Project Report

On

Personalized Products Recommendation Model

Submitted
In partial fulfilment
For the award of the Degree of

PG-Diploma in Big Data Analytics

(C-DAC, ACTS (Pune))

Guided By:

Mr. Prakash Sinha

Submitted By:

Bhagyashri Golhar (240340125015)

Sumati Bhanage (240340125016)

Sakshi Bhoyar (240340125017)

Sonali Nayak (240340125048)

Suraj Mahalle (240340125050)

Vaishali Hanwate (240340125055)

Centre for Development of Advanced Computing

(C-DAC), ACTS (Pune- 411008)

Acknowledgement

This is to acknowledge our indebtedness to our Project Guide, **Mr. Prakash Sinha,** C-DAC ACTS, Pune for her constant guidance and helpful suggestion for preparing this project **Personalized Product Recommendation model.** We express our deep gratitude towards his for inspiration, personal involvement, constructive criticism that he provided us along with technical guidance during the course of this project.

We take this opportunity to thank Head of the department **Mr. Gaur Sunder** for providing us such a great infrastructure and environment for our overall development.

We express sincere thanks to **Mrs. Namrata Ailawar (Process Owner)** for their kind cooperation and extendible support towards the completion of our project.

It is our great pleasure in expressing sincere and deep gratitude towards **Mrs. Risha P R (Program Head)** and **Ms. Pratiksha Gacche (Course Coordinator, PG-DBDA)** for their valuable guidance and constant support throughout this work and help to pursue additional studies.

Also, our warm thanks to **C-DAC ACTS Pune**, which provided us this opportunity to carry out, this prestigious Project and enhance our learning in various technical fields.

Bhagyashri Golhar (240340125015)

Sumati Bhanage (240340125016)

Sakshi Bhoyar (240340125017)

Sonali Nayak (240340125048)

Suraj Mahalle (240340125050)

Vaishali Hanwate (240340125055)

ABSTRACT

This machine learning-based project provides personalized product recommendations to users based on their browsing and purchase history. The system utilizes collaborative filtering and content-based filtering algorithms to analyze user behavior and generate relevant recommendations. In the ever-evolving landscape of e-commerce, personalized product recommendations have become critical for optimizing the shopping experience and driving sales growth. This study introduces a comprehensive exploration and implementation of a collaborative filtering (CF) recommendation system. The goal is to fine-tune product recommendations to meet user preferences effectively. By implementing the recommendation system in both PySpark and Python, this project provides insights into the efficiency and scalability of distributed computing. It aims to identify the most effective approach for personalized product recommendations. This project sheds light on the benefits of using PySpark for large-scale data processing in recommendation systems

Table of Contents

Chapter 1
Introduction

1.1 Introduction

This project builds a smart recommendation system for an e-commerce website. It uses data from user actions like what they view, add to their cart, and purchase to suggest products they might like. This recommendation engine uses implicit feedback data from an e-commerce website, which includes user interactions such as views, carts, and purchases.

By analyzing user interactions, we create an interaction matrix and use a method called Alternating Least Squares (ALS) which is an approach to Collaborative Filtering to recommend products that each user is most likely to be interested in, that is Personalized Product Suggestions.

The goal is to improve user experience, increase satisfaction, and boost sales and profits for the e-commerce platform. The engine provides personalized recommendations to users by using matrix factorization with the Alternating Least Squares algorithm along with other features such as purchase per view. Additionally, it identifies substitute and supplementary products by comparing their vectors using the Locality Sensitive Hashing algorithm. By improving the user experience and increasing customer satisfaction and retention, the project aimes to increase sales and profit on the platform. The highly personalized nature of the recommendation scores make this engine a valuable tool for any e-commerce company looking to improve their product recommendations.

Alternating Least Square– ALS is an optimization algorithm used to factorize large matrices, especially in recommendation systems. It's widely applied for collaborative filtering tasks, where we predict missing entries in a user-item interaction matrix. Imagine we have a user-item matrix (e.g., Event Types of Product Recommendation). ALS decomposes this matrix into two lower-dimensional matrices: one for users and

one for items. These matrices capture latent features (or factors) that explain user preferences and item characteristics.

Personalized recommendations are data-driven suggestions provided to users based on their preferences, behavior, and historical interactions.These recommendations enhance the user experience by tailoring product suggestions to individual tastes. Recommendation models analyze user-item interactions (e.g., views, purchases) to predict relevant items.

1.2 Objective

The objectives of the project work are as -

- To Enhance User Experience.
- To Increase Sales and Revenue.
- To Improve Customer Retention and Loyalty.
- To understand User Preferences and Trends.

personalized product recommendation models aim to create a win-win situation: users find relevant products, and businesses achieve higher sales and customer satisfaction. It’s like having a knowledgeable shopping assistant who knows users taste and guides users to the perfect items. By analyzing user interactions with recommended products, businesses gain insights into customer preferences and emerging trends. The primary goal is to provide personalized and relevant product suggestions to each user.By analyzing user behavior (such as browsing history, past purchases, and preferences), the model tailors recommendations to individual tastes.

Chapter 2
LITERATURE REVIEW

Sarwar et al. *(Collaborative filtering*, 2001) recommends products based on the preferences of similar users. Machine learning models like matrix factorization or nearest neighbor algorithms are used to identify users with similar preferences and suggest popular items among those users.

Burke, R. (User Modeling and User-Adapted Interaction, 2002): This article provides an early yet influential survey of hybrid recommender systems, which combine multiple recommendation techniques to improve performance. The paper categorizes various hybrid models and presents empirical results to demonstrate their effectiveness.

Lops, P., de Gemmis, M., & Semeraro, G. (Recommender Systems Handbook, 2011): This chapter in the "Recommender Systems Handbook" offers an in-depth review of content-based filtering techniques. It discusses the methodologies behind content-based recommendations, their applications, and the challenges associated with them, such as feature extraction and overspecialization.

Bobadilla, J., Ortega, F., Hernando, A., & Gutiérrez, A. (Knowledge-Based Systems, 2013): This review provides an extensive overview of personalized recommender systems, focusing on collaborative filtering, hybrid methods, and emerging techniques. It also discusses the challenges faced by these systems, such as scalability

and cold start, and suggests possible extensions for future research.

Berkovsky, S., Eirinaki, M., & Vazirgiannis, M. (Recommender Systems Handbook, 2015): This chapter provides an overview of privacy-preserving techniques in recommender systems. It discusses methods like anonymization, data perturbation, and differential privacy, focusing on how these approaches can protect user data while still delivering personalized recommendations.

Khanal SS, Prasad PWC, Alsadoon A, Maag A (2020) A systematic review: machine learning based recommendation systems for e-learning.

Shokrzadeh Z, Feizi-Derakhshi MR, Balafar MA, Mohasefi JB (2023) Knowledge graph-based recommendation system enhanced by neural collaborative filtering and knowledge graph embedding.

Chapter 3

Methodology and Techniques

3.1 Methodology:

1. **Raw Data:**
    - The process begins with raw, unprocessed data. This could be anything from user interactions on a website (clicks, purchases, ratings) to sensor data from IoT devices.
2. **Transformation & Actions (Using PySpark):**
    - The raw data undergoes transformation using PySpark, a powerful tool for large-scale data processing.
    - This step involves cleaning, aggregating, and structuring the data to make it suitable for analysis.
3. **Exploratory Data Analysis (EDA):**
    - The upper path in the workflow leads to EDA. Here, data scientists explore the cleaned data to uncover patterns, correlations, and potential insights.
    - Techniques like data visualization, statistical summaries, and feature engineering are used**.**
4. **Clean & Modified Data:**
    - The cleaned and modified data (resulting from EDA) moves forward. It's now in a more usable format for subsequent steps.
5. **Recommendation (Alternating Least Squares - ALS):**
    - The lower path focuses on recommendation. ALS is an algorithm commonly used for collaborative filtering in recommendation systems.
    - ALS predicts missing entries in a user-item interaction matrix, enabling personalized recommendations.

6. **MySQL Database:**
    - The processed data is stored in a MySQL database. This step ensures persistence and efficient retrieval for future use.
7. **Visualization (Using Tableau):**
    - Finally, the workflow culminates in visualization. Tableau, a popular data visualization tool, is employed to create interactive dashboards.
    - These dashboards allow users to explore insights, trends, and recommendations visually.

This workflow takes raw data, transforms it, analyzes it, generates recommendations, stores it in a database, and presents visual insights.

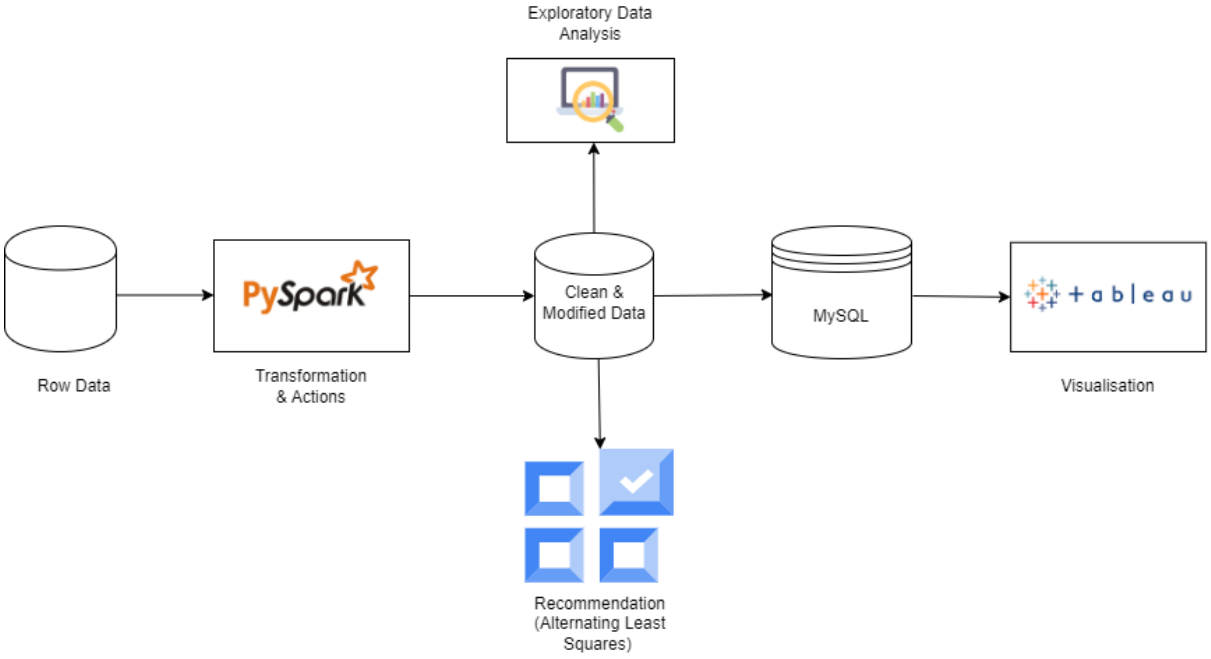


Fig.1 Architectural Workflow

3.2 Dataset

The Personalized Products Recommendation Dataset contains behavior data for 1 months (April 2020) from a large multi-category online store.

Each row in the file represents an event. All events are related to products and users. Each event is like many-to-many relation between products and users.

Event_time column represents the timestamp when the event occurred. It indicates the exact moment when a user performed an action related to a product (e.g., viewing, purchasing).

Event_type column Describes the type of event associated with the product. In your example, the event type is “view,” which means the user viewed the product.

Other possible event types include “purchase,” which means the user purchased the product and “cart,” which means the user add the products to cart .

product_id is a unique identifier for the product. This allows you to track specific products across different events.

category_id represents the category to which the product belongs.

category_code is an optional field that provides a more human-readable version of the product category. For instance, “electronics.smartphone” indicates that the product is a smartphone in the electronics category.

Brand indicates the brand or manufacturer of the product.

The price of the product. Price data is crucial for various analyses, including recommendations and understanding user behavior based on affordability.

user_id is a unique identifier for the user who interacted with the product. Each user has their own user ID.User IDs allow tracking individual behavior and personalization.

user_session represents a session or interaction session for a user. A session groups together related events performed by the same user.Sessions help analyze user behavior within a specific timeframe.

3.1.3 Model Description

Preprocessing-

These preprocessing steps are essential for preparing the data for further analysis, modeling, or visualization. By converting data types, handling cart events, and extracting meaningful category information, you've set the stage for more insightful exploration.
The dataset contains events of different types, including "view" and "cart."The code separates out the cart events (cart_df) and removes them from the main dataset (df). This ensures that each user-product pair has at most one cart event per session.
The category_code column contains hierarchical information (e.g., "electronics.smartphone").The code splits this column into three new columns: Category, Sub_category, and Product.

- Category: Represents the top-level category (e.g., "electronics").
- Sub_category: Represents the sub-category (e.g., "smartphone").
- Product: Represents the specific product within the sub-category.

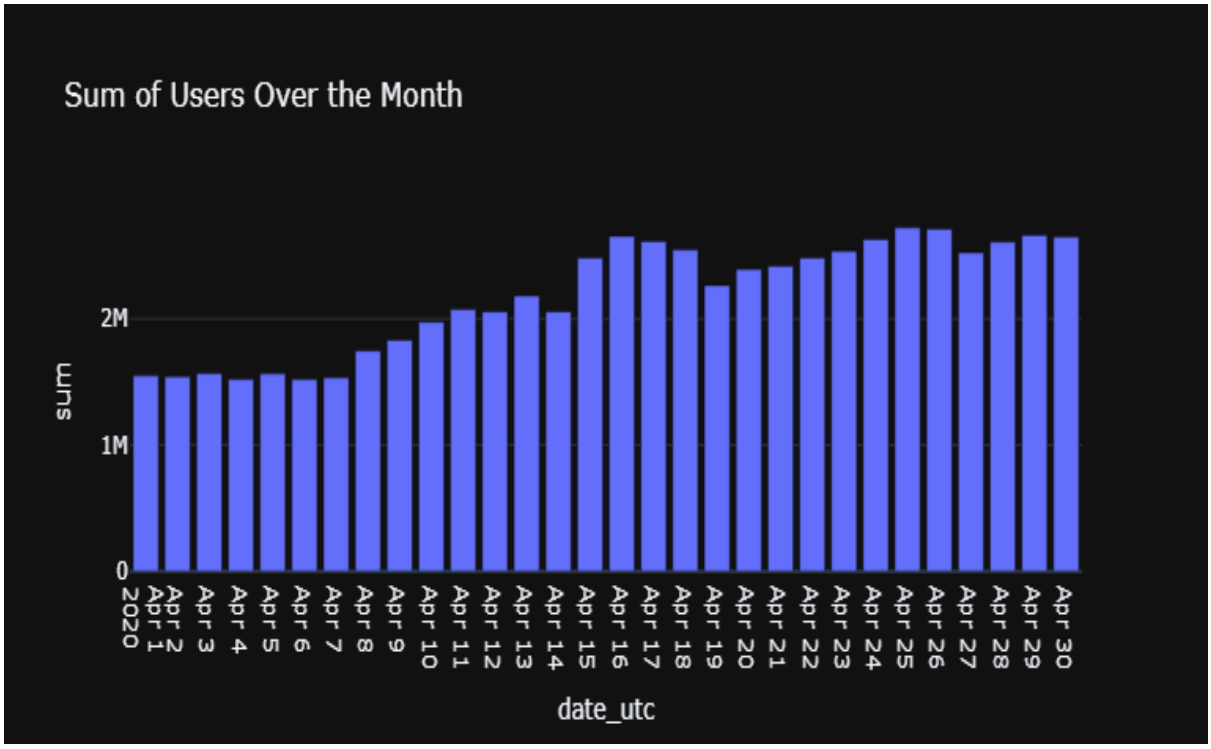
The resulting processed DataFrame is stored in df_processed.

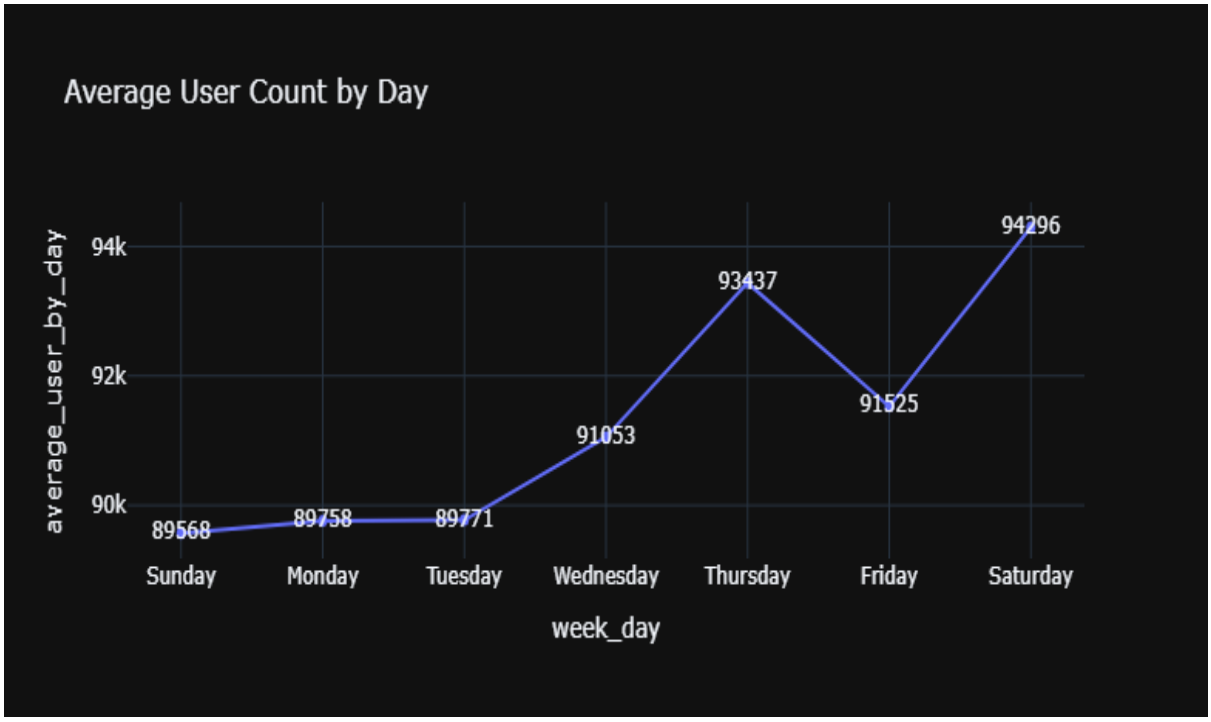
Customer Analysis-

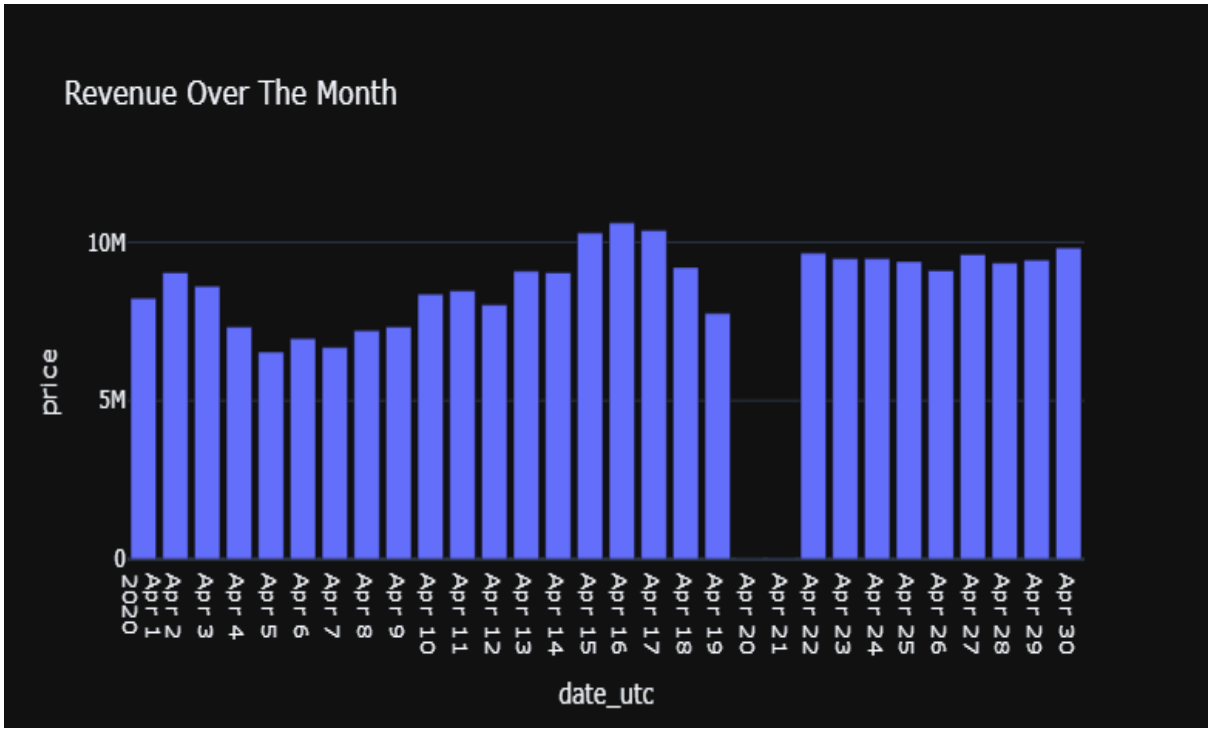
It convert event_time to Date, it extracts the date portion from the event_time column and assigns it to the new column.The to_date function converts the timestamp to a date format.

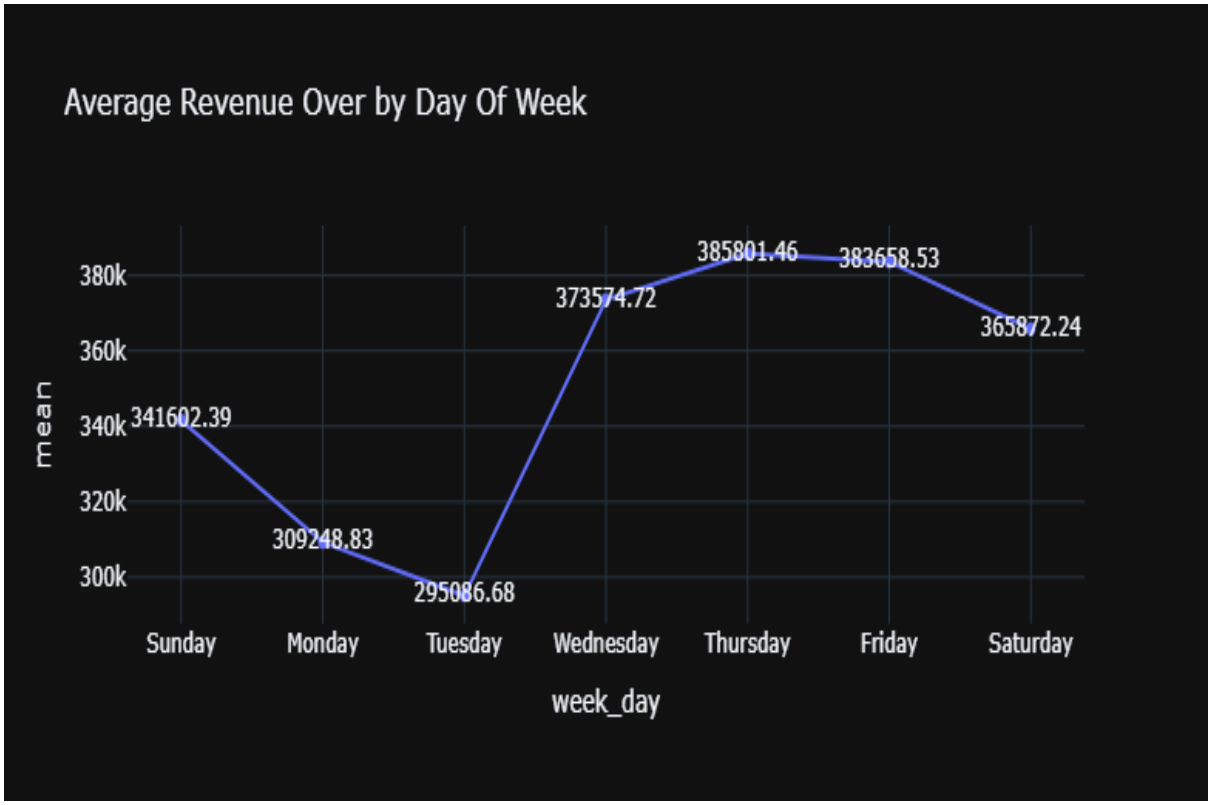
Extract Hour from event_time it extracts the hour component from the event_time column.The hour function returns the hour (0-23) from the timestamp.
This code prepares the data for customer analysis by extracting the date and hour components from the event_time column and creating new columns for these values. The resulting DataFrame, df_apr, now includes the processed date and hour information.

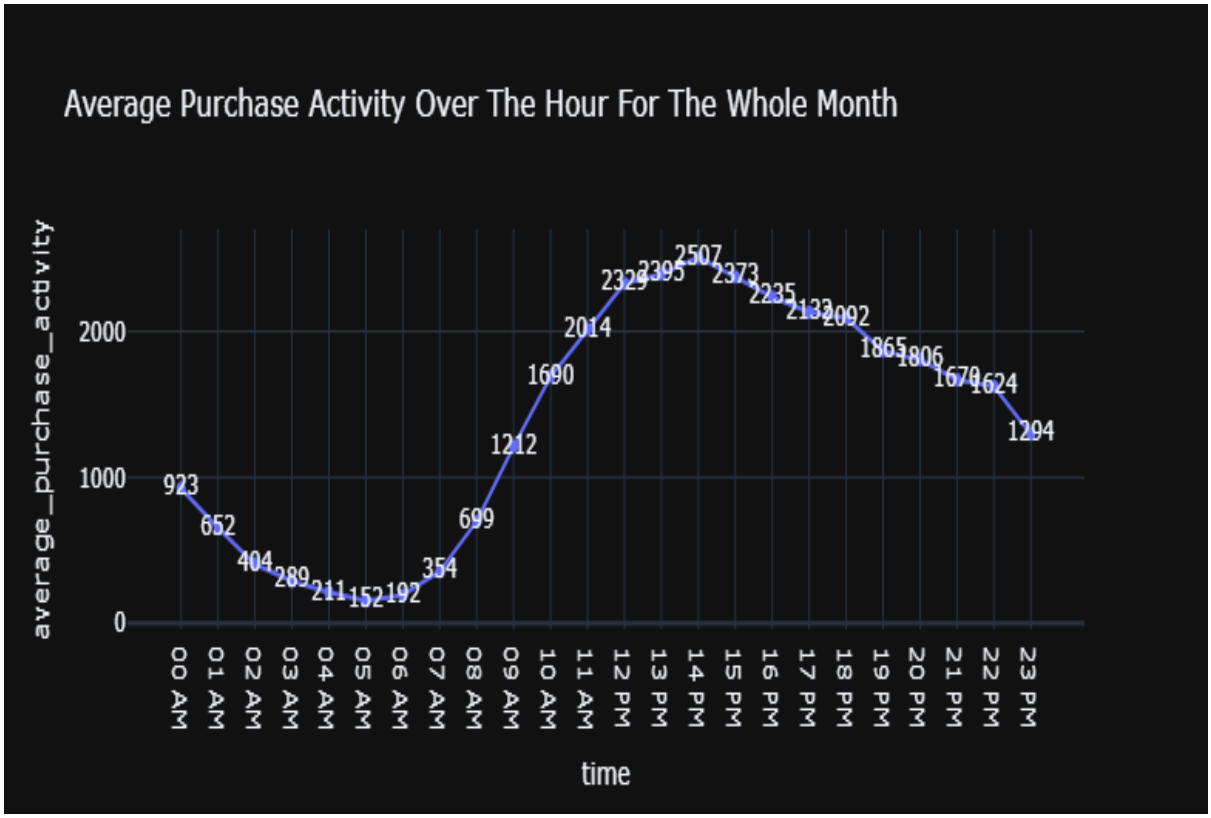
Revenue Analysis-

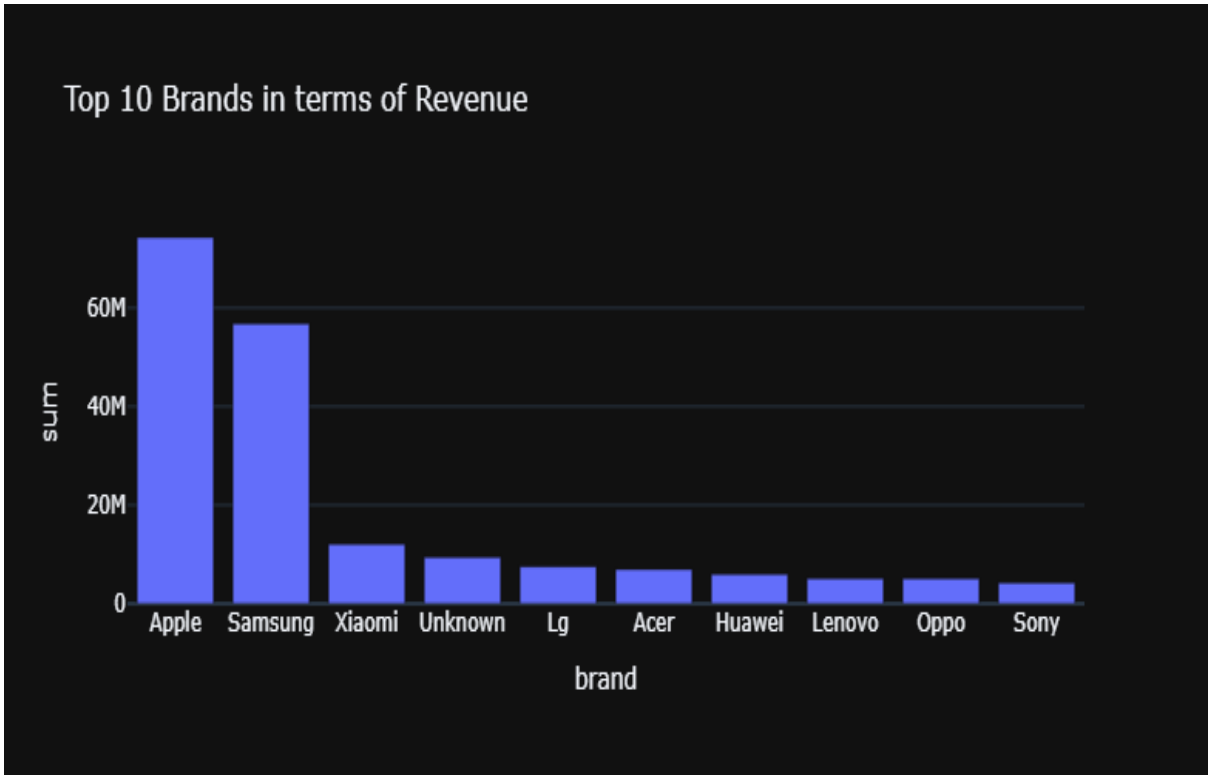


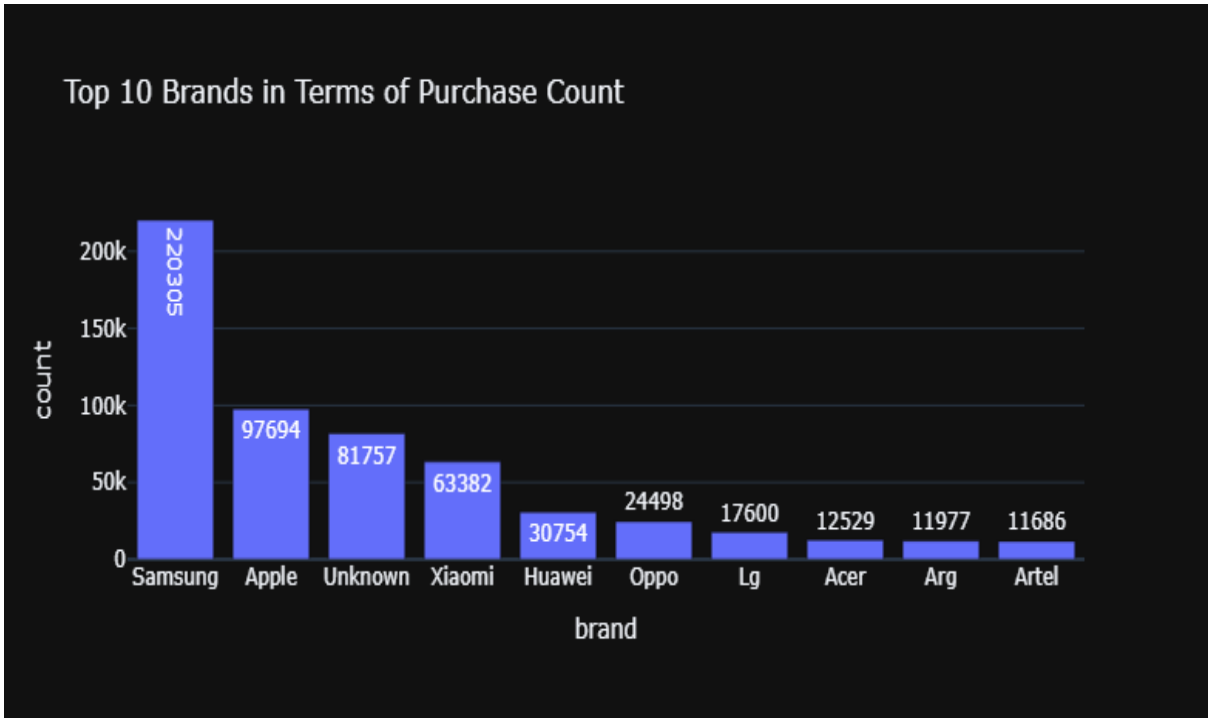


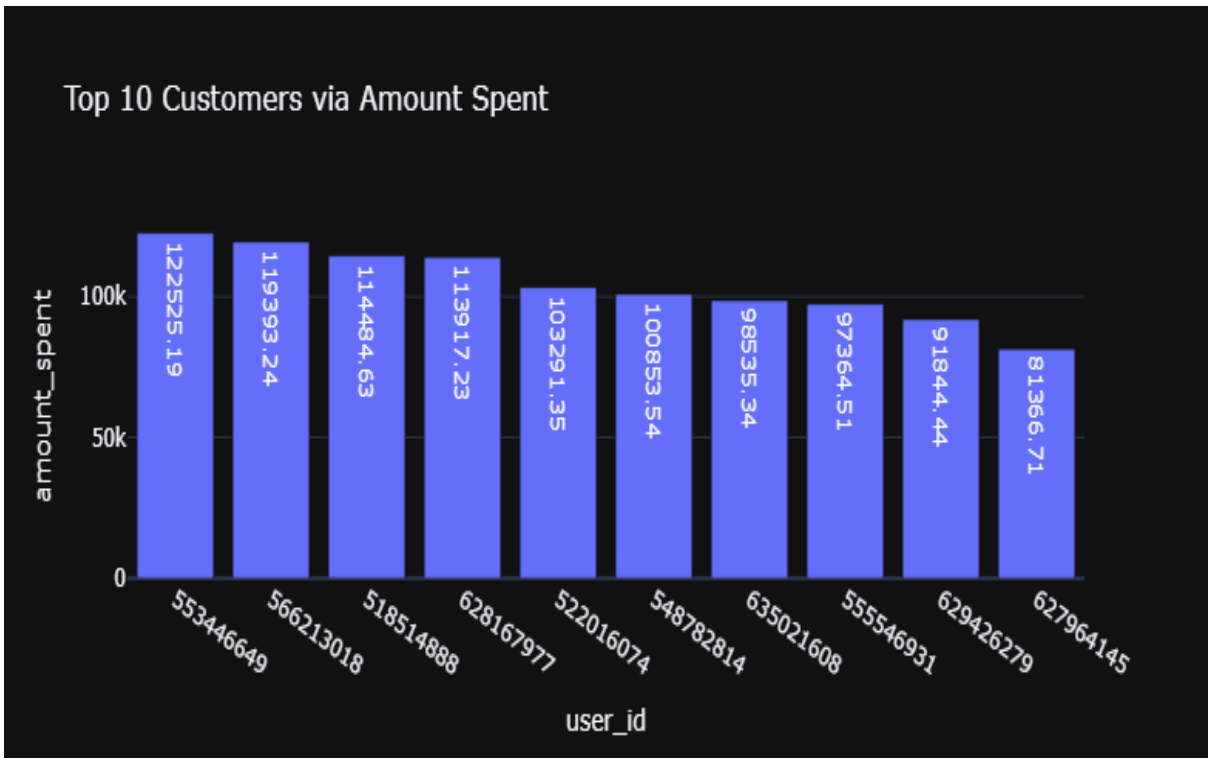

Revenue Over The Month
10M
5M
0
price
2020
Apr 1
Apr 2
Apr 3
Apr 4
Apr 5
Apr 6
Apr 7
Apr 8
Apr 9
Apr 10
Apr 11
Apr 12
Apr 13
Apr 14
Apr 15
Apr 16
Apr 17
Apr 18
Apr 19
Apr 20
Apr 21
Apr 22
Apr 23
Apr 24
Apr 25
Apr 26
Apr 27
Apr 28
Apr 29
Apr 30
date_utc


Average Revenue Over by Day Of Week
380k
360k
340k
320k
300k
mean
341602.39
309248.83
295086.68
373574.72
385801.46
383658.53
365872.24
Sunday
Monday
Tuesday
Wednesday
Thursday
Friday
Saturday
week_day


Average Purchase Activity Over The Hour For The Whole Month
average_purchase_activity
2000
1000
0
923
652
404
289
211
152
192
354
699
1212
1690
2014
2329
2395
2507
2373
2235
2133
2092
1865
1806
1670
1624
1294
00 AM
01 AM
02 AM
03 AM
04 AM
05 AM
06 AM
07 AM
08 AM
09 AM
10 AM
11 AM
12 PM
13 PM
14 PM
15 PM
16 PM
17 PM
18 PM
19 PM
20 PM
21 PM
22 PM
23 PM
time


Top 10 Brands in terms of Revenue
sum
60M
40M
20M
0
Apple
Samsung
Xiaomi
Unknown
Lg
Acer
Huawei
Lenovo
Oppo
Sony
brand


Top 10 Brands in Terms of Purchase Count
count
200k
150k
100k
50k
0
220305
97694
81757
63382
30754
24498
17600
12529
11977
11686
Samsung
Apple
Unknown
Xiaomi
Huawei
Oppo
Lg
Acer
Arg
Artel
brand


Top 10 Customers via Amount Spent
amount_spent
100k
50k
0
122525.19
119393.24
114484.63
113917.23
103291.35
100853.54
98535.34
97364.51
91844.44
81366.71
553446649
566213018
518514888
628167977
522016074
548782814
635021608
555546931
629426279
627964145
user_id

Chapter 4

Implementation

1. Use of PySpark Platform for writing the code with MLlib, MySQL, Tableau
2. Hardware and Software Configuration:

Hardware Configuration:

- CPU: 8 GB RAM, Quad core processor
- GPU: 16GB RAM **Nvidia's GTX 1080Ti**

Software Required:

- **Anaconda**: It is a package management software with free and open-source distribution of the Python and R programming language for scientific computations (data science, machine learning applications, large-scale data processing, predictive analytics, etc.), that aims to simplify deployment.
- **Jupyter Lab**:
  Jupyter is a web-based interactive development environment for Jupyter notebooks, code, and data.
  Jupyter is flexible: configure and arrange the user interface to support a wide range of workflows in data science, scientific computing, and machine learning.
  Jupyter is extensible and modular: write plugins that add new components and integrate with existing ones.

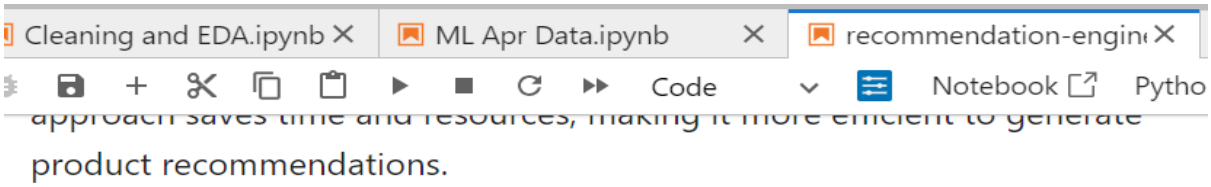
Preprocessing:

Preprocessing

[17]:
```
from pyspark.sql.functions import to_timestamp
def preprocess(df):

    # Change data types
    df = df.withColumn('event_time', to_timestamp('event_time'))
    df = df.withColumn('user_id', col('user_id').cast('integer'))
    df = df.withColumn('product_id', col('product_id').cast('integer'))
    df = df.withColumn('category_id', col('category_id').cast('long'))

    # Limit the number of carts to 1 per session for each user-product pair
    cart_df = df.filter(col('event_type') == 'cart')
    df = df.filter(col('event_type') != 'cart')
    cart_df = cart_df.dropDuplicates(subset=['product_id', 'user_id', 'user_session'])
    df = df.union(cart_df)

    # Split category codes into sub categories
    from pyspark.sql.functions import split

    df = (df
      .withColumn('Category', split(df['category_code'], "\\.").getItem(0))
      .withColumn('Sub_category', split(df['category_code'], "\\.").getItem(1))
      .withColumn('Product', split(df['category_code'], "\\.").getItem(2)))

    return df

df_processed = preprocess(df_apr)
```

[19]:
```
df_processed.show()
```

ALS Algorithm:



[26]:

```
from pyspark.ml.recommendation import ALS
from pyspark.ml.evaluation import RegressionEvaluator

def simple_als(interaction_matrix):

    # Train-test split
    (train, test) = interaction_matrix.randomSplit([0.8, 0.2])

    # Initialize the model with the optimized parameters
    als = ALS(userCol='user_id', itemCol='product_id', ratingCol='interac
              alpha=1, regParam=0.005, rank=15, implicitPrefs=True,
              nonnegative=True, coldStartStrategy='drop')

    # Fit the ALS model on the ratings data
    model = als.fit(train)

    # Make predictions
    predictions = model.transform(test)

    # Calculate the RMSE and MAE metrics
    evaluator = RegressionEvaluator(metricName='rmse', labelCol='interact
    rmse = evaluator.evaluate(predictions)
    mae = evaluator.setMetricName('mae').evaluate(predictions)
    print('test rmse:' + str(rmse) + ' mae:' + str(mae))

    return model
```

[27]:

```
als_model = simple_als(interaction_matrix)
```

```
[Stage 779:=================================================>     (9 +
1) / 10]
test rmse:0.09362263416331866 mae:0.05455085317688715
```

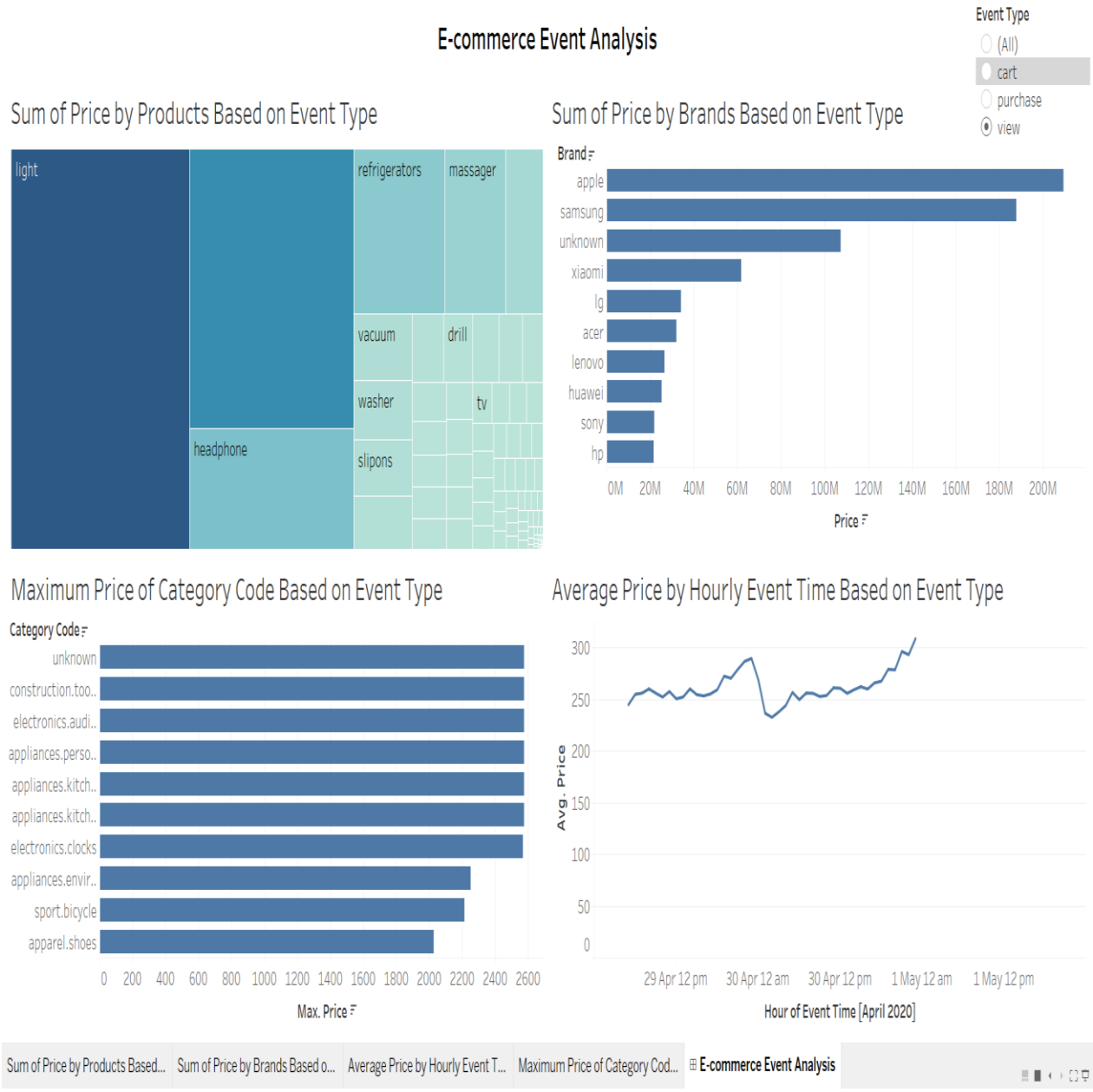
Chapter 5

Results

[28]:
```
# 3 random users
user_subset = [565606905, 570112140, 564068124]

# Recommend top 500 products for the users
recommendations = sc.createDataFrame([(user, 0) for user in user_subset], ['user_id', 'product_id'])
recommendations = als_model.recommendForUserSubset(recommendations, 500)
```

[29]:
```
recommendations.show()
```

```
[Stage 835:=================================================> (310 + 8) / 320]
+---------+--------------------+

|  user_id|     recommendations|

+---------+--------------------+

|570112140|[{1004833, 0.8508...|

|564068124|[{1004870, 1.1288...|

|565606905|[{1004767, 0.9786...|

+---------+--------------------+
```


E-commerce Event Analysis
Event Type
(All)
cart
purchase
view
Sum of Price by Products Based on Event Type
light
refrigerators
massager
vacuum
drill
washer
tv
headphone
slipons
Sum of Price by Brands Based on Event Type
Brand
apple
samsung
unknown
xiaomi
lg
acer
lenovo
huawei
sony
hp
0M 20M 40M 60M 80M 100M 120M 140M 160M 180M 200M
Price
Maximum Price of Category Code Based on Event Type
Category Code
unknown
construction.too..
electronics.audi..
appliances.perso..
appliances.kitch..
appliances.kitch..
electronics.clocks
appliances.envir..
sport.bicycle
apparel.shoes
0 200 400 600 800 1000 1200 1400 1600 1800 2000 2200 2400 2600
Max. Price
Average Price by Hourly Event Time Based on Event Type
Avg. Price
300
250
200
150
100
50
0
29 Apr 12 pm
30 Apr 12 am
30 Apr 12 pm
1 May 12 am
1 May 12 pm
Hour of Event Time [April 2020]
Sum of Price by Products Based...
Sum of Price by Brands Based o...
Average Price by Hourly Event T...
Maximum Price of Category Cod...
E-commerce Event Analysis

Chapter 6

Conclusion

6.1 Conclusion

Personalized product recommendation models have evolved significantly over the past decades, becoming essential tools for enhancing user experience and driving business growth in e-commerce and digital content platforms. The field has progressed from basic techniques like collaborative filtering and content-based filtering to more advanced models incorporating deep learning, matrix factorization (such as Alternating Least Squares), and hybrid methods.

Collaborative Filtering (CF) remains a cornerstone of recommendation systems, particularly effective due to its ability to leverage user interactions and collective behavior. However, CF faces challenges such as the cold start problem and data sparsity, which have led to the development of more sophisticated approaches, including matrix factorization techniques like ALS. **ALS**, in particular, has shown great promise in scaling recommendations to large datasets and handling implicit feedback, making it a valuable tool in modern recommender systems.

The integration of deep learning and **interaction matrices** has further enhanced the capability of recommendation systems to capture complex, non-linear relationships between users and items, improving both accuracy and diversity of recommendations. Despite these advancements, several challenges persist, such as ensuring fairness, protecting user privacy, and providing explainable recommendations that build trust with users.

The model can be adapted for scalability, better convergence, and better accuracy.

6.2 Future Enhancement –

- **Explainability and Transparency**: As recommendation systems become more complex, providing users with understandable explanations for recommendations will become increasingly important. Future models are expected to incorporate explainable AI techniques that can articulate why certain products are recommended, enhancing user trust and satisfaction.

  Transparency Tools: Developing tools and frameworks that offer insights into the decision-making processes of recommendation models will help users and developers understand and control the recommendations better.
- **Federated Learning**: This approach allows models to be trained across multiple decentralized devices without sharing raw data, thereby protecting user privacy. Future recommendation systems may widely adopt federated learning to enhance privacy without compromising the quality of recommendations.

Chapter 7

References

[1] **Title**: "Item-based Collaborative Filtering Recommendation Algorithms".
Badrul M. Sarwar, George Karypis, Joseph Konstan, John Riedl: Proceedings of the 10th International Conference on World Wide Web (WWW), 2001.

[2] **Title**: "Large-Scale Parallel Collaborative Filtering for the Netflix Prize".
Yunhong Zhou, Dennis Wilkinson, Robert Schreiber, Rong Pan: Proceedings of the 4th ACM Conference on Recommender Systems (RecSys), 2008

[3] **Title**: "Matrix Factorization Techniques for Recommender Systems"
Yehuda Koren, Robert Bell, Chris Volinsky: IEEE Computer, 2009.

[4] **Title**: "Collaborative Filtering with Implicit Feedback Datasets".
Yifan Hu, Yehuda Koren, Chris Volinsky: Proceedings of the 8th IEEE International Conference on Data Mining (ICDM), 2008

[5] **Title**: "A Survey of Collaborative Filtering Based Recommender Systems: From Traditional Methods to Hybrid Models".
Su, X., & Khoshgoftaar, T. M.: Artificial Intelligence Review, 2009

[6] **Title**: "Collaborative Filtering for Implicit Feedback Datasets".
Steffen Rendle, Lars Schmidt-Thiem: Proceedings of the 7th IEEE International Conference on Data Mining (ICDM), 2007

[7] Collaborative Filtering for Implicit Feedback Datasets

[8] A Survey of Collaborative Filtering Based Recommender Systems

[9] Matrix Factorization Techniques for Recommender Systems