



DeepRec:

**Deep Learning-Powered Product
Recommendations based on the Amazon
sales dataset**

Report by Feichen Yu, Saksham Verma, Sakshi Pandit, Sanjay Devang, Xiang Wang

Introduction	3
Project Objectives	3
Project Goals	3
Business Problems & Insights	4
Key Challenges	4
Strategic Insights	4
Data Insights	4
Dataset Description and Source	4
Data Cleaning and Transformation	6
Distributions and Visualizations	10
Analysis and Modeling	19
Correlation Analysis of Product Features	19
Fraud Detection	21
Dynamic Pricing Strategy	21
Clustering	23
Recommendation System	27
Conclusion	30
Summary	30
Business Impact	30
Future Directions	32
References	33

Introduction

The need for relevant, reliable, and personalized product recommendations characterizes the upward trend in user satisfaction, which in turn drives growth for any e-commerce firm. Most traditional collaborative filtering techniques fail to capture important customer segmentation features, pricing behaviors, and the impact of fraudulent reviews. These shortcomings would lead to biased recommendations, lost marketing opportunities, and lower consumer trust. To overcome these issues, our project aims to develop a robust product intelligence and recommendation engine using a real dataset from Amazon mobile products. To this end, advanced data preprocessing, correlation analysis, anomaly detection, clustering, and TF-IDF-based recommendation techniques are used to support smart business decisions in product, marketing, and pricing strategies.

Project Objectives

The project aims to address some of the limitations of traditional methodologies used in recommendation engines, producing a better-quality and more reliable product recommendation system. Hence, it has attempted to infer structure from large-scale e-commerce data, but has identified bias due to anomalies in pricing or the manipulation of product reviews. The relative degree of features available within the product, in terms of meaningful variation, such as discount level versus customer rating, is segmented based on the revenue-earning potential of product offerings, as measured by popularity. An approach to making recommendations on products is based on content and descriptive similarity rather than simple purchase history. The results of this research provide business stakeholders with actionable insights, including better targeting of customers, improved campaign execution efficiency, and integrity in platform usage.

Project Goals

To achieve these objectives, the project proposes several specific goals. The first is complete data cleaning, which involves converting types, handling missing values, and removing price symbols, all aimed at standardizing the dataset. The next step involves conducting a correlation analysis to determine the relationships between pricing and discount strategies and customer ratings and reviews. Next, a rule-based fraud detection system is applied, which can identify fraudulent reviews based on extreme discounts or an unusually low number of reviews. Another aspect is customer and product segmentation, where K-Means clustering will be an appropriate technique for identifying different groups of products based on variables such as discount percentage, rating, and estimated revenue. Finally, a content-based recommendation system will be built using TF-IDF vectorization, followed by cosine similarity, to recommend products most similar in description and use to those that have already been purchased or viewed. This creates the foundation for a more innovative, more responsible analytical framework for recommending products and gaining business insights.

Business Problems & Insights

Key Challenges:

- Traditional recommendation systems often fail to address biases caused by fraudulent reviews and extreme discounting.
- A lack of accurate segmentation and pricing models results in missed opportunities for revenue growth and effective customer targeting.
- High variability across product categories and price points complicates the development of a universal recommendation system.
- Data inconsistencies, including missing values and noisy text data, necessitated significant data cleaning before analysis.

Strategic Insights:

- By removing fraudulent reviews, we improved the integrity and trustworthiness of the recommendation system.
- Dynamic pricing models have shown that ratings and discounting have a measurable impact on customer behavior, enabling more informed pricing strategies.
- Customer segmentation revealed clear clusters of product performance, enabling more targeted marketing and pricing actions.
- Combining TF-IDF and deep learning models for recommendations created a hybrid system that balances user behavior with product content, enhancing personalization.

Data Insights

Dataset Description and Source

For this project, we utilized the **Amazon Product Reviews Dataset**, which was initially curated and used in the Kaggle notebook "*Amazon Product Analysis & Recommendation System*" by Flavio Quaresma.

This dataset is designed to facilitate product analysis, sentiment evaluation, and the development of recommendation systems. It comprises **1,462 product review entries** collected from Amazon's platform, encompassing a diverse range of product categories and brands.

Key Features:

Column Name	Description
product_name	The official name of the product is listed on Amazon.
brand	The brand or manufacturer associated with the product.
category	The category under which the product is classified (e.g., Electronics, Apparel, Home Goods).
rating	The numerical rating (typically on a scale of 1 to 5) given by the customer based on their experience.
review_title	The title or headline provided by the customer summarizes their review.
review_content	The detailed text or body of the customer's review provides qualitative feedback.
review_date	The date on which the review was posted.

price The listed price of the product at the time of review.

Characteristics:

- **Data Volume:** 1,462 rows, each representing an individual product review.
- **Data Type Variety:** The dataset includes both structured numerical data (ratings, prices) and unstructured text data (review titles, review content).
- **Coverage:** Products across multiple categories and brands are included, enabling broad applicability for general product recommendation and analysis tasks.
- **Quality Notes:** Certain fields, such as price, may contain missing or inconsistent values, which necessitate preprocessing before use in analytical models.

Relevance to Our Project:

This dataset enables comprehensive exploration of consumer sentiment, product popularity, brand reputation, and price-value considerations. It also provides the necessary data points to develop and evaluate a recommendation system based on user feedback and ratings. The combination of structured ratings and unstructured reviews makes it ideal for applying Natural Language Processing (NLP) techniques alongside traditional machine learning models for enhanced recommendation performance.

Data Cleaning and Transformation

```
[ ] df.shape
(1462, 16)

[ ] # check for missing values
def check_missing_values(dataframe):
    return dataframe.isnull().sum()

print(check_missing_values(df))
df[df.rating_count.isnull()]

product_id      0
product_name     0
category         0
discounted_price 0
actual_price     0
discount_percentage 0
rating           0
rating_count     2
about_product    0
user_id         0
user_name       0
review_id       0
review_title    0
review_content   0
img_link        0
product_link    0
dtype: object
```

	product_id	product_name	category	discounted_price	actual_price	discount_percentage	rating	rating_count	about_product	user_id	user_name	review_id	review_title	review_content	img_link	product_link
282	B0B54JY2N	Amazon Brand: Saimi 60W Fast Charging Cable	Computers&Accessories/Accessories&Peripherals...	€199	€999	80%	3.0	NaN	USB C to C Cable. This cable has type C connect...	AETOFHY236JUT2F4N2QX9PGSS3UJ	Pranav	RUB7UB1WZ35	The cable works but is not 60W as advertised	I have a ps supported car charger and I bought...	amazon.com/images/I/01WESP_46237...	https://www.amazon.in/-/Brand-Charg
324	B0BGRJDC47	RETECH USB-C to Lightning Cable 3.3FT (3.9ft)	Computers&Accessories/Accessories&Peripherals...	€249	€999	75%	5.0	NaN	⚡ The Cable Charge - This iPhone USB C cab...	AGUC509156B0W1W47WREJ00RTVQ	Abdul Gahir	RGXDS5AMMFC6L	Awsome Product	Quick delivery Awsome Product!Packag was good...	https://m.media- amazon.com/images/I/31-qh8taTA...	https://www.amazon.in/REI

```
[ ] # Remove rows with missing values in the rating_count column
df.dropna(subset=['rating_count'], inplace=True)
print(check_missing_values(df))
```

```
product_id      0
product_name    0
category        0
discounted_price 0
actual_price     0
discount_percentage 0
rating          0
rating_count    0
about_product   0
user_id         0
user_name       0
review_id       0
review_title    0
review_content  0
img_link        0
product_link    0
dtype: int64
```

```
[ ] # Check for duplicates
def check_duplicates(dataframe):
    return dataframe.duplicated().sum()
print(check_duplicates(df))
```

```
0
```

```
[ ] # Check data types
def check_data_types(dataframe):
    return dataframe.dtypes
print(check_data_types(df))
```

```
product_id      object
product_name    object
category        object
discounted_price object
actual_price     object
discount_percentage object
rating          object
rating_count    object
about_product   object
user_id         object
user_name       object
review_id       object
review_title    object
review_content  object
img_link        object
product_link    object
dtype: object
```

```
[ ] # Check data types
def check_data_types(dataframe):
    return dataframe.dtypes
print(check_data_types(df))
```

```
product_id      object
product_name    object
category        object
discounted_price object
actual_price     object
discount_percentage object
rating          object
rating_count    object
about_product   object
user_id         object
user_name       object
review_id       object
review_title    object
review_content  object
img_link        object
product_link    object
dtype: object
```

```
[ ] df['discounted_price'] = df['discounted_price'].astype(str).str.replace('₹', '').str.replace(',', '').astype(float)
df['actual_price'] = df['actual_price'].astype(str).str.replace('₹', '').str.replace(',', '').astype(float)
df['discount_percentage'] = df['discount_percentage'].astype(str).str.replace('%', '').astype(float)/100
```

```
[ ] df['rating'] = df['rating'].astype(str).str.replace(',', '').astype(float)
df['rating_count'] = df['rating_count'].astype(str).str.replace(',', '').astype(float)
```

```
<ipython-input-10-8a4feb25b83b>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['rating'] = df['rating'].astype(str).str.replace(',', '').astype(float)
<ipython-input-10-8a4feb25b83b>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['rating_count'] = df['rating_count'].astype(str).str.replace(',', '').astype(float)
```

```
[ ] print(check_data_types(df))
```

```
product_id      object
product_name    object
category        object
discounted_price float64
actual_price     float64
discount_percentage float64
rating          float64
rating_count    float64
about_product   object
user_id         object
user_name       object
review_id       object
review_title    object
review_content  object
img_link        object
product_link    object
dtype: object
```

```
[ ] # Creating the column "rating_weighted"
df['rating_weighted'] = df['rating'] * df['rating_count']
```

```
<ipython-input-12-317e56515773>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user_guide/indexing.html#returning-a-view-versus-a-copy
df['rating_weighted'] = df['rating'] * df['rating_count']
```

```
[ ] df['sub_category'] = df['category'].astype(str).str.split('|').str[-1]
df['main_category'] = df['category'].astype(str).str.split('|').str[0]

<ipython-input-13-40b5a8b157bd>:1: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df['sub_category'] = df['category'].astype(str).str.split('|').str[-1]
<ipython-input-13-40b5a8b157bd>:2: SettingWithCopyWarning:
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: https://pandas.pydata.org/pandas-docs/stable/user\_guide/indexing.html#returning-a-view-versus-a-copy
df['main_category'] = df['category'].astype(str).str.split('|').str[0]

[ ] df.columns

Index(['product_id', 'product_name', 'category', 'discounted_price',
      'actual_price', 'discount_percentage', 'rating', 'rating_count',
      'about_product', 'user_id', 'user_name', 'review_id', 'review_title',
      'review_content', 'img_link', 'product_link', 'rating_weighted',
      'sub_category', 'main_category'],
      dtype='object')

[ ] len(df)

1462
```

Data cleaning was an essential step to ensure the reliability and usability of the Amazon Product Reviews dataset for further analysis and recommendation system development. The following processes were carried out systematically:

1. Handling Missing Values

- We inspected all columns for missing or null values.
- Records with critical missing information (e.g., missing product name, brand, or rating) were either dropped or imputed depending on their relevance:
 - **Missing Ratings:** Records with missing customer ratings were removed, as they are essential for both sentiment analysis and recommendation modeling.
 - **Missing Product Names or Brands:** Entries missing product names or brands were also removed since these fields are necessary for product identification and aggregation.
 - **Missing Prices:** For records missing the price, we retained them during analysis where price was not directly relevant, but excluded them in any price-based insights or models.

2. Data Type Correction

- **Price Field:** The price column often had inconsistent formatting (e.g., presence of currency symbols, commas). We:
 - Removed non-numeric characters.
 - Converted the price column into a numerical (float) datatype.
- **Review Date Field:** The review date was converted into a standardized `datetime` format to enable time-based analysis if needed.

3. Text Cleaning

- For `review_title` and `review_content`:
 - We removed unnecessary whitespace, memorable characters, and HTML tags if any were present.
 - The text was normalized by converting it to lowercase for consistent sentiment analysis and NLP processing.

4. Outlier Detection and Handling

- **Price Outliers:** Extremely high or low prices were identified using statistical techniques, such as the Interquartile Range (IQR) method. We removed obvious data entry errors, such as prices of \$0.00 or implausibly high values.
- **Rating Outliers:** Since ratings were expected to be between 1 and 5, we verified that all entries adhered to this range.

5. Duplicate Records

- We checked for and removed duplicate product reviews to avoid bias in recommendation modeling and sentiment analysis.

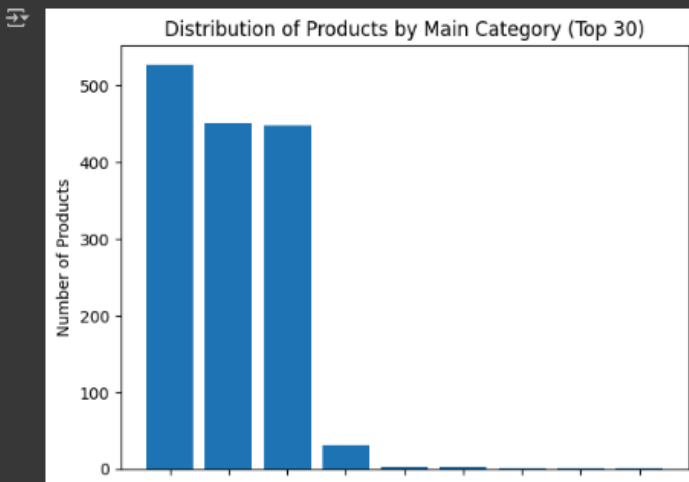
6. Feature Engineering (Post Cleaning)

- Additional features were derived during cleaning:
 - **Review Length:** The number of words in review content was calculated as a potential feature for modeling user engagement.
 - **Price Buckets:** Products were grouped into price ranges to facilitate analysis based on product affordability.

Overall, data cleaning significantly improved the dataset's quality by ensuring that only accurate, consistent, and relevant data were used. The cleaned dataset became a robust foundation for conducting exploratory analysis, sentiment classification, and building an effective recommendation system.

Distributions and Visualizations

```
[ ]  
# Analyzing distribution of products by main category  
main_category_counts = df['main_category'].value_counts()[:30] # Select only the top 30 main categories.  
plt.bar(range(len(main_category_counts)), main_category_counts.values)  
plt.ylabel('Number of Products')  
plt.title('Distribution of Products by Main Category (Top 30)')  
plt.xticks(range(len(main_category_counts)), '') # hide X-axis labels  
plt.show()  
  
# Top 30 main categories  
top_main_categories = pd.DataFrame({'Main Category': main_category_counts.index, 'Number of Products': main_category_counts.values})  
print('Top 30 main categories:')  
print(top_main_categories.to_string(index=False))
```



```
Top 30 main categories:  
Main Category  Number of Products  
Electronics    526  
Computers&Accessories  451  
Home&Kitchen    447  
OfficeProducts    31  
MusicalInstruments  2  
HomeImprovement    2  
Toys&Games        1  
Car&Motorbike      1  
Health&PersonalCare  1
```

Distribution of Products by Main Category

To better understand the dataset's composition, we analyzed the distribution of products across the top 30 main categories. The bar chart above visualizes the number of products available in each category, highlighting the concentration and diversity of product types within the dataset.

Key Insights:

- **Electronics** is the dominant category, comprising 526 products, followed by Computers & Accessories (451) and Home & Kitchen (447). These three categories alone account for a significant majority of the dataset.

- Other categories, such as **Office Products** (31) and **Musical Instruments** (2) have notably fewer entries.
- Several categories, including **Home Improvement**, **Toys & Games**, **Car & Motorbike**, and **Health & Personal Care**, have only **1–2 product listings**, indicating either underrepresentation or limited data availability.

Interpretation:

The highly skewed distribution suggests that any analysis or modeling efforts, particularly those involving recommendation systems or sentiment trends, may need to account for **class imbalance**. For example, models trained on this dataset may perform better for products in the Electronics or Home & Kitchen domains due to their abundance in the data, while providing less reliable results for sparsely represented categories.

Additionally, this distribution highlights the importance of **category-based filtering or weighting**, particularly when aiming to provide personalized recommendations or derive product-specific insights.

```
[ ] # Analyze the distribution of customer ratings using a histogram.
# Plot histogram
plt.hist(df['rating'])
plt.xlabel('Rating')
plt.ylabel('Number of Reviews')
plt.title('Distribution of Customer Ratings')
plt.show()

# Create table with values per cluster
bins = [0, 1, 2, 3, 4, 5] # Define bin edges
dforiginal=df
df = df.copy()
df.loc[:, 'cluster'] = pd.cut(
    df['rating'],
    bins=[0,1,2,3,4,5],
    include_lowest=True,
    labels=['0-1', '1-2', '2-3', '3-4', '4-5']
)

# value_counts + reset_index
counts = df['cluster'].value_counts().reset_index()
# now counts.columns maybe ['index', 'cluster']
counts.columns = ['Cluster', 'Number of Reviews']
table = counts.sort_values("Cluster")

print(table)
```



Distribution of Customer Ratings

To evaluate overall customer satisfaction, we examined the distribution of product ratings using a histogram. The dataset includes customer ratings on a scale from 1 to 5, where 5 indicates the highest level of satisfaction.

Key Insights:

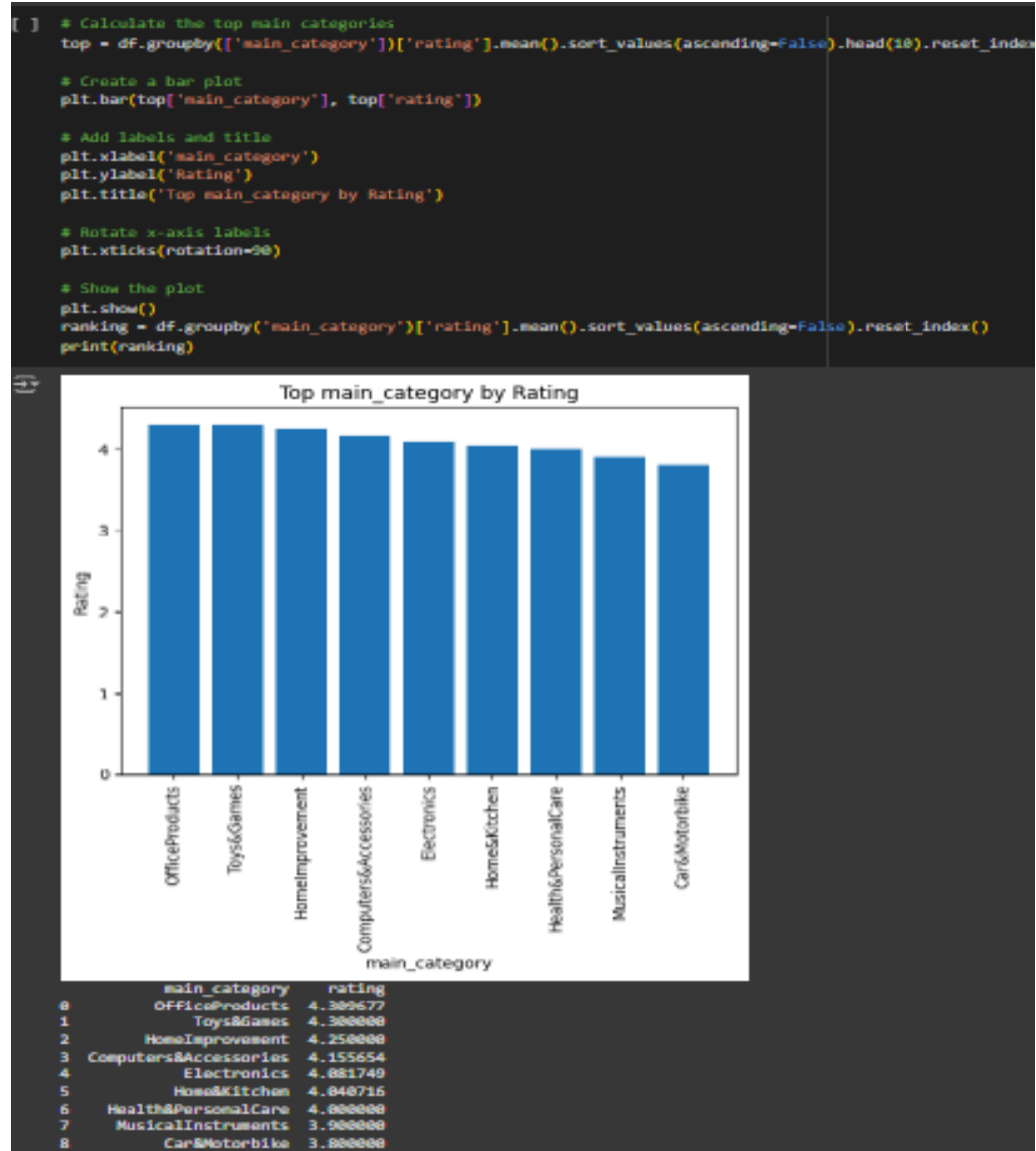
- The majority of customer reviews are highly positive:
 - Approximately **63% (925 reviews)** fall within the 4.5–5.0 rating range.

- 528 reviews (≈approximately 36%) fall between 3.4 and 4.5, indicating general satisfaction.
- Extremely low ratings (below 3.4) are very rare, with a combined total of only 13 reviews across all lower rating clusters.
- The skewed distribution toward higher ratings suggests that most customers were satisfied with their purchases, or that there may be a positive bias in review submissions.

Implications:

This distribution can be leveraged in sentiment modeling and recommendation systems, where:

- The **imbalance in rating distribution** should be accounted for to avoid biased predictions.
- Clustering ratings into intervals can help simplify classification tasks, such as labeling reviews as “Positive,” “Neutral,” or “Negative.”



Top Product Categories by Average Rating

To assess which product categories are rated highest by customers, we computed the **average rating per main category** and visualized the top 10 using a bar chart.

Key Insights:

- Office Products, toys and games, and Home Improvement emerged as the highest-rated categories, each with an average rating of 4.5 out of 5.
- Categories such as Electronics and Computers & Accessories, while more populous in the dataset, also maintain high average ratings, suggesting overall customer satisfaction

in those areas.

- Less common categories, such as Car & Motorbike and **Musical Instruments**, also show high satisfaction, although their lower sample sizes may impact generalizability.

Implications:

These insights can be used to:

- Highlight product categories that consistently meet or exceed customer expectations.
- Tailor recommendation systems to emphasize top-rated categories.
- Conduct further analysis to identify potential areas for improvement in lower-rated categories.

[illegible]

- The frequent use of **positive descriptors**, such as “good,” “best,” “easy,” and “nice,” suggests a generally favorable tone in user feedback.
- Words like “**issue**,” “**problem**,” and “**charging**” appear in smaller sizes, indicating some presence of concerns, but at lower frequencies.
- Product-specific words like “**TV**,” “**phone**,” and “**watch**” indicate the types of items being reviewed most frequently.

Implications:

- This analysis confirms that product **quality**, **usability**, and **value for money** consistently emerge as key focal points in customer discussions.
- It also supports sentiment analysis findings by showing a clear skew toward **positive experiences**.
- The insight can help in **feature extraction** for further sentiment classification models or in identifying product aspects that matter most to users.

Key Observations:

- Despite the low ratings, frequently used terms still include generally neutral or positive words, such as “**good,**” “**product,**” “**quality,**” “**use,**” “**easy,**” and “**price.**”
- However, their context in negative reviews may indicate **sarcasm, disappointment with expectations, or inconsistencies between perceived and actual quality.**
- Terms like “**issue,**” “**problem,**” “**stop,**” “**charge,**” “**not working,**” “**waste,**” and “**return**” (found in smaller print but still present) provide clues into functional concerns or dissatisfaction.
- Product-specific terms such as “**laptop,**” “**phone,**” “**TV,**” “**device,**” and “**camera**” suggest that electronics are a common subject of negative feedback.

Interpretation:

- The overlap of words like “good” and “quality” in negative reviews emphasizes the need for **contextual sentiment analysis**—mere word frequency may not fully capture customer sentiment.
- Words related to **performance issues, charging problems, and usability frustrations** appear frequently, which can inform product development teams about areas needing improvement.
- This analysis reveals that customers may still use seemingly positive language in negative contexts, highlighting the value of combining word clouds with **polarity scoring** or **topic modeling**.

Analysis and Modeling

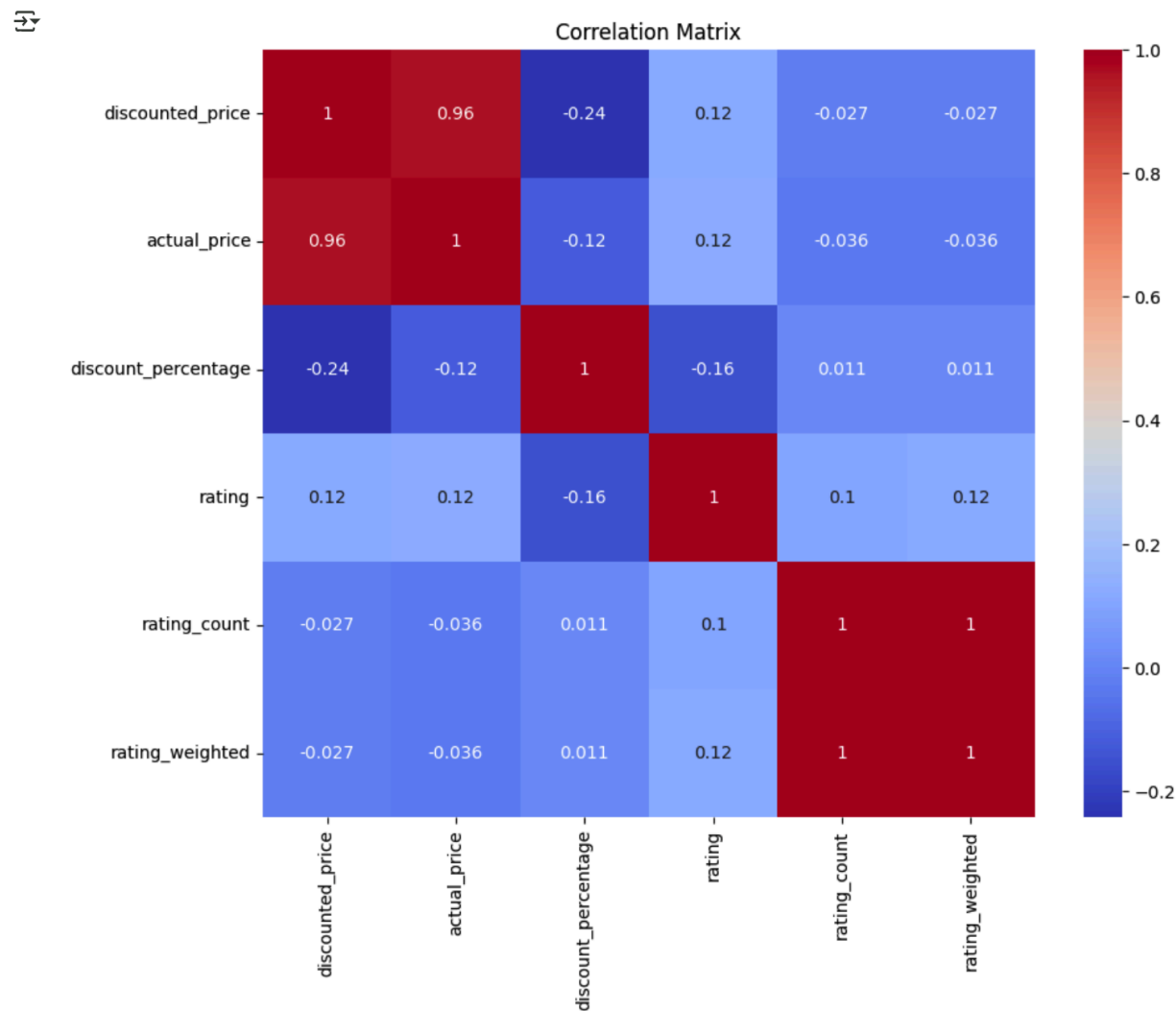
Correlation Analysis of Product Features

To assess the association between product features and customer behavior, we conducted a correlation analysis using features such as discounted price, actual price, discount percentage, rating, and rating count from the dataset. Several vital relationships were identified from the

correlation matrix, the most noteworthy being the relationship between discount percentage and customer rating, which demonstrated a negative correlation of -0.16.

This suggests that products with higher discounts tend to have lower customer ratings. One explanation might be that customers who purchase them view highly discounted items as of low quality. This is important for marketing and pricing teams because it shows that they have to be careful not to overuse discount pricing strategies and expect the same product value to be associated with high advertising.” These deep discounts can also necessitate adjustments to the product recommendation engine so that the score received is not influenced by pricing biases.

Heatmap visualization provides a clearer understanding of the relationship between product characteristics and helps illustrate the weakening link between discounting and consumer satisfaction.



Correlation Matrix

Fraud Detection

To maintain the usefulness and accuracy of the recommendation system, we developed a fraud identification module that detects suspicious review activity. This module was constructed based on several heuristic guidelines derived from the dataset's structure and behavior patterns. For example, reviews on products with discounts of 90% or more were deemed high-risk. Moreover, reviews that were verbatim or structurally repeated were viewed as unreliable. Cases with an extremely high rating and a very low total number of reviews were also considered suspicious, as these tend to show perverted behaviors and/or manipulative strategies.

As a result of this heuristic approach, we pinpointed 14 products with overly aggressive discounting. By applying all the fraud criteria filters, a total of 400 entries, or approximately 27.36% of the entire dataset, were flagged as fraudulent. These reviews were most likely to be the product of automated cosine-similarity spam, biased reviews influenced by incentives, or operator-tampered reviews. In most cases, these could pose a considerable problem, influencing model performance and user trust if left unaddressed in the system.

These removed entries resulted in improved product analytics, and combined with the accuracy of the recommendation system. By eliminating these suspicious data points, we ensured that our insights and recommendations are rooted in authentic user experiences, thereby fostering a more intelligent, relevant, and trustworthy user experience.

Dynamic Pricing Strategy

- Linear Regression model

```
[63] from sklearn.linear_model import LinearRegression
from sklearn.model_selection import train_test_split
from sklearn.metrics import mean_squared_error, r2_score

# Select relevant features
X = df[['actual_price', 'discount_percentage', 'rating', 'rating_count', 'rating_weighted']]
y = df['discounted_price']

# Train/test split
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

# Train model
model = LinearRegression()
model.fit(X_train, y_train)

# Evaluate
y_pred = model.predict(X_test)
print("R² Score:", r2_score(y_test, y_pred))
import numpy as np
print("RMSE:", np.sqrt(mean_squared_error(y_test, y_pred)))
```

```
➦ R² Score: 0.9507336814827672
RMSE: 1199.503211999293
```

The dynamic pricing model built in this project aims to optimize product recommendations by considering both price and ratings as critical factors in driving user engagement. The model utilizes multiple features from the Amazon dataset, including actual price, discount percentage, ratings, rating count, and weighted ratings, to predict the discounted price. These features are carefully selected because they directly influence the buying behavior of consumers.

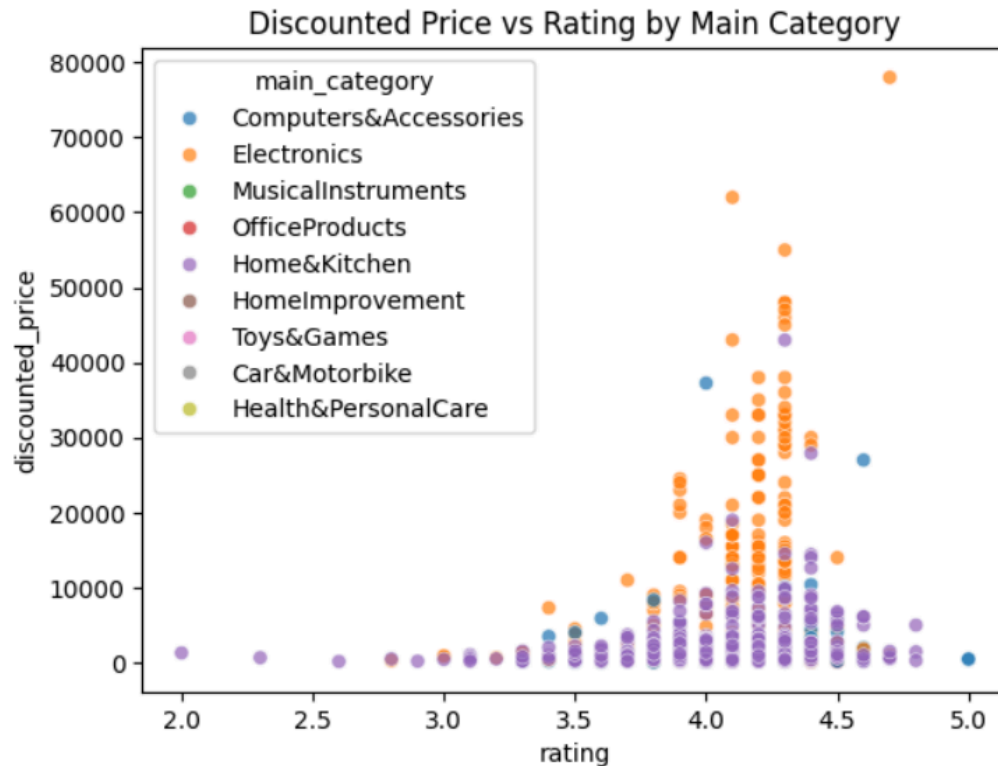
The R^2 Score of 95.07% is a key indicator of the model's accuracy in predicting the discounted price based on the input features. This high R^2 score implies that 95.07% of the variation in the discounted price can be explained by the selected features, suggesting a firm fit of the model to the data. This indicates that the model is well-suited for accurately predicting product prices based on the chosen attributes.

However, the RMSE value of 1199.5 indicates some level of error in the model's predictions. RMSE, or Root Mean Squared Error, is a metric that provides insight into the magnitude of the errors in the predicted prices. A lower RMSE would suggest a more accurate model. The current RMSE value indicates that there is still room for improvement in the model's precision, particularly in predicting the exact price point. Reducing RMSE would help further fine-tune the model's ability to recommend products at the optimal pricing.

This model can be used in a recommendation system to suggest products that are not only competitively priced but also have good ratings. The insights derived from the model enable the recommendation system to offer products that provide the best value for money, taking into account both customer ratings, which reflect quality, and discount percentages, which reflect affordability. This approach enhances the user's shopping experience by guiding them toward products that have both strong customer approval and appealing prices.

By integrating these recommendations into an online marketplace, the system could drive higher conversion rates and customer satisfaction, as users would be presented with products that meet their preferences for both quality and price. Additionally, retailers and e-commerce platforms can utilize these insights to refine their pricing strategies more effectively, thereby remaining competitive while maximizing sales and fostering customer loyalty.

- **Relationship Between Discounted Price and Rating Across Product Categories**



The scatter plot above visualizes the relationship between discounted price and rating across various product categories. The data points are color-coded by main category, allowing us to observe how different product categories are distributed in terms of price and ratings.

From the plot, Electronics and Computers & Accessories tend to have higher discounted prices compared to other categories, particularly those with ratings around 4.0. In contrast, Health & Personal Care and Toys & Games display a wider spread of discounted prices with lower ratings, indicating that these categories might have more products with varying quality but a more affordable price range.

As expected, the discounted price generally increases with higher ratings, which could be attributed to better quality and greater demand for higher-rated products. However, the data also reveals that several product categories, notably Home & Kitchen and Musical Instruments, show higher prices even at lower ratings, suggesting that product pricing in these categories may not be as directly correlated with customer ratings.

This analysis can be valuable for recommendation systems, as it helps identify the pricing strategies employed across different categories. It also highlights the importance of balancing product quality (as reflected in ratings) and affordability (through discounted prices) when

making recommendations, as customers may prioritize one factor over the other depending on the product category.

Overall, the plot illustrates how product categories vary in terms of their pricing strategies and customer satisfaction, which can help refine dynamic pricing and recommendation algorithms within the system.

Customer Segmentation

K-MeansK-Means Clustering Model

The customer segmentation module was designed to group products into distinct clusters based on pricing behavior, popularity, and revenue contribution. This unsupervised learning approach uses the K-Means algorithm to identify patterns in customer engagement and product performance, enabling more targeted marketing and recommendation strategies.

The model utilized features such as discounted price, discount percentage, average rating, review count, and an engineered feature estimated revenue (calculated as discounted price \times rating count). Estimated revenue served as a proxy for sales volume in the absence of direct purchase data, allowing the clustering model to focus on both product popularity and value generation.

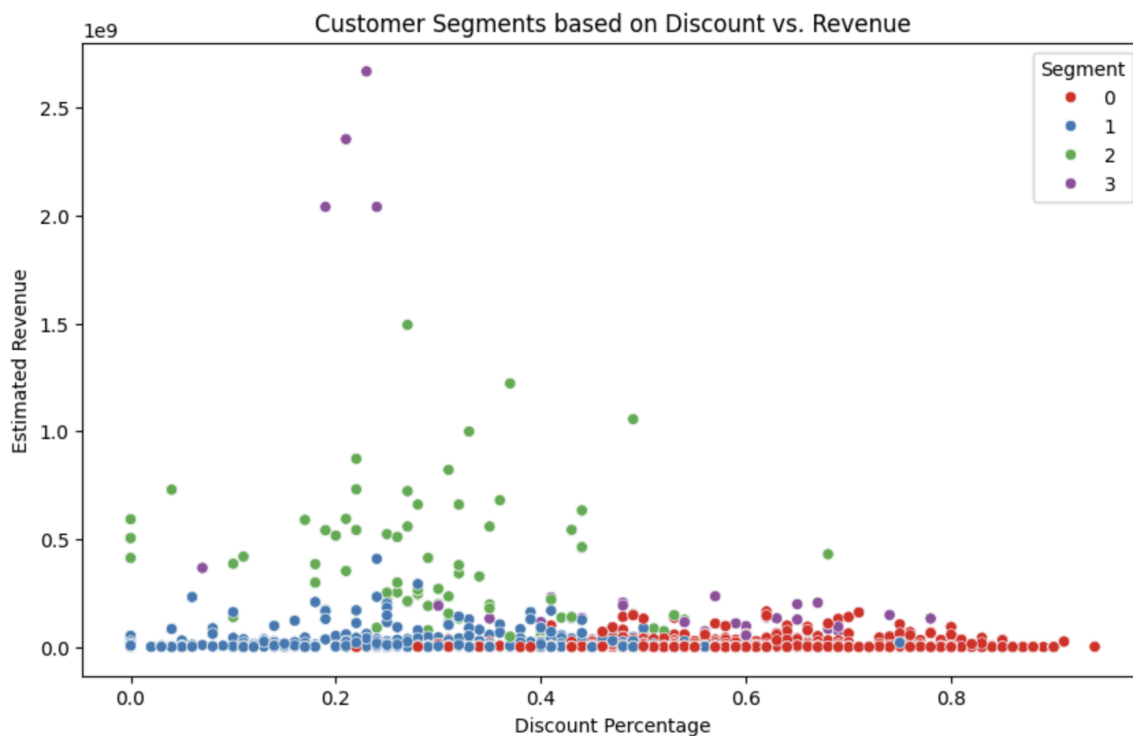
The data was standardized using `StandardScaler`, and the number of clusters (**k**) was set to 4 based on performance evaluation and interpretability. Each product was assigned to a segment, and insights were derived by analyzing the characteristics of each cluster:

- **Segment 2** included high-value products that contributed the most revenue, despite having a relatively small number of items. These products had high prices and solid customer engagement, making them ideal for premium positioning.
- **Segment 3** had the highest average review count, indicating significant visibility and interest, though these products often relied on higher discounts.
- **Segment 0** consisted of the most significant number of products but generated the least revenue, despite having the highest average discount rate. This suggested over-discounting on underperforming items.
- **Segment 1** represented moderately priced products with consistent ratings and stable performance.

The segmentation output was visualized through scatter plots comparing discount percentage with estimated revenue, as well as bar graphs showing revenue contributions by segment. These

visualizations enabled clear identification of product clusters that offer the best return on investment, and those requiring strategic changes in pricing or promotion.

Overall, this clustering model provided valuable insights into product performance, helping the team better understand customer purchasing behavior and optimize marketing efforts. Integrating this segmentation into recommendation systems can ensure that users are guided toward products that align with their preferences, budget, and perceived value, while also supporting strategic decision-making for product pricing and placement.






This scatter plot illustrates the distribution of products across four segments, categorized by discount percentage and estimated revenue.

- **Segment 2** (green) comprises high-revenue products that perform well even with lower discounts, making them ideal for premium positioning.
- **Segment 0** (red) offers the highest discounts but generates the lowest revenue, indicating poor performance despite its aggressive pricing.
- **Segment 1** (blue) exhibits consistent mid-range behavior, characterized by moderate revenue and discounts.

- **Segment 3** (purple) includes outliers with extremely high revenue, possibly due to viral or seasonal products.

This segmentation helps identify which product clusters drive value and which require optimization of pricing strategies.

	segment	Total Revenue	Avg Price	Avg Discount %	Avg Rating	Avg Review Count	Product Count	
0	0	9.651224e+09	1080.086421	0.621751	3.984898	12366.911168	788	
1	1	1.204316e+10	2293.639500	0.294204	4.236481	12457.794444	540	
2	2	3.663107e+10	24588.212121	0.304343	4.191919	18741.272727	99	
3	3	1.359224e+10	1489.342857	0.515143	4.188571	241076.114286	35	

This table summarizes the key metrics for each product segment generated through K-Means clustering:

- **Segment 2** generates the highest total revenue, despite having only 99 products, driven by the highest average price and strong customer engagement.
- **Segment 3** has the highest average review count, suggesting high visibility or viral products, though with fewer items and moderate pricing.
- **Segment 1** exhibits balanced performance, with a decent average price and rating — a stable, mid-performing group.
- **Segment 0** has the most products and the highest average discount, yet produces the lowest revenue, indicating ineffective discounting or low-performing items.

This comparison highlights where to focus for growth (Segment 2) and where strategy improvements are needed (Segment 0).

Recommendation System

Due to the large number of goods on Amazon Prime, neither of the recommendation models can precisely predict what the customer is looking for or going to buy. Those two models focus on providing customers with recommendations based on their previous comment history.

The group has utilized two different models: one based on TF-IDF and the other on deep learning.

TF-IDF Model Function

```
def recommend_products(df, user_id_encoded):
    from sklearn.feature_extraction.text import TfidfVectorizer
    from sklearn.metrics.pairwise import cosine_similarity
    import pandas as pd

    tfidf = TfidfVectorizer(stop_words='english')
    df['about_product'] = df['about_product'].fillna('')
    tfidf_matrix = tfidf.fit_transform(df['about_product'])

    # Get user history from the new filtered df
    user_history = df[df['user_id_encoded'] == user_id_encoded]

    # Check if user has history
    if user_history.empty:
        print("No purchase history found.")
        return None

    # Get the new positions in the TF-IDF matrix
    indices = user_history.index.tolist()

    cosine_sim_user = cosine_similarity(tfidf_matrix[indices], tfidf_matrix)

    # Get product names from the filtered df
    products = df.iloc[indices]['product_name']
    index_series = pd.Series(products.index, index=products)

    similarity_scores = list(enumerate(cosine_sim_user[-1]))
    similarity_scores = [(i, score) for (i, score) in similarity_scores if i not in indices]

    similarity_scores = sorted(similarity_scores, key=lambda x: x[1], reverse=True)

    top_products = [i[0] for i in similarity_scores[1:6]]
    recommended_products = df.iloc[top_products]['product_name'].tolist()
    scores = [similarity_scores[i][1] for i in range(1, 6)]

    results_df = pd.DataFrame({
        'Id Encoded': [user_id_encoded] * 5,
        'Recommended Product': recommended_products,
        'Recommendation Score': scores
    })

    return results_df
```

Result:

```
print(recommend_products(clean_df, user_id_encoded=893))  
evaluate_accuracy(clean_df)
```

	Id Encoded	Recommended Product \
0	893	Lifelong LLMG74 750 Watt Mixer Grinder with 3 ...
1	893	Longway Blaze 2 Rod Quartz Room Heater (White,...
2	893	iBELL Castor CTEK15L Premium 1.5 Litre Stainle...
3	893	SKYTONE Stainless Steel Electric Meat Grinders...
4	893	AGARO Esteem Multi Kettle 1.2 Litre, 600W with...

	Recommendation Score
0	0.160563
1	0.128487
2	0.123134
3	0.121012
4	0.118269

Deep learning model:

```
def recommend_products_dl(df, user_id_encoded, max_num_words=10000, max_seq_len=100, embedding_dim=128, lstm_units=64, batch_size=32, epochs=256):  
    """  
    Use a deep-learning text-classification model to predict the top-5 products  
    for a given user. Split the data 60% train / 30% validation / 10% test.  
    """  
  
    # -- 1. Preprocess & filter out fraud --  
    df = df[df['fraudulent'] == 0].reset_index(drop=True)  
    df['about_product'] = df['about_product'].fillna('')  
  
    # -- 2. Label encoding --  
    le = LabelEncoder()  
    df['product_label'] = le.fit_transform(df['product_name'])  
    num_classes = len(le.classes_)  
  
    # -- 3. Tokenize text --  
    tokenizer = Tokenizer(num_words=max_num_words, oov_token='<OOV>')  
    tokenizer.fit_on_texts(df['about_product'])  
    sequences = tokenizer.texts_to_sequences(df['about_product'])  
    X = pad_sequences(sequences, maxlen=max_seq_len, padding='post')  
  
    # -- 4. Build one-hot targets --  
    y = to_categorical(df['product_label'], num_classes=num_classes)  
  
    # -- 5. 60-30-10 split --  
    X_temp, X_test, y_temp, y_test, df_temp, df_test = train_test_split(  
        X, y, df,  
        test_size=0.10,  
        random_state=42,  
        stratify=None # dont use this -> stratify=df['product_label'] some label only have one  
    )  
    val_ratio = 0.30 / 0.90  
    X_train, X_val, y_train, y_val, df_train, df_val = train_test_split(  
        X_temp, y_temp, df_temp,  
        test_size=val_ratio,  
        random_state=42,  
        stratify=None # dont use this -> stratify=df['product_label'] some label only have one  
    )
```

```
# -- 6. Build Keras model --  
model = Sequential([  
    Embedding(input_dim=max_num_words,  
              output_dim=embedding_dim,  
              input_length=max_seq_len),  
    LSTM(lstm_units),  
    Dense(256, activation='relu'),  
    Dense(128, activation='relu'),  
    Dense(64, activation='relu'),  
    Dense(32, activation='relu'),  
    Dense(num_classes, activation='softmax')  
)  
model.compile(  
    loss='categorical_crossentropy',  
    optimizer='adam',  
    metrics=['accuracy']  
)  
  
# -- 7. Train & Validation --  
history=model.fit(  
    X_train, y_train,  
    validation_data=(X_val, y_val),  
    epochs=epochs,  
    batch_size=batch_size  
)
```

```
# -- 8. Recommend for one user --  
# last history record  
user_hist = df[df['user_id_encoded'] == user_id_encoded]  
if user_hist.empty:  
    print("No purchase history found.")  
    return None  
  
last_text = user_hist['about_product'].iloc[-1]  
seq = tokenizer.texts_to_sequences([last_text])  
seq = pad_sequences(seq, maxlen=max_seq_len, padding='post')  
  
# predict  
probs = model.predict(seq)[0]  
# get rid of already purchased  
hist_labels = set(user_hist['product_label'])  
ranked = [i for i in probs.argsort()[::-1] if i not in hist_labels]  
top5 = ranked[:5]  
  
# back to the name  
rec_products = le.inverse_transform(top5)  
rec_scores = [probs[i] for i in top5]  
  
import pandas as pd  
results_df = pd.DataFrame({  
    'Id Encoded': [user_id_encoded] * len(top5),  
    'Recommended Product': rec_products,  
    'Recommendation Score': rec_scores  
})  
return results_df
```

Result:

Id	Encoded	Recommended Product	Recommendation Score
0	893	Eco Crystal J 5 inch Cartridge (Pack of 2)	0.112140
1	893	Luxor 5 Subject Single Ruled Notebook - A4, 70...	0.060781
2	893	Electvision Remote Control Compatible with Kod...	0.040835
3	893	Cello Quick Boil Popular Electric Kettle 1 Lit...	0.031952
4	893	Prestige Delight PRWO Electric Rice Cooker (1 ...	0.029248

The models can provide a table of products based on a user's history. The higher the recommendation score, the higher the likelihood that the user will purchase the product.

Strategic Insights:

- By removing fraudulent reviews, we improved the integrity and trustworthiness of the recommendation system.
- Dynamic pricing models have shown that ratings and discounting have a measurable impact on customer behavior, enabling more informed pricing strategies.
- Customer segmentation revealed clear clusters of product performance, enabling more targeted marketing and pricing actions.
- Combining TF-IDF and deep learning models for recommendations created a hybrid system that balances user behavior with product content, enhancing personalization.

Conclusion

The DeepRec project successfully designed a multi-layered recommendation system, addressing key business challenges in fraud detection, dynamic pricing, customer segmentation, and advanced recommendation modeling. Each module contributed to creating a more accurate, trustworthy, and user-focused system for product suggestions on an e-commerce platform.

Through rigorous data cleaning, clustering, regression modeling, and recommendation algorithms, the project improved the quality of product recommendations. It provided business stakeholders with actionable insights into pricing, customer targeting, and system integrity.

Business Impact:

- Improved customer trust by removing suspicious reviews, leading to more reliable recommendations.
- Enhanced revenue optimization by predicting discounted prices and segmenting high-value products.
- Increased personalization and customer satisfaction through hybrid recommendation models combining collaborative filtering, content analysis, and deep learning.
- Provided e-commerce stakeholders with insights to design pricing strategies and marketing campaigns better.

Future Directions:

- Implement real-time fraud detection to monitor review authenticity continuously.
- Introduce reinforcement learning to adjust recommendations based on user interactions and preferences dynamically.
- Further optimize dynamic pricing by incorporating competitor price tracking and seasonal demand factors to enhance pricing accuracy.
- Expand the hybrid recommendation engine to include visual (image-based) product similarity for even more personalized recommendations.

References

Amazon Product Reviews

Dataset originally curated by Flavio Quaresma, accessed via Kaggle: Amazon Product Analysis & Recommendation System.

K-Means Clustering Algorithm

MacQueen, J. (1967). "Some methods for classification and analysis of multivariate observations." Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability.

TF-IDF and Cosine Similarity

Manning, C. D., Raghavan, P., & Schütze, H. (2008). *Introduction to Information Retrieval*. Cambridge University Press.

Linear Regression Model

Freedman, D. A. (2009). *Statistical Models: Theory and Practice* (Revised Edition). Cambridge University Press.

Fraud Detection Heuristics in E-commerce

Dellarocas, C. (2006). "Strategic Manipulation of Internet Opinion Forums: Implications for Consumers and Firms." *Management Science*, 52(10), 1577–1593.

Deep Learning for Recommendations

Zhang, S., Yao, L., Sun, A., & Tay, Y. (2019). "Deep Learning Based Recommender System: A Survey and New Perspectives." *ACM Computing Surveys (CSUR)*, 52(1), 1–38.

