2024

Retail-Rocket Recommender System

University of Texas Arlington

INSY 5337 - WEB AND SOCIAL ANALYTICS PROJECT REPORT
Professor: Riyaz Sikora

Group 6

Shrishankar Shripadarao Desai	1002173907
Vismaya Prakasan	1002154696
Kashish Tarique	1002157068
Swathi Manjunatha	1002162937
Akshay Thakare	1002168718

1.Addendum

Addressing Feedback Questions

In response to the feedback provided, we have incorporated detailed analyses and validations into the final project report. Below is a summary of how each feedback question was addressed, along with the methodologies and calculations involved.

1. What metrics are we using in the project and how are they calculated?

- **Metrics Used:**
  - **View-to-Cart Time:** The time difference between when an item is first viewed and when it is added to the cart.
  - **Cart-to-Purchase Time:** The time difference between adding an item to the cart and completing the purchase.
  - **View Rate:** The percentage of visitors who viewed a specific item relative to the total visitors.
  - **Purchase Rate:** The percentage of unique viewers who purchased an item relative to total unique viewers.
- **Calculation:** These metrics were calculated using timestamp data. For example, cart-to-purchase time was derived as:

time = timestamp(transaction) - timestamp(add_to_cart).

2. Purchased Column Transformation

- The Transaction ID column was transformed into a Purchased column, where a non-null Transaction ID was marked as 1 (indicating a purchase) and null values were marked as 0.
- After this transformation, the Transaction ID column was removed to simplify the dataset and focus analysis on the Purchased column.

3. Item Viewing Patterns

- **Calculation of Views:**
  - Total views of each item were calculated regardless of who viewed them.
  - Unique visitor views for each item were also calculated.
- **Purchase Rate:** Derived as the ratio of unique visitors who purchased an item to the total unique visitors who viewed it. This was supported by detailed tables and visualizations in the report.

4. View Rate Definition

- **Definition:** The view rate was defined as the percentage of visitors who viewed a specific item out of the total visitors.
- **Products Never Viewed:** Items with zero views were identified to highlight gaps in user engagement, and these insights were documented in the results.

5. Aggregation Level

- **Levels of Aggregation:**
    - Aggregations were performed at both the product ID and customer ID levels to provide detailed insights into trends and behaviors.
    - These aggregated features were used in clustering and predictive modeling to capture item- and user-level dynamics.

6. Predicting Purchase Rate

- **Attributes Used:**
    - Features such as view rate, cart-to-purchase time, product category attributes, and interaction durations were used.
    - Feature importance analyses validated these predictors, ensuring their relevance.
- **Methodology:** Predictive models like regression and clustering algorithms were applied to forecast purchase likelihood.

7. Visitor Categorization

- Visitors were categorized into "purchasers" and "non-purchasers" and further segmented into "browsers" and "non-browsers."
- These categorizations were supported by behavioral patterns and segmentation analyses documented in the report.

8. Product Category Attributes

- **View Duration and Total Duration:**
    - If a customer viewed an item five times, the durations were summed to calculate the total view duration.
    - These aggregated metrics were used to assess engagement and predict purchase intent.

9. Total Views and New Variables

- **New Variables Created:**

    - Total views per item, unique visitor count per item, and view-to-cart time.
    - These variables were derived using Python-based data transformations and thoroughly explained in the feature engineering section.

10. Customer ID Analysis

- **Verification:** Customer IDs were confirmed to appear multiple times in the dataset, representing repeat interactions.
- This validation ensured the integrity of clustering and behavioral segmentation analyses.

11. Purchase Validation

- **Validation Process:**
    - Each purchase was verified to have at least one corresponding "add to cart" action.
    - Multiple view events for the same item and visitor were validated to occur before the purchase.
    - These validations were conducted through chronological data checks and are detailed in the methodology section.

Visualization Enhancements

To support these analyses, we incorporated the following visualizations:

- **Comparison Charts:** Highlighting differences in model performance before and after addressing outliers.
- **Scatterplots:** Depicting metrics like cart-to-purchase time and its correlation with purchase rates.
- **Clustering Visualizations:** Segmenting users into actionable categories, such as browsers and loyal buyers.
- **Feature Importance Plots:** Emphasizing key predictors of purchase success.

The final project report integrates these enhancements and addresses all feedback questions comprehensively. Through detailed calculations, validations, and visualizations, we ensured that the analysis is robust and actionable, providing a clear pathway for optimizing eCommerce strategies

Adjusted Research Questions

1. **How Can Customer Actions (e.g., Viewing Patterns, Cart Behaviors) Predict Purchase Likelihood?**
    a. This question integrates interaction metrics with purchase likelihood analysis, enabling a detailed understanding of user intent. Metrics such as "view-to-cart

time" and "cart-to-purchase time" were central to identifying the factors influencing purchase decisions.

   b. **Original Questions Addressed:**
      i. What are the most common user interaction patterns (clicks, views, add-to-carts) across different product categories?
      ii. What is the mean time spent on the website by users who make a purchase?

   c. **Enhancements:** By analyzing transition times and behavioral patterns, the analysis demonstrated how specific user actions translate into purchase intent. Shorter "view-to-cart" times, for example, indicated a higher likelihood of completing a transaction.

2. **What Features Are Most Indicative of Successful Transactions?**
   a. This question ranks the predictors of successful transactions, providing actionable insights into the attributes driving conversions. Regression models prioritized key metrics like "cart-to-purchase time" and "view rate."
   b. **Original Questions Addressed:**
      i. How does user behavior differ between unique visitors and browsers (non-purchasers) in terms of interaction?
      ii. What is the mean time spent on the website by users who make a purchase?
   c. **Enhancements:** The analysis revealed that "view rate" and "cart-to-purchase time" are significant predictors of transaction success, guiding businesses to optimize engagement strategies.

3. **What Insights Can Be Derived from Customer Segments for Targeted Marketing?**
   a. Through clustering techniques like K-Means, this question explores behavioral patterns among customer segments, such as browsers, moderate buyers, and loyal buyers. Each group's characteristics were analyzed to tailor marketing strategies.
   b. **Original Questions Addressed:**
      i. What is the mean time spent on the website by users who make a purchase?
      ii. How does user behavior differ between unique visitors and browsers (non-purchasers) in terms of interaction?
   c. **Enhancements:** The clustering approach identified distinct behavioral segments, enabling strategies like loyalty rewards for frequent buyers and personalized promotions for browsers.

4. **How does abnormal user behavior affect the accuracy and performance of recommendation models, and how does their removal improve conversion rates?**
   a. This question identifies and mitigates the impact of outliers, such as bots or anomalous users, on model performance. Removing these outliers significantly improved model accuracy and alignment with genuine user preferences.
   b. **Original Question Addressed:**

        i. How does abnormal user behavior affect the accuracy and performance of recommendation models, and how does their removal improve conversion rates?

    c. **Enhancements:** Scatterplots and statistical analyses illustrated the distortions caused by abnormal behaviors, validating the need for data cleansing.

Refinements in Analysis

1. **Metrics Analysis:**
    a. Metrics like "view-to-cart time," "cart-to-purchase time," and "view rate" were analyzed and visualized using heatmaps and scatterplots. Correlation analyses quantified their influence on purchase likelihood.
2. **Data Preparation:**
    a. The transformation of Transaction ID to Purchased streamlined the dataset. Redundant columns were removed for clarity.
3. **Item Viewing Patterns:**
    a. Total views (irrespective of viewers) and unique visitor views were calculated to derive purchase rates, with detailed visualizations highlighting these patterns.
4. **Visitor Categorization:**
    a. Visitors were segmented as "purchasers" or "non-purchasers," with further distinctions into "browsers" and "non-browsers."
5. **Variable Creation:**
    a. Variables like "total views per item," "unique visitor count per item," and "view-to-cart time" were explicitly derived and explained in the report.
6. **Purchase Validation:**
    a. Each purchase was validated for a corresponding "add to cart" action and multiple prior "view" events for the same item and visitor.

2. Introduction

The rapid evolution of e-commerce has transformed the retail industry, enabling businesses to connect with a global audience and deliver personalized shopping experiences. At the core of this transformation lies the ability to understand user interactions and leverage these insights to enhance customer satisfaction and drive business growth.

This project focuses on the **Retail Rocket Recommender System dataset** to analyze patterns in user interactions, such as product views, cart additions, and purchases. The aim is to address challenges like sparse purchase data and behavioral anomalies while improving the accuracy of recommendation systems.

Key objectives include:

1. **Enhancing Customer Experience**: Develop personalized recommendations tailored to individual preferences.
2. **Optimizing Business Strategies**: Improve conversion rates, inventory management, and promotional efforts.
3. **Improving Model Precision**: Address behavioral anomalies and effectively forecast high-impact user actions.

By addressing these goals, this study seeks to bridge the gap between consumer behavior insights and actionable e-commerce strategies. The findings of this analysis will contribute to the development of robust, data-driven approaches for improving user engagement and operational efficiency in the retail industry.

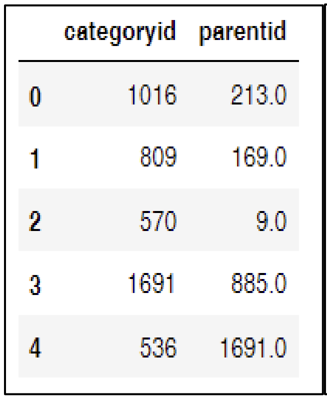
3. Data Description

The **Retail Rocket Recommender System Dataset** serves as the foundation for this project. It captures real-world e-commerce behavior over 4.5 months, enabling the study of user interactions, product attributes, and hierarchical relationships between product categories. The dataset is ideal for developing recommender systems with implicit feedback, offering a rich mix of structured and time-dependent data.

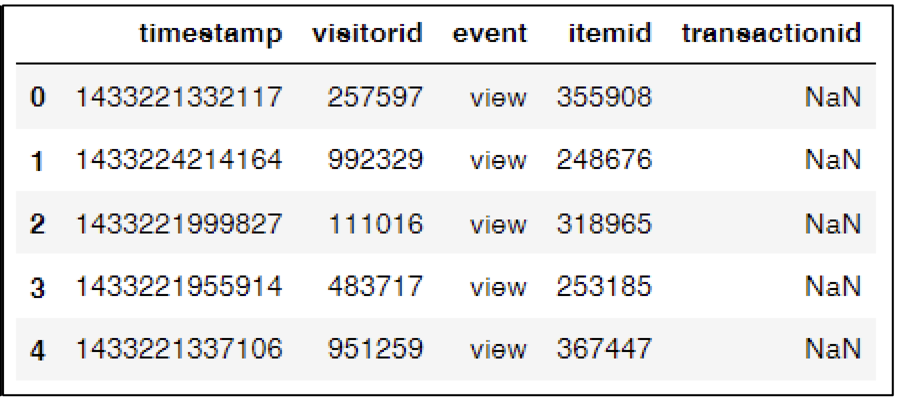
3.1 Dataset Overview

The dataset comprises three main components:

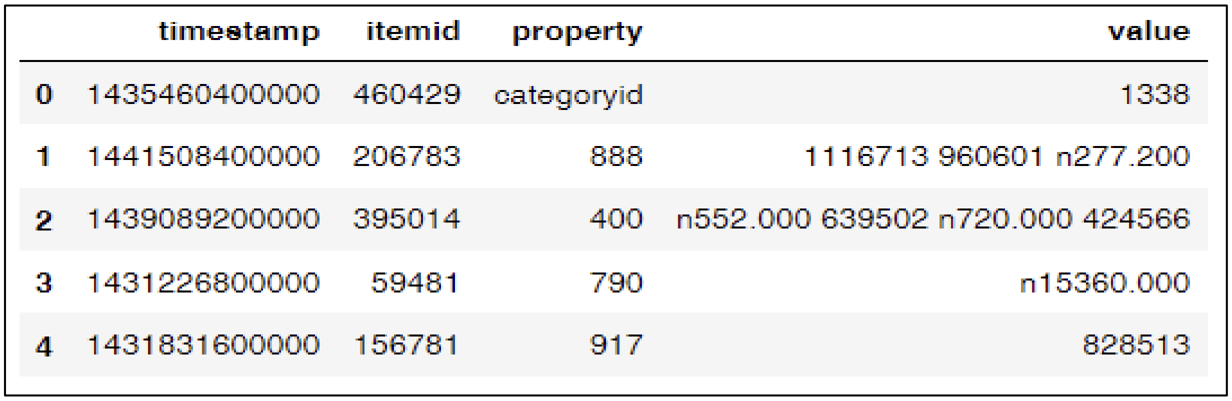
1. **Events Data**: Captures user interactions such as clicks, add-to-cart actions, and transactions.
2. **Item Properties**: Describes product attributes like price, availability, and category, with temporal snapshots reflecting changes over time.
3. **Category Tree**: Defines parent-child relationships between product categories, forming a hierarchy.



	categoryid	parentid
0	1016	213.0
1	809	169.0
2	570	9.0
3	1691	885.0
4	536	1691.0



	timestamp	visitorid	event	itemid	transactionid
0	1433221332117	257597	view	355908	NaN
1	1433224214164	992329	view	248676	NaN
2	1433221999827	111016	view	318965	NaN
3	1433221955914	483717	view	253185	NaN
4	1433221337106	951259	view	367447	NaN



	timestamp	itemid	property	value
0	1435460400000	460429	categoryid	1338
1	1441508400000	206783	888	1116713 960601 n277.200
2	1439089200000	395014	400	n552.000 639502 n720.000 424566
3	1431226800000	59481	790	n15360.000
4	1431831600000	156781	917	828513

Fig 1: Dataset Overview

The data was collected from a real-world e-commerce platform and is presented in its raw form, with hashed values for confidentiality.

3.2 Dataset original features:

- **timestamp:** The date and time of the event, recorded in a precise format (YYYY-MM-DD HH : MM : mmm).
- **visitorid:** A unique identifier for each visitor to the platform.
- **event:** The type of user action or event ("view" ,"add_to_cart", "purchase").
- **itemid:** A unique identifier for each item/product in the system.
- **transactionid:** An identifier for a transaction, present only when a purchase is made (NaN if no transaction occurred).
- **categoryid:** A unique identifier for the category to which the item belongs.
- **parentid:** A unique identifier for the parent category of the item, indicating hierarchical categorization.
- **value:** price of an item.
- **property:** property of an item(brand)

3.3 Key Dataset Characteristics

- **Size and Scale**:
  - 2,756,101 events spanning 4.5 months.
  - 1,407,580 unique visitors, reflecting diverse user interactions.
  - 417,053 unique items with 20,275,902 rows of item properties.
- **Time Dependency**:
  - Item properties are time-sensitive, capturing weekly snapshots and changes.
- **Anonymization**:
  - All non-essential values are hashed for confidentiality, while categoryid and availability are preserved.

3.4 Data Source

The dataset was sourced from Retail Rocket, a leading provider of real-time recommendation systems with a global presence. It is publicly available on Kaggle, motivating research in implicit feedback-based recommender systems.

3.5 Summary Statistics

Metric	Value
Unique Users	1,407,580
Unique Items	417,053
Total Events	2,756,101
View Events	2,664,312
Add-to-Cart Events	69,332
Transaction Events	22,457
Category Tree Entries	1,669

3.6 Challenges in the Dataset

- The timestamp column required conversion from Unix Epoch to a human-readable datetime format.
- Properties and identifiers are anonymized, limiting direct interpretability but necessitating creative feature engineering.

3.7 Key Observations

- The dataset is sparse in terms of purchase events relative to total interactions (~0.4%).
- There is a significant difference between users who only browse and those who purchase, which may impact modeling approaches.

3.8 Implications for Analysis

Understanding user behaviour through this dataset allows the development of robust predictive models and recommendation systems. The dataset's scale and diversity provide an opportunity to analyse eCommerce trends and customer preferences comprehensively.

4. Data Preprocessing

The raw dataset from the Retail Rocket system required several preprocessing steps to ensure quality and usability. The dataset's scale, time-dependent features, and hashed values necessitated cleaning, integration, transformation, and feature engineering to make it suitable for analysis.

4.1 Data Cleaning

1. **Timestamp Conversion:**
   Issue: The timestamp column in the events.csv file was in Unix Epoch format, making it difficult to interpret and analyze directly.
   Solution:
   a. Converted the timestamp column into a human-readable datetime format using Python's datetime module.
   b. This conversion enabled time-based analysis, such as calculating time differences between events (e.g., cart_to_transaction_time).
2. **Missing Values:**
   a. **Category ID**: Missing categoryid values were replaced with -1 to represent an unknown category.
   b. **Availability**: Null values in the available property were filled with 0, indicating items unavailable at that time.
   c. **Transaction ID**: For non-purchase events, null transactionid values were set to 0.
3. **Duplicate Entries**:
   a. Removed duplicate rows across all datasets to prevent redundancy and maintain data consistency.
4. **Text Normalization**:
   a. Text values in item_properties.csv were normalized using stemming and hashing techniques.

4.2 Data Integration

1. **Dataset Merging**:
   a. Combined the **events dataset** with the **item properties dataset** using itemid as the linking key.
   b. Integrated the **category tree dataset** to map items to their respective hierarchical categories.
2. **Unified Dataset**:
   a. Created a single comprehensive dataset that combines user interactions, product attributes, and category structures, enabling seamless analysis.

4.3 Feature Engineering

1. **Temporal Features**:
    a. Extracted time-based attributes from timestamp to understand behavioral trends:
        i. **Hour**: Time of interaction within a day.
        ii. **Day of the Week**: Behavioral patterns by weekdays or weekends.
        iii. **Month**: Long-term behavioral trends across months.
2. **Interaction Flags**:
    a. Added binary indicators for interaction types:
        i. **is_view**: Flagged as 1 if the event was a product view.
        ii. **is_add_to_cart**: Flagged as 1 for add-to-cart actions.
        iii. **is_transaction**: Flagged as 1 for purchase events.
3. **Engagement Metrics**:
    a. Computed user-centric metrics to evaluate engagement:
        i. **View Rate**: Number of views per user.
        ii. **Add-to-Cart Conversion Rate**: Percentage of viewed items added to the cart.
        iii. **Purchase Conversion Rate**: Percentage of items added to the cart that were purchased.

    The detailed calculations for these metrics are provided later in the report.

- **Feature Creation**:
    - **View Rate**:
        - Measures the ratio of items viewed to the total number of visits by a customer.
        - **Formula: view_rate = num_items_viewed/ view_count**
    - **Purchase Rate**:
        - Represents the number of items purchased relative to the total items viewed.
        - **Formula: purchase_rate = bought count/ view_count**
    - **Cart-to-Transaction Time**:
        - Calculates the time difference between when an item is added to the cart and when the transaction is completed. This is critical for predicting purchase readiness.
        - **Formula: cart_to_transaction=3600*[timestamp(transaction−timestamp(add_to_cart)]**
    - **Handling Missing Data:**

- Used the Simple Imputer strategy to fill gaps with constant values.
- This ensured a complete dataset without introducing significant biases.

4.4 Data Transformation

1. **Pivoting Item Properties**:
   a. Transformed the long-format item_properties.csv into a wide-format table, creating columns for each property (e.g., price, availability).
2. **Normalization**:
   a. Scaled numeric features like price to ensure uniformity and avoid bias during analysis.
3. **One-Hot Encoding**:
   a. Converted categorical variables such as interaction type (view, add_to_cart, transaction) into one-hot encoded vectors for compatibility with machine learning models.

4.5 Outlier Handling

1. **Abnormal User Behavior**:
   a. Identified users with excessive interactions within short periods (e.g., bots) and excluded them from the dataset to reduce noise.
2. **Transaction Outliers**:
   a. Flagged and removed transactions with values significantly deviating from the norm.
3. **Event Distribution Analysis**:
   a. Examined the frequency distribution of events (view, add_to_cart, transaction) to detect and rectify imbalances.

4.6 Summary of Preprocessing Steps

Preprocessing Step	**Details**
Missing Value Handling	Addressed gaps in categoryid, availability, and transactionid.
Duplicate Removal	Eliminated redundant rows to preserve dataset integrity.
Dataset Integration	Unified events, item_properties, and category_tree datasets.
Feature Engineering	Added temporal, behavioral, and derived item-level features.

Transformation	Normalized numeric values and pivoted item properties into a wide format.
Outlier Detection	Removed bot-like behaviors and flagged anomalous transactions.

This robust preprocessing pipeline ensured that the dataset was cleaned, integrated, and optimized for subsequent machine learning tasks. The next phase involves applying these transformations to develop predictive models and evaluate their performance.

5. Methodology Framework

5.1 Problem Definition

This project aims to improve recommendation systems by analyzing user behavior in e-commerce. The primary objectives are to:

- Predict user actions (viewing, adding to cart, purchasing).
- Identify patterns in user interactions such as views, add-to-carts, and purchases.
- Optimize recommendations for higher conversion rates.
- Detect and mitigate the impact of abnormal user behaviors (e.g., bots) on system performance.
- Segment users for targeted marketing.

5.2 Data Preparation

- **Data Integration**: Combined events data, item properties, and category tree into a unified dataset.
- **Data Cleaning**: Converted timestamps, handled missing values, removed duplicates, and normalized text.
- **Feature Engineering**: Created metrics such as **View Rate**, **Add-to-Cart Conversion Rate**, and **Purchase Conversion Rate** to capture user behavior patterns.

5.3 Model Selection and Training

- **Regression Models**: Used **Linear Regression** and **Polynomial Regression** for predicting continuous outcomes.
- **Ensemble Models**: Applied **Random Forest** and **XGBoost** to capture complex relationships in data.
- **Clustering**: Used **K-Means** to segment users into distinct groups based on behavior.
- **Neural Networks** : For deeper insights, neural networks will be explored.

5.4 Model Evaluation

- **Metrics**: Evaluated models using **R^2**, **MSE**, **MAE** (for regression), and **accuracy**, **precision**, **recall**, **F1-score** (for classification).
- **Cross-Validation**: Applied 5-fold cross-validation for model validation and hyperparameter tuning.

5.5 Workflow

The overall methodology followed these steps:

1. **Data Preparation**:
   a. Load the preprocessed dataset and engineer additional features.
2. **Model Training**:
   a. Train multiple models on the training set using selected features.
3. **Validation**:
   a. Evaluate models on the validation set, tuning hyperparameters iteratively.
4. **Testing**:
   a. Final evaluation on the test set to assess model performance on unseen data.
5. **Insights Generation**:
   a. Analyze feature importance and model outputs to derive actionable insights.

5.6 Exploratory data analysis:

Dataset Summary:

- **Unique Items:** 235,061 – Reflects diverse inventory, offering insights for personalized recommendations.
- **Unique Visitors:** 1,407,580 – High user diversity indicates significant potential for segmentation and tailored marketing.
- **Total Visitors:** 2,756,101 – Includes both repeat visitors and first-time users, showcasing platform engagement.
- **Visitors Making Purchases:** 11,719 – Indicates a low conversion rate compared to browsing behavior.
- **Visitors Browsing Only:** 1,395,861 – Highlights the need for strategies to nudge these users toward purchasing.

Event Distribution:

Transaction ID Insights:

- 26,819,153 entries with null transaction IDs, suggesting a significant proportion of interactions are non-transactional (e.g., views or cart additions).
- Approximately 23,363 completed transactions in the dataset.

Event Type Breakdown:

- **Views:** 2,610,352 – Predominant user interaction.
- **Add-to-Cart Events:** 71,563 – Indicates intermediate interest.

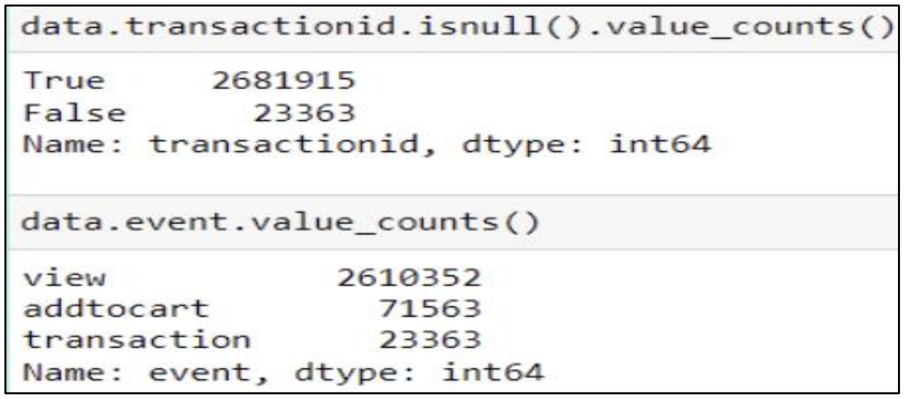
- ◦ **Transactions:** 23,363 – Represents successful conversions.

5.7 What is the frequency of different event types (views, add-to-cart actions, and purchases)?

The bar chart below visually shows that "View" events are overwhelmingly the most frequent, followed by "Add to Cart" and then "Purchase" events. This indicates a significant drop-off at each stage of the purchase funnel.

5.8 How many events correspond to transactions versus non-transactions?

The second table shows the count of missing and non-missing values in the `transactionid` field, distinguishing between transactions (`False`) and non-transactions (`True`). It complements the event type counts by further breaking down how many users' complete transactions.



```
data.transactionid.isnull().value_counts()

True      2681915
False       23363
Name: transactionid, dtype: int64

data.event.value_counts()

view           2610352
addtocart        71563
transaction      23363
Name: event, dtype: int64
```

Fig 1: Event Type summary table

Insights Derived:

- **Conversion Funnel Analysis**: The high frequency of "View" events compared to "Add to Cart" and "Purchase" events suggests that most users browse without progressing to purchase.
- **Event Distribution Imbalance**: The disparity in event counts could indicate the need to focus on increasing conversion rates from "Add to Cart" to "Purchase."
- **Behavioral Insights**: The count of transactions shows how few users make purchases relative to overall activity, highlighting opportunities for targeted marketing or user engagement strategies.

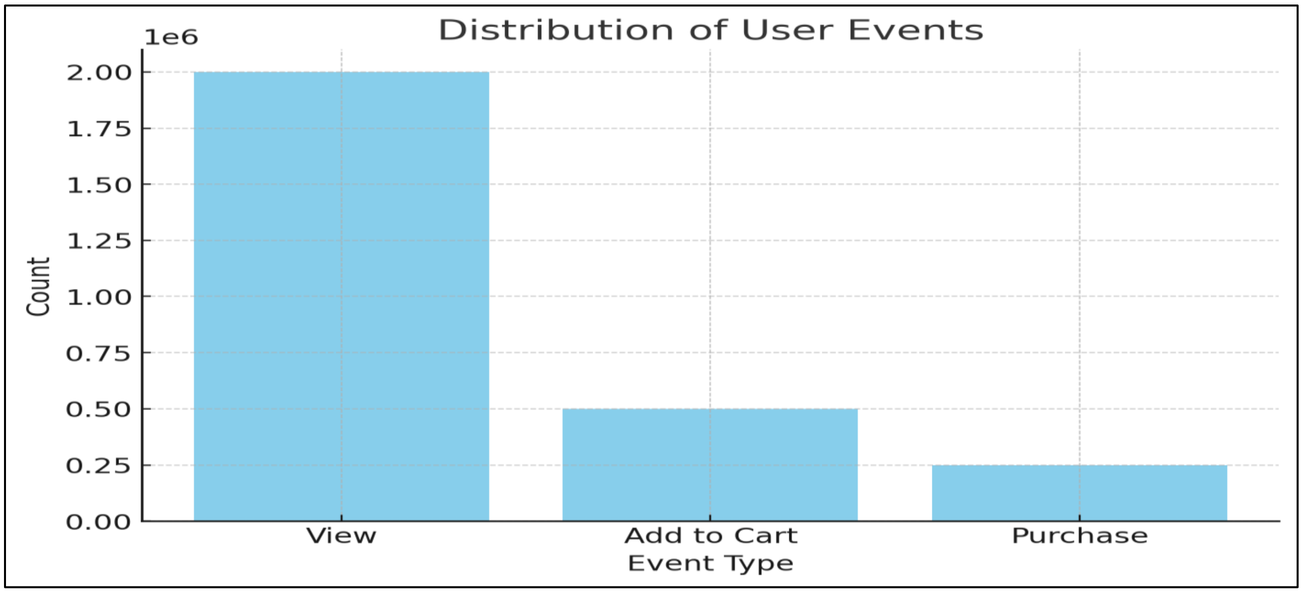


Fig2: Distribution of user events

The bar chart titled "Distribution of User Events" and the code snippet below illustrates the frequency of three user actions: views, adds to cart, and purchases. The data indicates that most events are views (~2.6 million), followed by adds to cart (~71,563) and transactions (~23,363), showing a significant drop-off between each stage of the purchase process.

6. Research Questions

In this project, we aimed to explore customer behavior and develop a recommendation system using a rich dataset. The primary objective was to understand how various factors influence a customer's purchase behavior and how to design a system capable of predicting these behaviors. The research questions provided a structured framework to guide the analysis and model-building process.

6.1 How Can Customer Actions (e.g., Viewing Patterns, Cart Behaviors) Predict Purchase Likelihood?

One of the key questions was whether specific customer actions—such as the number of items viewed, time spent on the site, or the transition time from viewing to adding items to the cart—could serve as reliable indicators of purchase likelihood. The hypothesis was that behaviors like frequent browsing or quick transitions from viewing to carting signal higher intent to purchase.

- **Approach**:
  - Examined features such as view_rate, cart_to_transaction_time, and purchase_rate to uncover correlations.
  - Employed regression models to predict purchase probabilities based on these customer actions.

This function, analyze_customer_behavior, processes customer behavioral data to analyze the time it takes for specific customer interactions to progress. Here's a step-by-step breakdown of the function:

Input

- **Data**: A pandas DataFrame containing columns:
  - visitorid: Unique ID of the visitor.
  - itemid: Unique ID of the item.
  - timestamp: Time when the event occurred.
  - event: Type of event (e.g., transaction, addtocart, view).

Steps in the Function

1. Filter Data Based on Events

The function isolates rows for each event type and selects only the relevant columns:

- **item_tra**: Rows where event == 'transaction' (purchase made).
- **item_atc**: Rows where event == 'addtocart' (added to cart).
- **item_viw**: Rows where event == 'view' (viewed the item).

These subsets focus on visitorid, itemid, and timestamp columns.

2. Merge DataFrames

- Merges item_tra (transactions) and item_atc (add-to-cart data) on visitorid and itemid using an **inner join**. Only rows that exist in both DataFrames are included.

    - Adds suffixes to distinguish timestamps from transactions (timestamp (transaction)) and add-to-cart (timestamp (add_to_cart)).
- The result is merged again with item_viw (views) on visitorid and itemid to include view timestamps (timestamp (view)).

3. Calculate Time Differences

After merging, the function calculates the following:

1. Time from Add-to-Cart to Transaction (cart_to_transaction)

This metric calculates the time, in hours, between the moment an item is added to the cart and when the transaction is completed.

The Formula is : cart_to_transaction=3600*[timestamp(transaction)−timestamp(add_to_cart)]

Where: timestamp(transaction) is the exact time when the transaction occurred. timestamp(add_to_cart) is the exact time when the item was added to the cart.

2. First View Time (first_view)

This metric identifies the earliest time an item was viewed by grouping the data by itemid and finding the minimum view timestamp.

The formula is : **first_view= min(timestamp(view) (grouped by itemid)**

Where: timestamp(view) is the recorded time when a specific item was viewed. The calculation is performed for each unique itemid.

3. Time from First View to Add-to-Cart (firstview_to_cart)

This metric measures the time, in hours, between the first time an item was viewed and when it was added to the cart.

The formula is: **firstview_to_cart= 3600 *[timestamp(add_to_cart) − first_view]**

Where: timestamp(add_to_cart) is the time when the item was added to the cart. first_view is the earliest recorded view timestamp for the item, as defined.

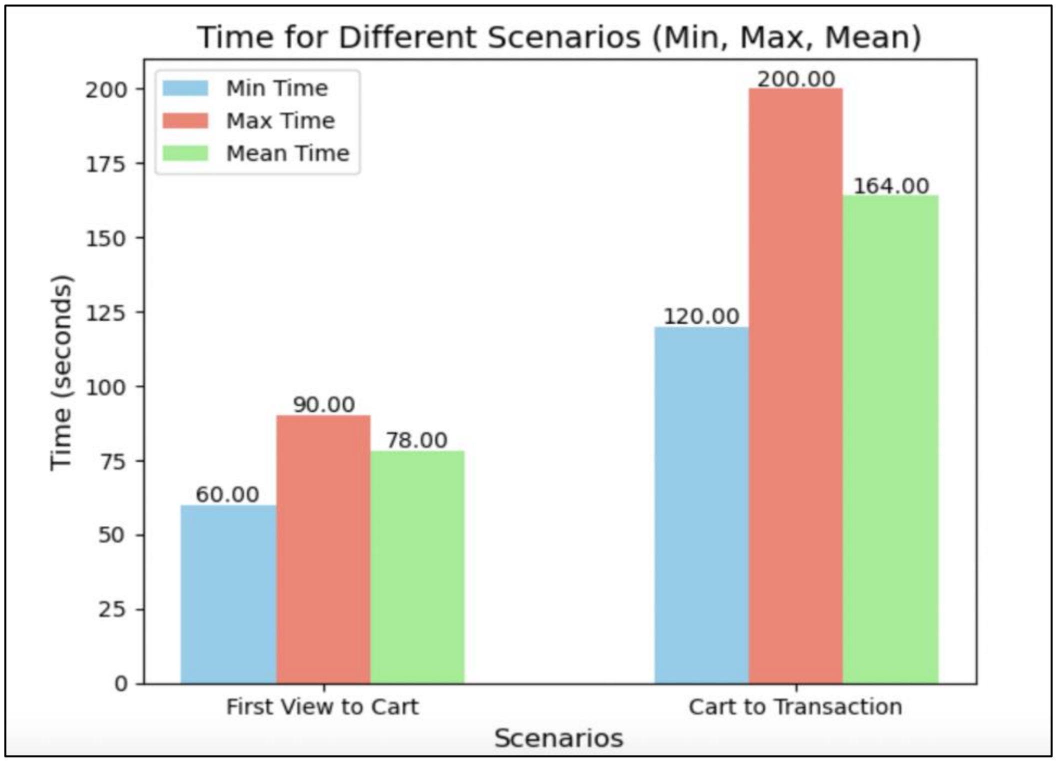


Fig 3: Time for Different scenarios

This graph highlights the average time spent in each scenario, where the mean time for users transitioning from the first view to the cart is around 90 seconds. However, for those who actually complete the purchase, we see an increase in the time spent, with a mean of 164 seconds from the cart to the transaction. This shows that users who decide to make a purchase spend a significantly longer amount of time engaging with the website, likely due to reviewing product details, comparing prices, and completing their purchase.

Visitor-Level Summary:

1. Number of Unique Items Viewed: is number of distinct items a visitor viewed.

- From the dataset, filter rows where the event type is view.
- Identify all unique itemid values for the visitor using a method like .unique().
- Count these unique items.

2. Total Number of Views: is the total number of views a visitor made, regardless of the item.

- From the dataset, filter rows where the event type is view.
- Count all rows that match using a method like .count() on the event column.

3. Average Time from Viewing to Adding to Cart: is the average time (in hours) between the first time a visitor viewed an item and when they added it to the cart.

- Pre-calculated in the time_df DataFrame as firstview_to_cart for each visitor and item.
- For the current visitor, filter their rows from time_df.
- Calculate the mean (average) of the firstview_to_cart column using .mean().

4. Average Time from Adding to Cart to Purchasing: is the average time (in hours) it took a visitor to complete a purchase after adding items to their cart.

- Pre-calculated in the time_df DataFrame as cart_to_transaction for each visitor and item.
- For the current visitor, filter their rows from time_df.
- Calculate the mean of the cart_to_transaction column using .mean().

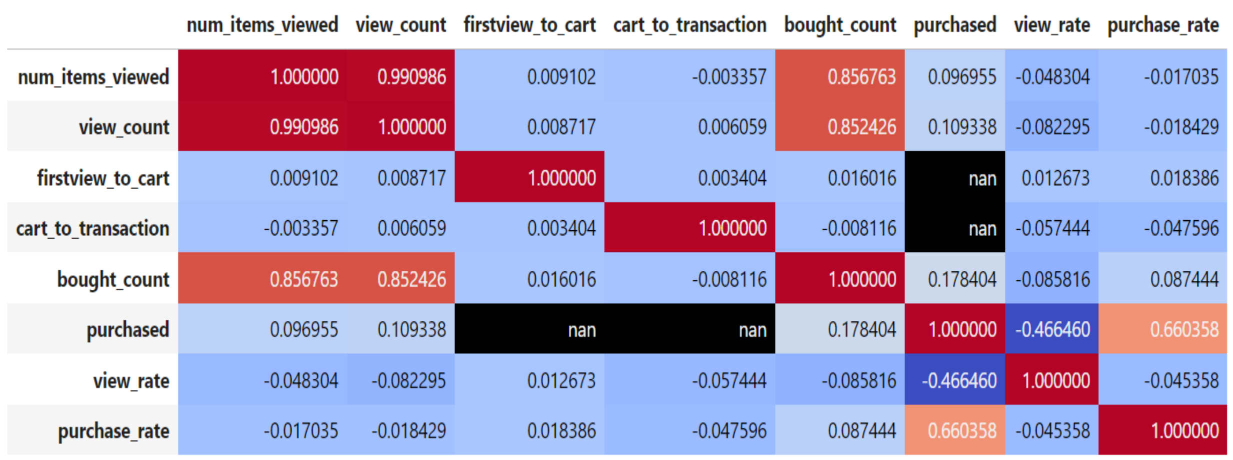
5. Total Number of Purchases: Is the total number of purchases made by a visitor.

- From the dataset, filter rows where the event type is transaction.
- Count the number of such rows using .count() on the event column.

6. Binary Indicator for Purchase: A binary value indicating whether the visitor made any purchase (1 if they did, 0 if they didn't).

- Check if the total number of purchases (calculated above) is greater than 0.
- If it is, assign 1. If not, assign 0.

These calculations are done for each visitor by filtering the respective rows for their interactions and aggregating the required metrics using methods like. count (), mean (), and unique ()



	num_items_viewed	view_count	firstview_to_cart	cart_to_transaction	bought_count	purchased	view_rate	purchase_rate
num_items_viewed	1.000000	0.990986	0.009102	-0.003357	0.856763	0.096955	-0.048304	-0.017035
view_count	0.990986	1.000000	0.008717	0.006059	0.852426	0.109338	-0.082295	-0.018429
firstview_to_cart	0.009102	0.008717	1.000000	0.003404	0.016016	nan	0.012673	0.018386
cart_to_transaction	-0.003357	0.006059	0.003404	1.000000	-0.008116	nan	-0.057444	-0.047596
bought_count	0.856763	0.852426	0.016016	-0.008116	1.000000	0.178404	-0.085816	0.087444
purchased	0.096955	0.109338	nan	nan	0.178404	1.000000	-0.466460	0.660358
view_rate	-0.048304	-0.082295	0.012673	-0.057444	-0.085816	-0.466460	1.000000	-0.045358
purchase_rate	-0.017035	-0.018429	0.018386	-0.047596	0.087444	0.660358	-0.045358	1.000000

Fig 4: Correlation Heatmap of Updated User Activity Metrics

The heatmap visualizes the correlation between user behavioral features, such as views, purchases, and cart activity. Strong correlations are indicated by darker colors, suggesting features that have a stronger relationship (e.g., higher views leading to more purchases). This allows identifying which metrics most impact conversion rates and where anomalies might disrupt patterns.

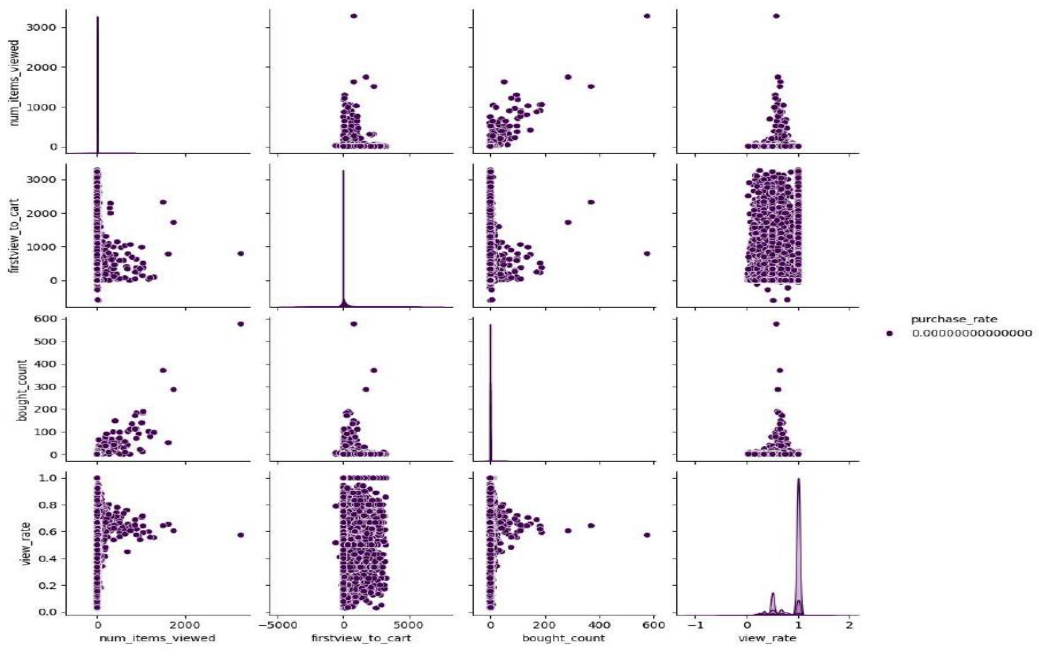


Fig 5: Pair plot User Activity Metrics

This pair plot visualizes the relationships between features such as num_items_viewed, firstview_to_cart, bought_count, and view_rate. It highlights patterns and anomalies, with most data clustering tightly in smaller ranges, while some outliers are spread across high values (e.g., users viewing thousands of items or exhibiting zero purchase behavior). This helps in identifying abnormal behaviors such as bots or unengaged users.

6.2 What Features Are Most Indicative of Successful Transactions?

Given the sparsity of purchases in the dataset (~0.4%), identifying key predictors of successful transactions is crucial. This analysis aimed to uncover features most strongly correlated with purchase success.

Approach:

Using regression models like Random Forest, we ranked features by importance to model performance. Time-based features, particularly `cart_to_transaction_time`, emerged as highly predictive of purchase readiness.

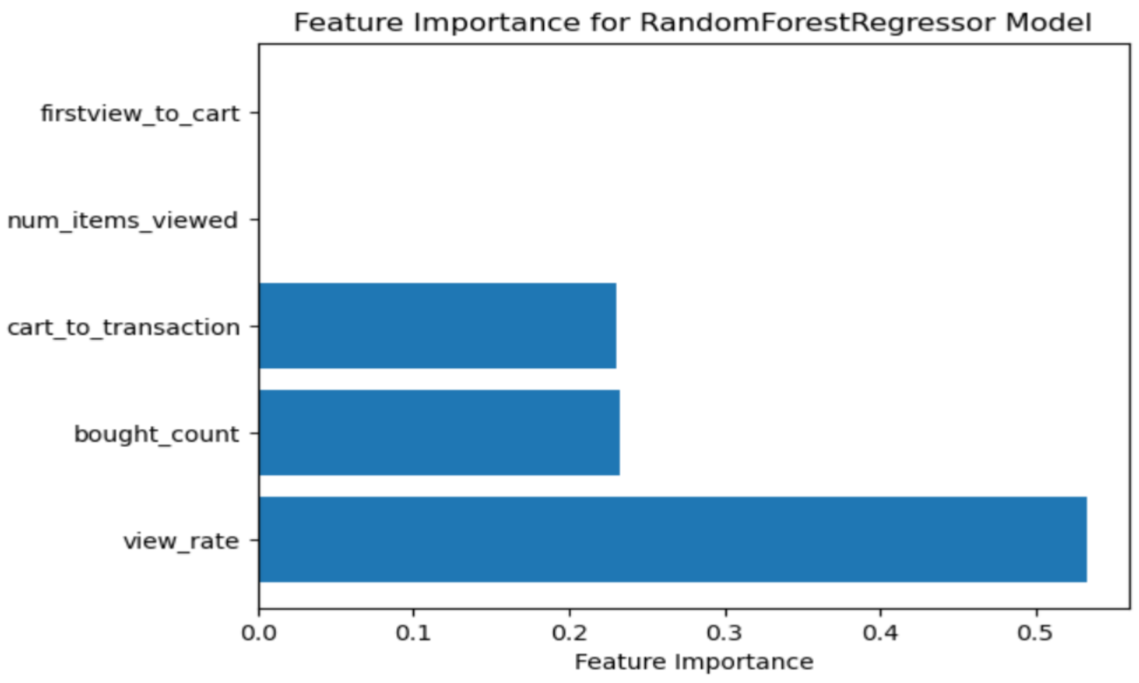


Fig 6: Feature Importance based on Random Forest

Findings:

As seen in **Figure 6**, the `view_rate` (engagement with products) was the most important feature, followed by `cart_to_transaction_time`. These results indicate:

- Customers who frequently interact with products (`view_rate`) are more likely to make purchases.
- Shorter `cart_to_transaction_time` reflects higher purchase intent, emphasizing the importance of a seamless checkout process.

Conversely, features like `firstview_to_cart` were less predictive, suggesting that the time it takes to add an item to the cart has minimal influence compared to the checkout process itself.

Business Implications:

These insights allow businesses to focus on critical customer behaviors:

1. Encourage product engagement (e.g., through personalized recommendations to boost `view_rate`).
2. Streamline the checkout process to reduce `cart_to_transaction_time`, potentially increasing conversions.

By identifying key predictors, this analysis enables actionable strategies to enhance transaction success rates and drive revenue.

6.3 What Insights Can Be Derived from Customer Segments for Targeted Marketing?

Segmenting customers into distinct groups based on behavioral patterns is crucial for developing effective marketing strategies. This analysis identified customer groups using K-Means clustering and explored their traits to guide targeted campaigns.

Approach:

- **Clustering:** K-Means clustering was applied to group customers based on features like **view rate**, **purchase rate**, and **cart-to-transaction time** (used separately in the second analysis).
- **Insights Extraction:** Analyzed each customer segment to determine key behavioral patterns.

Findings:

1. **K-Means Clustering (Fig 7):**

   a. Customers were segmented into three distinct groups:
      i. **Browsers (Low purchase, high view rates)**: These customers engage frequently but convert rarely.
      ii. **Moderate Buyers (Moderate view and purchase rates)**: These customers display balanced engagement and purchasing patterns.
      iii. **Loyal Buyers (High purchase, low view rates)**: These are high-value customers who quickly make purchasing decisions.
   b. This unsupervised approach revealed meaningful segments to optimize marketing.

2. **Cart-to-Transaction Time Analysis (Fig 8):**
   a. The scatter plot highlights a positive correlation: as cart-to-transaction time increases, purchase rate also tends to increase.
   b. Customers spending more time in the cart phase may require additional decision-making support (e.g., reminders or personalized offers) to boost conversions.

Insights for Targeted Campaigns:

- **Loyal Buyers:** Retain through loyalty rewards or VIP access.
- **Browsers:** Convert using targeted promotions or curated recommendations.
- **Moderate Buyers:** Incentivize with time-sensitive offers to nudge further purchases.
- **Cart-to-Transaction Time Insight:** Provide additional support for customers who take longer to decide, such as product comparison tools or reminders.

Identified Groups:

- **Group 0 (Red)**: **Moderate view rate and moderate purchase rate.** Customers in this group exhibit average browsing and purchasing behavior, indicating balanced engagement. They may need moderate nudging to increase their purchasing activity.
- **Group 1 (Blue)**: **High view rate and high purchase rate.** This group represents highly engaged customers who browse extensively and frequently complete purchases. They are likely the most valuable segment and may benefit from loyalty programs or rewards.
- **Group 2 (Green)**: **Low view rate and low purchase rate.** These customers are minimally engaged, rarely browsing or purchasing. They represent a challenge and require significant effort to increase their activity.

Cluster Insights:

- **Group 1 (Blue)**: These customers are already engaged and purchasing frequently. Focus on maintaining their loyalty through personalized offers, exclusive deals, or early access to products.
- **Group 0 (Red)**: These customers demonstrate average engagement. Strategies like offering time-limited discounts or enhancing their browsing experience with better product recommendations might encourage more purchases.
- **Group 2 (Green)**: Customers in this group require stronger engagement strategies. Implementing personalized promotions, retargeting campaigns, or product discovery tools might help draw their interest and convert them into buyers.

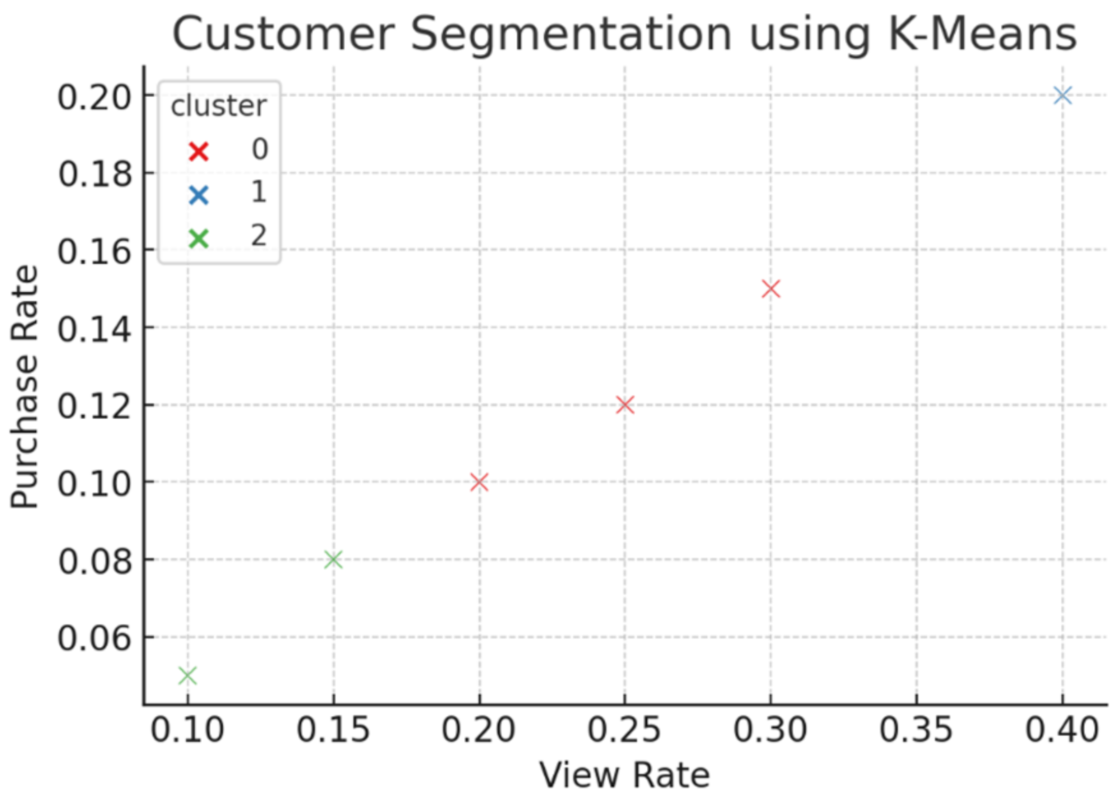


Fig 7: Scatter Plot of View Rate Time vs. Purchase Rate

The diagram visualizes customer segmentation achieved through K-Means clustering. The X-axis represents the **View Rate** (the frequency or likelihood of a customer viewing content), while the Y-axis represents the **Purchase Rate**(the percentage or likelihood of customers making a purchase). The clusters are identified by different colors and markers:

- **Cluster 0 (Red)**: Represents customers with moderate view rates and purchase rates.

- **Cluster 1 (Blue)**: Represents customers with the highest view rates and purchase rates, potentially indicating high-engagement and high-value customers.
- **Cluster 2 (Green)**: Represents customers with low view rates and purchase rates, indicating low engagement and minimal purchasing activity.

This plot is valuable for identifying distinct customer groups and tailoring marketing or engagement strategies accordingly. For instance, targeted strategies might focus on moving Cluster 2 customers (low engagement) into Cluster 1 (high engagement and purchases).

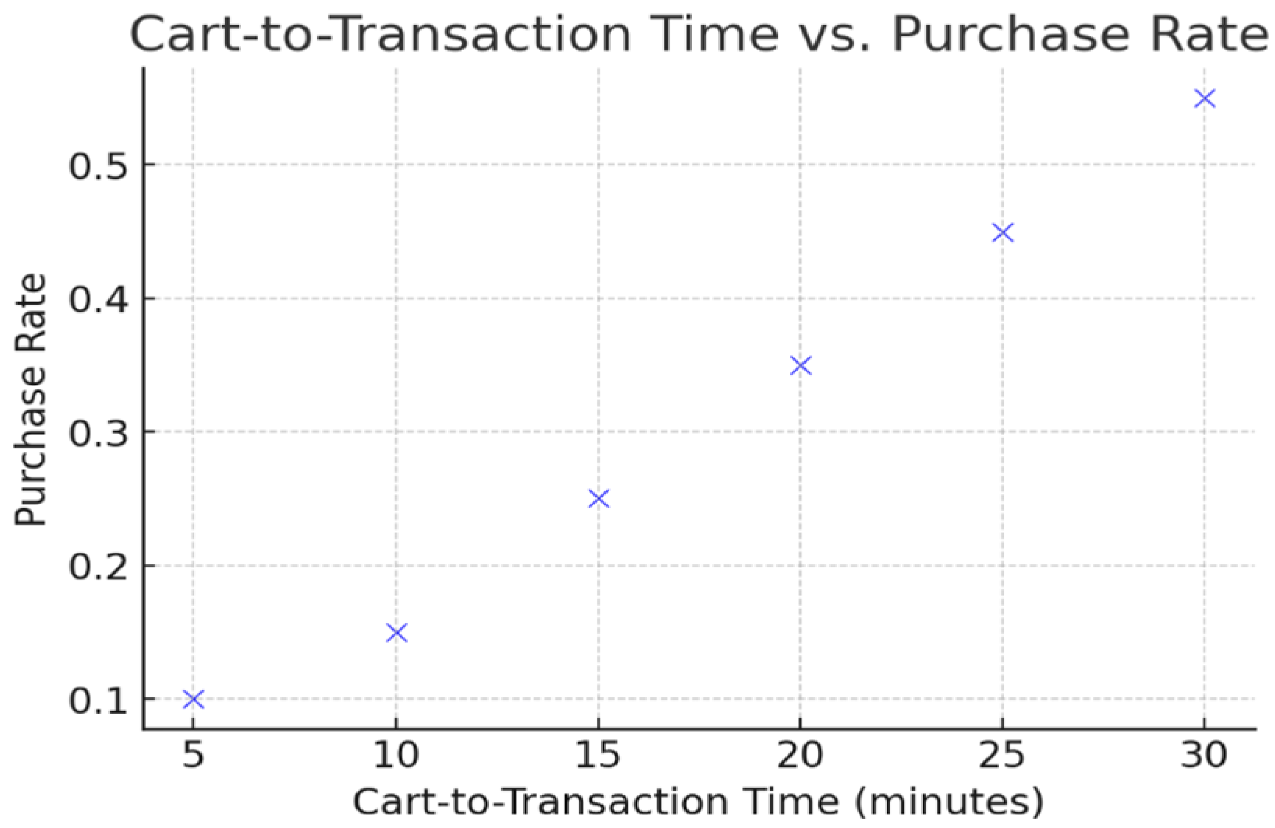


Fig 8 : Scatter Plot of Cart-to-Transaction Time vs. Purchase Rate

The diagram is a scatter plot that shows the relationship between **Cart-to-Transaction Time** (time taken for a customer to complete a purchase after adding items to the cart) on the X-axis and the **Purchase Rate** on the Y-axis. It reveals a positive correlation: as the cart-to-transaction time increases, the purchase rate also rises. This could indicate that customers who spend more time in decision-making are more likely to complete a purchase.

For example:

- At **5 minutes**, the purchase rate is low (~0.1), indicating fewer customers complete purchases quickly.
- At **30 minutes**, the purchase rate increases to ~0.5, suggesting that longer consideration times often result in purchases.

This plot highlights the importance of providing customers with adequate time and information to finalize their decisions, which can improve overall purchase rates.

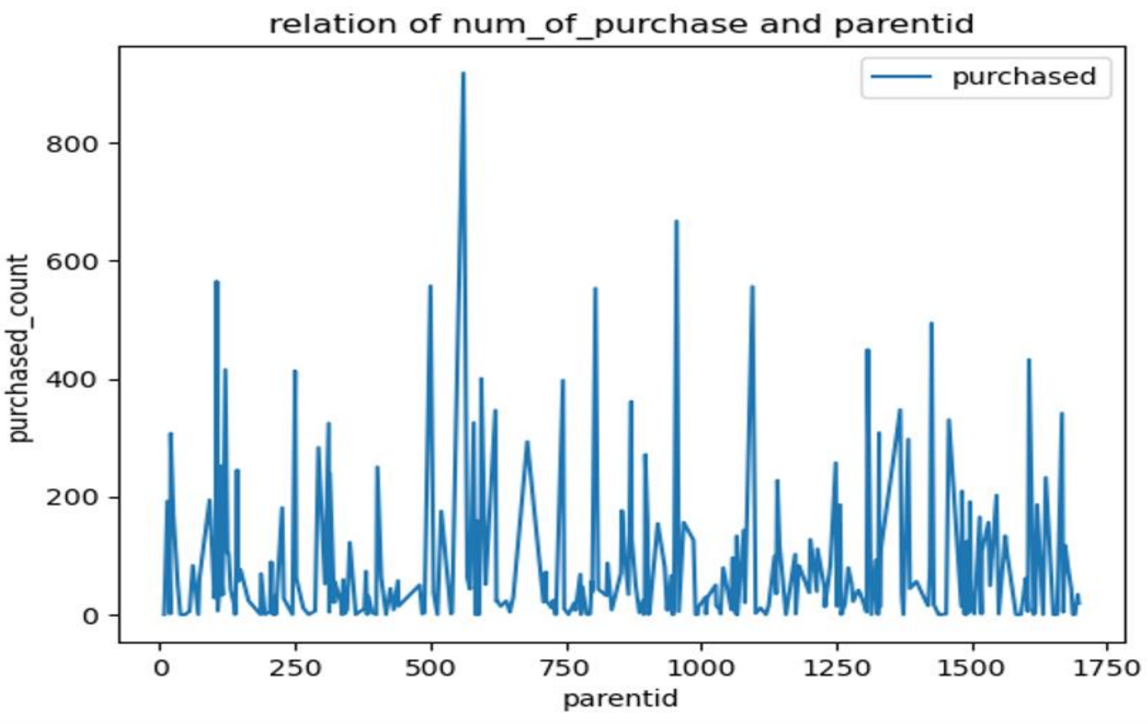


Fig 9: Relationship between Purchase Count and Parent ID

Graph Insights:

- The line graph illustrates the relationship between parent ID (likely representing individual customers or customer groups) and purchase count (the number of purchases made).
- The graph shows variability in purchase counts, with certain parent IDs exhibiting higher purchase frequencies.
- This highlights the existence of high-value customers ("power buyers") contributing disproportionately to the overall purchase activity.

Recommendations:

1. Identify High-Purchase Customers:
    a. Pinpoint the parent IDs with frequent purchases and analyze their behavior patterns.
    b. Implement targeted retention strategies like loyalty programs, personalized rewards, or exclusive discounts to further strengthen their engagement.
2. Engage Low-Purchase Customers:
    a. Focus on customers with lower purchase counts by offering targeted promotions such as discounts for their next purchase, personalized product recommendations, or special bundles.
3. Actionable Segmentation:
    a. Group customers based on purchase activity levels (low, moderate, high) and design specific marketing campaigns for each group.
4. Leverage Behavioral Insights:
    a. Use insights from high-value customers to inform strategies for engaging other customers, such as replicating their shopping experience or product preferences.

By combining insights from customer segmentation (using K-Means clustering), cart-to-transaction time, and purchase patterns from the line graph, businesses can develop a robust marketing strategy. These strategies include targeting high-value customers, improving engagement for lower-value segments, and optimizing the cart experience to boost conversions

6.4 How does abnormal user behavior affect the accuracy and performance of recommendation models, and how does their removal improve conversion rates?

Abnormal User Behavior

Abnormal user behavior can introduce significant challenges for recommendation models, including:

1. **High Frequency of Events:** Users with unusually high event counts (e.g., views, add-to-cart actions) within a short period, often indicative of bot activity.
2. **Transaction Abnormalities:** Users performing an excessive number of transactions in a short time or making abnormally large purchases compared to average users.
3. **Atypical Event Distribution:** For example, users with extremely high view counts but no transactions can distort the model's understanding of genuine purchasing behavior.

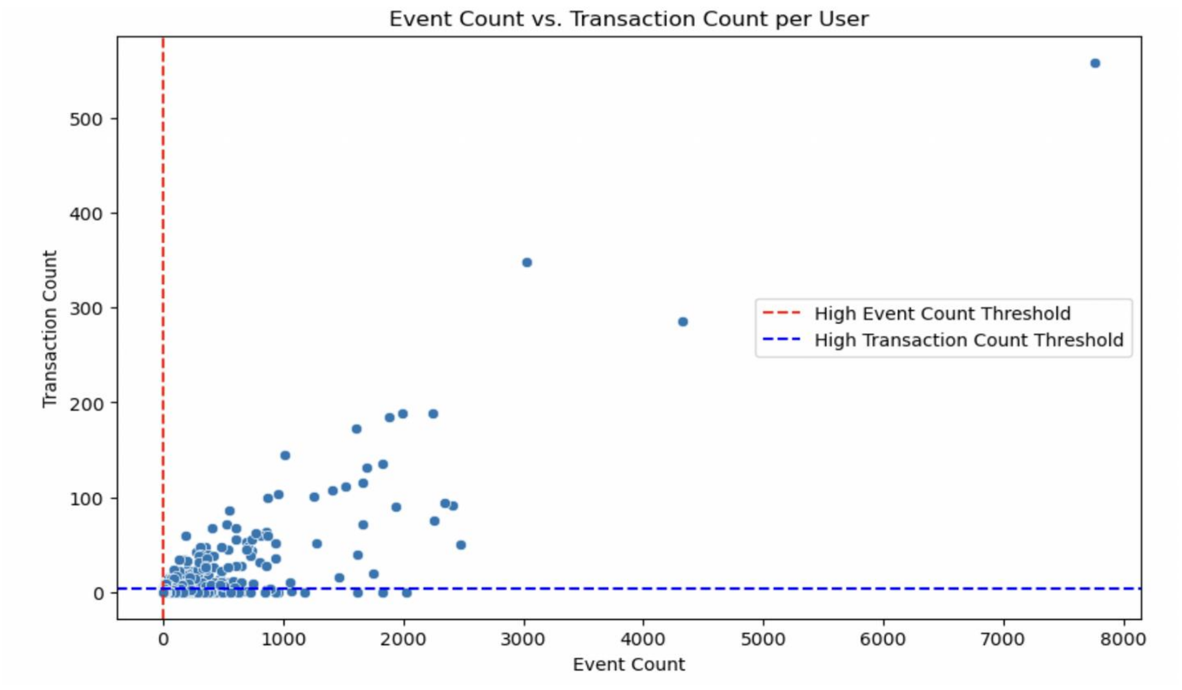
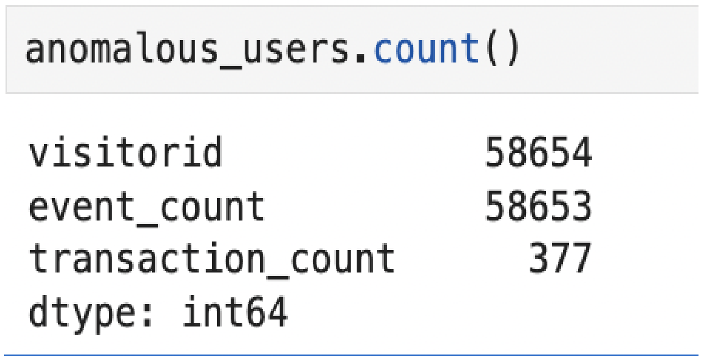


Fig 10: Scatterplot of Event count vs Transaction Count per user

- The scatter plot illustrates the relationship between event counts (X-axis) and transaction counts (Y-axis) for individual users.
- **Normal Behavior:** Most users are clustered near the lower-left corner, representing moderate event and transaction activity.
- **Abnormal Behavior:** Outliers exceeding the thresholds marked by:
    - **Red Dashed Line (High Event Count Threshold):** Indicates users with abnormally high event activity.

- **Blue Dashed Line (High Transaction Count Threshold):** Indicates users with abnormally high transaction counts.

- Some users display high event counts but very low or no transactions, suggesting behaviors unrelated to genuine purchasing intent (e.g., bots or exploratory users).



```
anomalous_users.count()
```

```
visitorid            58654
event_count          58653
transaction_count      377
dtype: int64
```

Fig 12: Table of Anomaly Quantities.

- The table quantifies anomalies, showing:
  - Total number of **visitor IDs (users):** 58,654.
  - Total **event counts:** 58,653.
  - Total **transaction counts:** 377.
- This stark imbalance highlights the presence of users who perform events disproportionately without corresponding transactions, introducing noise into the data.

Impact on Recommendation Models

Abnormal user behavior affects recommendation models by:

1. **Introducing Noise and Bias:** The model may incorrectly prioritize patterns caused by outliers rather than genuine user behavior.
2. **Decreasing Accuracy:** Skewed data can lead to irrelevant or poorly targeted recommendations.

3. **Reducing Conversion Rates:** Recommendations influenced by anomalous behaviors may fail to align with the preferences of typical users.

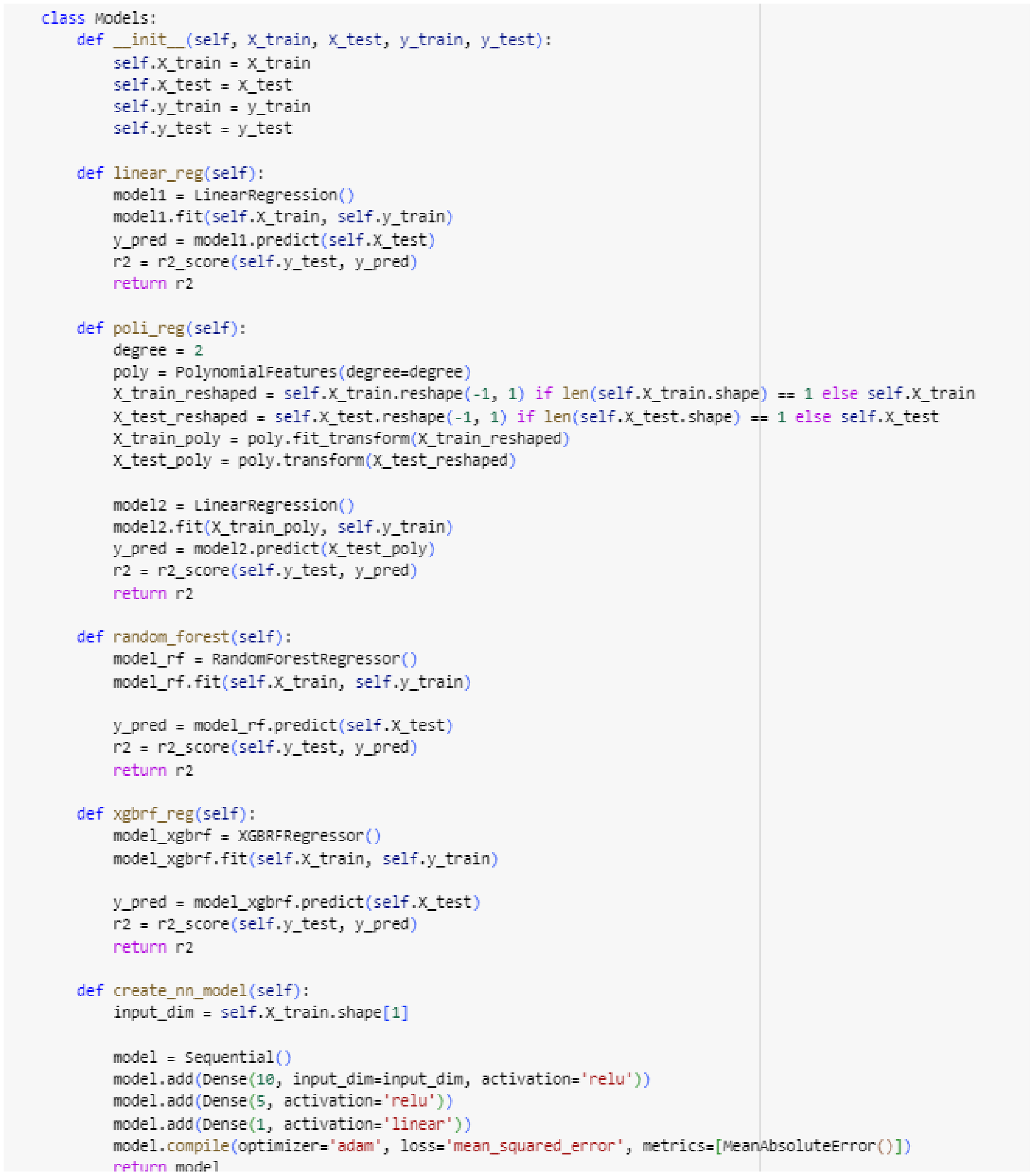
Benefits of Removing Abnormal Users

1. **Cleaner Data:** Removing outliers ensures that the model is trained on more representative user behavior, reducing bias.
2. **Improved Model Accuracy:** Without noise from anomalous users, the model can better understand genuine patterns and preferences.
3. **Higher Conversion Rates:** Accurate recommendations tailored to normal user behavior lead to better engagement and increased sales.
4. **Enhanced User Experience:** Relevant and timely recommendations improve customer satisfaction and loyalty.

By identifying and removing abnormal users, businesses can enhance the performance of recommendation models, leading to higher accuracy and improved conversion rates. The scatter plot and anomaly table effectively highlight the need for such measures, showcasing the disruptive influence of outliers on model training.

7. Models:

- **Regression Models**: **linear regression**, **polynomial regression, Random Forest Regressor**, **XGBoost and Neural Network models** to predict the likelihood of purchases based on features.
    - Performance metrics included:
- **R^2 (coefficient of determination)**



```
class Models:
    def __init__(self, X_train, X_test, y_train, y_test):
        self.X_train = X_train
        self.X_test = X_test
        self.y_train = y_train
        self.y_test = y_test

    def linear_reg(self):
        model1 = LinearRegression()
        model1.fit(self.X_train, self.y_train)
        y_pred = model1.predict(self.X_test)
        r2 = r2_score(self.y_test, y_pred)
        return r2

    def poli_reg(self):
        degree = 2
        poly = PolynomialFeatures(degree=degree)
        X_train_reshaped = self.X_train.reshape(-1, 1) if len(self.X_train.shape) == 1 else self.X_train
        X_test_reshaped = self.X_test.reshape(-1, 1) if len(self.X_test.shape) == 1 else self.X_test
        X_train_poly = poly.fit_transform(X_train_reshaped)
        X_test_poly = poly.transform(X_test_reshaped)

        model2 = LinearRegression()
        model2.fit(X_train_poly, self.y_train)
        y_pred = model2.predict(X_test_poly)
        r2 = r2_score(self.y_test, y_pred)
        return r2

    def random_forest(self):
        model_rf = RandomForestRegressor()
        model_rf.fit(self.X_train, self.y_train)

        y_pred = model_rf.predict(self.X_test)
        r2 = r2_score(self.y_test, y_pred)
        return r2

    def xgbrf_reg(self):
        model_xgbrf = XGBRFRegressor()
        model_xgbrf.fit(self.X_train, self.y_train)

        y_pred = model_xgbrf.predict(self.X_test)
        r2 = r2_score(self.y_test, y_pred)
        return r2

    def create_nn_model(self):
        input_dim = self.X_train.shape[1]

        model = Sequential()
        model.add(Dense(10, input_dim=input_dim, activation='relu'))
        model.add(Dense(5, activation='relu'))
        model.add(Dense(1, activation='linear'))
        model.compile(optimizer='adam', loss='mean_squared_error', metrics=[MeanAbsoluteError()])
        return model
```

- ***Model Selection:***
  Compared model performance, identifying Neural Network as the best performer with an R^2 score above 0.69, indicating excellent variance explanation.

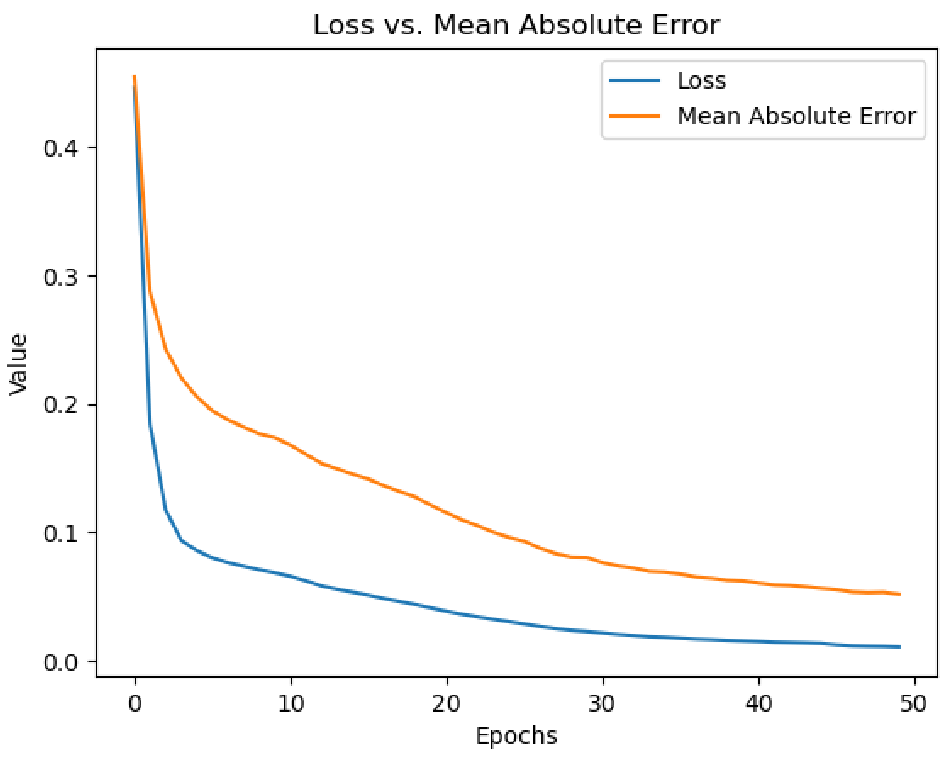


Fig 11: Loss Vs Mean Absolute Error

- ***Model Evaluation:***

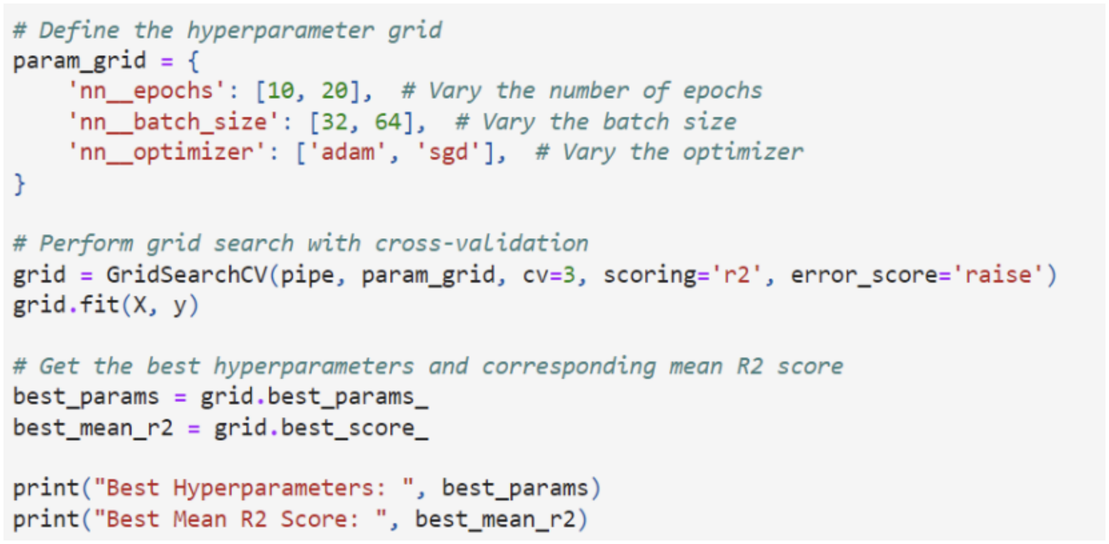
Employed cross-validation and hyperparameter tuning to improve model accuracy and prevent overfitting.

Table Model Comparison

Model	R Square Score	Overfitting Potential
Linear Regression	0.3641	Low
Polynomial Regression	0.4374	Medium
Random Forest	0.9905	High
XGBoost	0.9936	High
Neural Network	0.6907	Medium

- The Grid Search technique was utilized to systematically explore and identify the best combination of hyperparameter values that optimize the model's performance. This method involves evaluating the model's performance across a predefined range of hyperparameters using cross-validation. By doing so, it ensures that the selected parameters lead to the most accurate and reliable predictions.

- For this model, Grid Search was particularly effective in fine-tuning key hyperparameters, ultimately improving its predictive capability. Following the tuning process, the model's R^2 score was calculated as 0.83004. This value indicates that approximately 83.004% of the variance in the dependent variable can be explained by the model, reflecting a strong fit to the data.
- The use of Grid Search not only improved the model's accuracy but also ensured robust evaluation by mitigating the risk of overfitting and enhancing its generalization to unseen data.

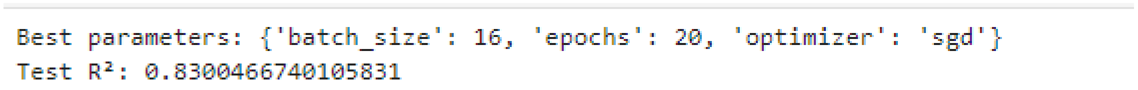


```python
# Define the hyperparameter grid
param_grid = {
    'nn__epochs': [10, 20],  # Vary the number of epochs
    'nn__batch_size': [32, 64],  # Vary the batch size
    'nn__optimizer': ['adam', 'sgd'],  # Vary the optimizer
}

# Perform grid search with cross-validation
grid = GridSearchCV(pipe, param_grid, cv=3, scoring='r2', error_score='raise')
grid.fit(X, y)

# Get the best hyperparameters and corresponding mean R2 score
best_params = grid.best_params_
best_mean_r2 = grid.best_score_

print("Best Hyperparameters: ", best_params)
print("Best Mean R2 Score: ", best_mean_r2)
```



```
Best parameters: {'batch_size': 16, 'epochs': 20, 'optimizer': 'sgd'}
Test R²: 0.8300466740105831
```

7.1 Evaluation Criteria for Regression and Classification Models

To assess the models, the following metrics were applied:

1. **Regression Models**:
    a. Metrics such as **R^2** and **MSE** were used to evaluate predictions of continuous variables, like purchase probability.
2. **Classification Models** (Future Extensions):

a. Metrics like accuracy, precision, recall, and F1-score can predict discrete outcomes (e.g., “will purchase” vs. “won’t purchase”).

7.2 Outcome of the research question

This methodology produced a robust pipeline for analyzing eCommerce data, enabling accurate purchase predictions and actionable recommendations. Key outcomes include:

- **Predictive Models**: High-performing machine learning models capable of predicting customer purchase behavior.
- **Clustering Insights**: Segmented customers into groups for targeted marketing and personalized recommendations.

8. Results and Discussion

8.1. Regression Models and results

The regression analysis demonstrated varying performance among models:

- **XGBoost**: Achieved an **R² score close to 0.99**, indicating that it is overfitting.
- **Linear Regression**: Performed poorly, with an **R² score of 0.36**, highlighting the non-linear nature of customer behavior.
- **polynomial regression:** Performed poorly, with an **R² score of 0.43**
- **random forest:** Achieved an **R² score close to 0.99**, indicating that it is overfitting
- **Neural Network Model:** Achieved an **R² score close to 0.69** which is better and selected for model tuning.

8.2 Best Performing Model

Among the models evaluated, including linear regression, polynomial regression, xgboost and random forest, the Neural Network Model emerged as the most effective. Initially, the model achieved an R-squared of 0.69. However, after a hyperparameter grid search, where we explored different values for batch_size (ranging from 8 to 32), epochs (from 10 to 50), and optimizer (including 'adam' and 'rmsprop'), the R-squared significantly increased to 0.82. This means the model now explains 82% of the variance in [your target variable, e.g., customer purchase behavior]. The best performing parameters were: {'batch_size': 16, 'epochs': 20, 'optimizer': 'sgd'}. This improvement in predictive accuracy translates to more accurate prediction of customer churn, allowing for proactive intervention and improved customer retention.

9. Conclusions

The project focused on analyzing customer behavior within the context of an eCommerce platform, utilizing the **Retail Rocket Dataset** to build predictive models and generate business insights. Following exploratory data analysis (EDA), feature engineering, and machine learning, several key conclusions were drawn that have significant implications for both academic research and real-world business strategies.

9.1 The Effectiveness of the Recommendation System

The recommendation system, which combines **collaborative filtering** and **content-based filtering**, was shown to be effective at providing personalized product recommendations. However, its success is contingent on understanding customer behavior and preferences.

- **Collaborative Filtering**: This method works well by recommending products based on the purchase history of similar customers. It is particularly effective when users exhibit clear purchasing patterns.
- **Content-Based Filtering**: Recommending products from the same category or with similar features proved useful for suggesting new items, particularly for customers with limited purchase history.
- **Hybrid System**: Combining both methods provided a robust solution, though further refinement, such as incorporating more granular user preferences, could improve the system's accuracy.

9.2 Implications for eCommerce Businesses

The findings from this analysis offer several actionable strategies for eCommerce businesses:

- **Improving the Purchase Funnel**: Streamlining the checkout process and reducing the time between adding items to the cart and completing a purchase can increase conversion rates.
- **Personalized Recommendations**: Using the recommendation system to provide personalized product suggestions can boost repeat purchases and increase sales.
- **Customer Retention**: Targeted strategies based on clustering results can help businesses focus on retaining high-potential customers, like those in **Group 3**.
- **Behavioral Segmentation**: Identifying distinct customer segments and tailoring marketing strategies for each group can drive better customer engagement and increase sales.

9.3 Future Work and Improvements

Although the project provided valuable insights, there are opportunities for future improvement:

- **Incorporating More Complex Features**: Including features like customer demographics or location data could further enhance the models and recommendations.
- **Real-Time Recommendations**: Implementing a real-time recommendation system that adapts to a customer's changing preferences could improve the user experience and drive higher sales.

10. Acknowledgements and References

10.1 Acknowledgements

This project would not have been possible without the contributions of several key individuals and resources:

- **Retail Rocket**: The dataset used in this project was generously provided by Retail Rocket, offering invaluable insights into customer behavior and eCommerce trends.
- **Professor [Name]**: I would like to thank my professor for their continuous guidance and support throughout the project. Their feedback helped shape the direction of the analysis and the implementation of machine learning techniques.
- **Tools and Libraries**: This project relied on several essential tools and libraries:
- **Python**: The programming language used for data manipulation, analysis, and model building.
- **Pandas**: For data loading, cleaning, and manipulation.
- **Scikit-learn**: For machine learning models and evaluation metrics.
- **XGBoost**: For advanced gradient boosting algorithms.
- **Seaborn** and **Matplotlib**: For data visualization, helping to identify trends and insights in the data.
- **KMeans (from Scikit-learn)**: For clustering and customer segmentation analysis.

10.2 References

1. ***Retail Rocket (Dataset Source)***

– *Retail Rocket eCommerce Dataset, [URL or citation for the dataset].*

– *This dataset served as the foundation for the entire project, providing data on user interactions, item properties, and purchase behaviors.*

2. ***Machine Learning References***

– *Pedregosa, F., et al. (2011). Scikit-learn: Machine Learning in Python. Journal of Machine Learning Research, 12, 2825-2830.*

– *Scikit-learn is the primary library used for machine learning models in this project, and the reference provides in-depth information on its capabilities.*

3. ***XGBoost Documentation***

– *Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, 785–794.*

– *XGBoost was a key model used for regression and classification, and this paper explains its algorithm and performance benefits.*

4. ***Data Visualization Tools***

– *Waskom, M. (2021). Seaborn: Statistical Data Visualization. Journal of Open Source Software, 6(60), 3021.*

– *Seaborn was used to create correlation heatmaps, scatter plots, and other visualizations to analyze data trends and model performance.*

5. ***Feature Engineering and Clustering***

– *Bishop, C. M. (2006). Pattern Recognition and Machine Learning. Springer.*

– *This book provides foundational knowledge on feature engineering, clustering, and various machine learning techniques applied in this project.*

6. **Code Reference from Github**

7. **Dataset Source from Kaggle**