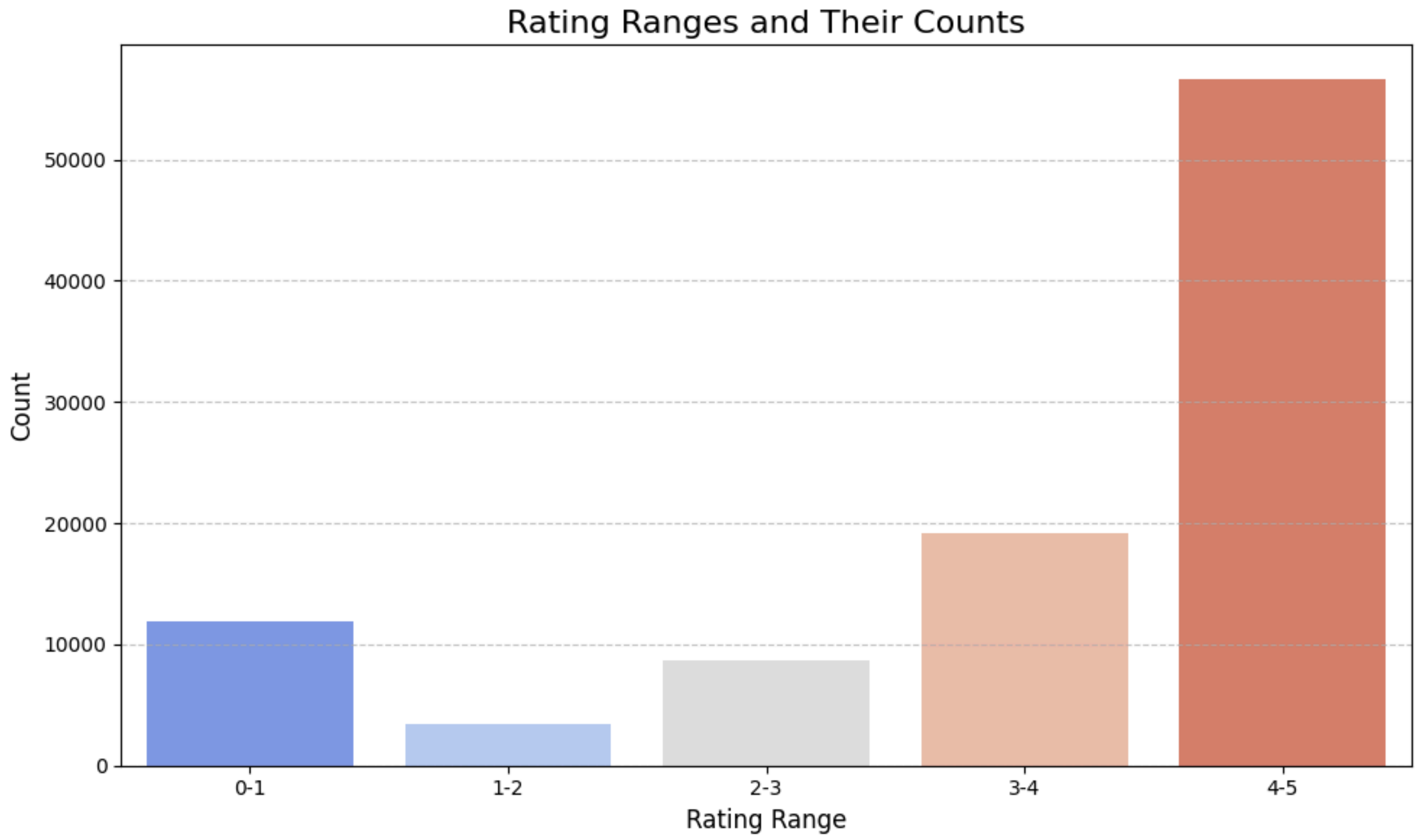
1. Calculated and visualized Correlation matrix
2. Analyzed high-rated products
3. Explored customers with highest number of purchases
4. Visualized most purchased products
5. Analyzed the distribution of ratings to understand user rating behavior


Correlation Matrix
order_products_value
order_freight_value
order_items_qty
order_sellers_qty
customer_zip_code_prefix
product_name_lenght
product_description_lenght
product_photos_qty
review_score
total_order_value
approval_time
delivery_time
product_category_encoded


Top 20 Highest Rated Products
Product Name
Average Rating
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


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


Top 20 Most Purchased Products
Number of Purchases
Product ID
99a4788cb24856965c36a24e339b6058
aca2eb7d00ea1a7b8ebd4e68314663af
422879e10f46682990de24d770e7f83d
d1c427060a0f73f6b889a5c7c61f2ac4
389d119b48cf3043d311335e499d9c6b
53759a2ecddad2bb87a079a1f1519f73
53b36df67ebb7c41585e8d54d6772e08
368c6c730842d78016ad823897a372db
2b4609f8948be18874494203496bc318
154e7e31ebfa092203795c972e5804a6
7c1bd920dbdf22470b68bde975dd3ccf
e0d64dcfaa3b6db5c54ca298ae101d05
bb50f2e236e5eea0100680137654686c
a92930c327948861c015c919a0bcb4a8
6cdd53843498f92890544667809f1595
d017a2151d543a9885604dc62a3d9dcc
e53e557d5a159f5aa2c5e995dfdf244b
42a2c92a0979a949ca4ea89ec5c7b934
35afc973633aaeb6b877ff57b2793310
c4baedd846ed09b85f78a781b522f126


Rating Ranges and Their Counts
Count
Rating Range
0-1
1-2
2-3
3-4
4-5

# 3. Model Development

## Collaborative Filtering

- We have used SVD (Singular Value Decomposition) for implementing collaborative filtering model.

- SVD - **Singular Value Decomposition** is a mathematical technique used to decompose a matrix into three smaller matrices that capture its essential structure.

### Collaborative Filtering

```
[ ]  # Prepare the dataset for Surprise
     reader = Reader(rating_scale=(1, 5))
     interaction_data = data[['customer_id', 'product_id', 'review_score']].dropna()
     interaction_dataset = Dataset.load_from_df(interaction_data, reader)

     # Split into train and test sets
     trainset, testset = surprise_split(interaction_dataset, test_size=0.2)

     # Train the SVD model
     svd = SVD()
     svd.fit(trainset)

     # Evaluate the model
     predictions = svd.test(testset)
     rmse = accuracy.rmse(predictions)
     print(f"Collaborative Filtering RMSE: {rmse:.4f}")
```

```
RMSE: 1.3484
Collaborative Filtering RMSE: 1.3484
```

# Content-Based Filtering

- We have used TF-IDF (Term Frequency - Inverse Document Frequency) for implementing content based filtering model.
- TF-IDF (**Term Frequency-Inverse Document Frequency**) is a statistical measure used to evaluate the importance of words (or features) in a document (or item) relative to a collection of documents.

Content Based Filtering

```python
# Content Based Filtering
# TF-IDF on product_category_name
tfidf = TfidfVectorizer()
tfidf_matrix = tfidf.fit_transform(data['product_category_name'])

# Calculate average review score and number of reviews for each product
# ----> This is the fix to create the necessary columns
product_review_data = data.groupby('product_id')['review_score'].agg(['mean', 'count'])
product_review_data.columns = ['avg_review_score', 'num_reviews']
data = data.merge(product_review_data, on='product_id', how='left')
# <---- End of fix

# Combine features
numeric_features = data[['avg_review_score', 'num_reviews']].fillna(0)
numeric_features_scaled = scaler.fit_transform(numeric_features)
combined_features = hstack([tfidf_matrix, csr_matrix(numeric_features_scaled)])

# Compute cosine similarity dynamically
def recommend_products_content(product_id, feature_matrix, product_ids, n=5):
    product_idx = product_ids.index(product_id)
    product_vector = feature_matrix[product_idx]
    similarity_scores = cosine_similarity(product_vector, feature_matrix).flatten()
    similar_indices = similarity_scores.argsort()[::-1][1:n+1]
    similar_product_ids = [product_ids[i] for i in similar_indices]
    return similar_product_ids

product_ids = data['product_id'].tolist()
```

# Hybrid Model

- Hybrid model can handle challenges such as cold start problems, sparsity issues and scalability.

- **Random Forest** is an ensemble machine learning algorithm that is used for both classification and regression tasks. It works by creating a collection of decision trees during training and outputs the prediction based on the majority vote or average from all the individual trees in the forest.

```python
rf = RandomForestRegressor(n_estimators=100, random_state=42)
rf.fit(X_train, y_train)

# 7. Evaluate Model
y_pred = rf.predict(X_test)
rmse = np.sqrt(mean_squared_error(y_test, y_pred))
print(f"Random Forest RMSE: {rmse:.4f}")

# ... (Previous code for model training and evaluation) ...

# 8. Prediction for a Specific Customer (Enhanced)
def recommend_products_for_customer(customer_id, top_n=5):
    """Recommends products for a given customer.

    Args:
        customer_id (str): The ID of the customer.
        top_n (int, optional): The number of products to recommend. Defaults to 5.

    Returns:
        list: A list of recommended product IDs.
    """
    specific_customer_data = data[data['customer_id'] == customer_id]

    if not specific_customer_data.empty:  # Existing customer
        pred_ratings = rf.predict(specific_customer_data[all_features])
        top_indices = np.argsort(pred_ratings)[-top_n:][::-1]
        recommended_products = specific_customer_data['product_id'].iloc[top_indices].tolist()
    else:  # New customer (cold start)
        # Recommend popular products or products from similar categories
        # (You'll need to implement this logic based on your data)
        # For example, you could recommend the top-rated products overall:
        popular_products = data.groupby('product_id')['review_score'].mean().sort_values(ascending=False).index
        recommended_products = popular_products[:top_n].tolist()

    print(f"Top {top_n} recommended products for customer {customer_id}:")
    for product_id in recommended_products:
        product_name = data[data['product_id'] == product_id]['product_category_name'].iloc[0]
```

# 4. Evaluate, Save and Load Model

```
[ ] # Evaluate the model
    y_pred = rf.predict(X_test)
    rmse = np.sqrt(mean_squared_error(y_test, y_pred))
    print(f"Hybrid Model RMSE: {rmse:.4f}")
```

```
Hybrid Model RMSE: 1.2530
```

## Save the Models

```
[ ] import joblib  # Import joblib for saving and loading models

    # Saving Content-Based Filtering components
    joblib.dump(tfidf, 'content_based_tfidf.pkl')
    joblib.dump(scaler, 'content_based_scaler.pkl')

    # Saving Collaborative Filtering (SVD) model
    joblib.dump(svd, 'collaborative_filtering_model.pkl')

    # Saving Hybrid Model (RandomForestRegressor)
    joblib.dump(rf, 'hybrid_model.pkl')
```

```
['hybrid_model.pkl']
```

## Load the Saved Models

```
[ ] import joblib

    # Content-Based Filtering components
    loaded_tfidf = joblib.load('content_based_tfidf.pkl')
    loaded_scaler = joblib.load('content_based_scaler.pkl')

    # Collaborative Filtering (SVD) model
    loaded_svd = joblib.load('collaborative_filtering_model.pkl')

    # Hybrid Model (RandomForestRegressor)
    loaded_rf = joblib.load('hybrid_model.pkl')
```

## Using the Loaded Models

```
[ ] predictions = loaded_svd.test(testset)  # Use loaded_svd for prediction

    # Example usage for Hybrid Model
    # ... (Prepare your data with content and latent features) ...
    y_pred = loaded_rf.predict(X_test)  # Use loaded_rf for prediction
```

# Results

- New User

```
Random Forest RMSE: 1.2530
Enter Cust...  12345
Get Recommendati...
Top 5 recommended products for customer 12345:
fff9553ac224cec9d15d49f5a263411f - fashion_bolsas_e_acessorios
49cd6408393770922f19ca2925832dcd - beleza_saude
a84c7b893ea37674ef896fa866793e7d - cama_mesa_banho
a84a9e06ec9e2e2a520aa8c5b6d11150 - relogios_presentes
a84531e9148e6c8066bfbd6ab814e830 - market_place
```

- Existing User

```
Random Forest RMSE: 1.2530
Enter Cust...  0c9ff9d8ed9b9bdd825487b3a66e
Get Recommendati...
Top 5 recommended products for customer 0c9ff9d8ed9b9bdd825487b3a66e05f5:
fff9553ac224cec9d15d49f5a263411f - fashion_bolsas_e_acessorios
49cd6408393770922f19ca2925832dcd - beleza_saude
a84c7b893ea37674ef896fa866793e7d - cama_mesa_banho
a84a9e06ec9e2e2a520aa8c5b6d11150 - relogios_presentes
a84531e9148e6c8066bfbd6ab814e830 - market_place
```

# Future Enhancements

- **Product Summarizer** - Product summarizers provide concise overviews of the content within a dataset.
- **Context-Aware Recommendations -** Enhancing recommendations by considering contextual information such as user location, time of day and user activity.
- **Diverse Algorithms -** The hybrid model is already a robust system, however experiments can be done by implementing various concepts like Neural Networks, AutoML.

# References:


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

# THANK YOU!!