Abstract

This project utilizes how sentiment analysis can enhance product recommendations and sales insights within e-commerce platforms. It applies Natural Language Processing (NLP) techniques—including text preprocessing, feature engineering, and TF-IDF vectorization—to Amazon product reviews to develop a sentiment-driven recommendation system. Supervised machine learning algorithms, including Logistic Regression, Naïve Bayes, and Support Vector Machine (SVM), are used to classify user reviews into positive, negative, or neutral sentiments.

The technology provides important data, including sentiment distribution, price-performance analysis, rating trends, and review patterns across product categories on an interactive visualization dashboard, in addition to producing individualized product suggestions. This two-pronged strategy helps customers and organizations to make more informed decisions by combining structured product criteria with unstructured user feedback.

Experimental results demonstrate that Logistic Regression achieved the best performance with an accuracy of **90.15%** and an F1-score of **0.9008**, outperforming Naïve Bayes (70.27%, 0.6416) and SVM (87.71%, 0.8745). These results confirm that real-time visualization and interpretable machine learning models work well together to turn user-generated content into actionable intelligence for more intelligent e-commerce tactics.

Table of Contents

List of Figures 1

List of Tables 4

1. Introduction 5

2. Literature Review and Theoretical Background 7

2.1 Machine Learning Algorithm for Sentiment classification 7

2.1.1 Logistic Regression 7

2.1.2 Naïve Bayes 8

2.1.3 Support Vector Machine 8

2.2 Sentiment Analysis Techniques in E-commerce 8

2.2.1 Lexicon-based approaches 9

2.2.2 Machine learning-based sentiment analysis 9

2.2.3 Deep learning-based methods 9

2.3 Natural Language Processing in Text Analysis 9

2.4 Visualization and Dashboard Tools in Sentiment Applications 10

2.5 Research Gaps and Opportunities 10

3. Proposed system 11

4. Methodology 14

4.1 Data Description 14

4.2 Data Preprocessing 17

4.2.1 Cleaning the Reviews Dataset 18

4.2.2 Cleaning the Product Metadata Dataset 21

4.2.3 Merging Dataset 26

4.3 Sentiment Analysis 27

4.3.1 Sentiment Classification and Feature Engineering 27

4.3.2 Defining Sentiment Polarity Score 28

5. Implementation 30

5.1 Exploratory Data Analysis (EDA) 30

5.1.1 Data Visualization 33

5.2 Sales Insight Exploration 38

5.2.1 Correlation Analysis of KPIs 40

5.2.2 Top Performing Products by Sentiment and Rating 41

5.2.3 Sentiment vs. Price Relationship 43
5.2.4 Outlier Product Detection 44
5.2.5 Strategic Product Segmentation 47
5.3 Machine Learning Model Implementation 48
5.3.1 Logistic Regression Algorithm 49
5.3.2 Naïve Bayes Algorithm 49
5.3.3 Support Vector Machine 50
6 Experimentation 51
6.1 Model Performance of Machine Learning 51
6.2 Hyperparameter Tuning with GridSearchCV 52
6.2.1 Model Accuracy and F1 Score Comparison After Hyperparameter Tuning .. 54
6.3 Comparative Performance of Models 56
7 Visualization Dashboard 57
7.1 Dashboard Component 57
7.2 Technical Stack 58
7.3 Screenshot and Deployment 58
8 Future Directions and Recommendations 62
9 Conclusion 63
References 64

List of Figures

Figure 3-1: Workflow Architecture of the Sentiment-Driven Product Recommendation and Sales Insight System. 12
Figure 4-1: Upload dataset. 16
Figure 4-2: Getting the dimension of dataset. 17
Figure 4-3: Initial Exploration of Review and Metadata Datasets. 17
Figure 4-4: Detection and Visualization of Missing Values. 18
Figure 4-5: Test the number of duplicate entries. 18
Figure 4-6: Data handle by dropping unnecessary columns from reviews dataset. 19
Figure 4-7: Data handle by dropping missing reviews entries. 19
Figure 4-8: Data handle by removing duplicate reviews rows. 19
Figure 4-9: Checking reviews dataset after cleaning. 19
Figure 4-10: Show length of rows and columns for reviews dataset after cleaning. 20
Figure 4-11: Checking column datatypes. 20
Figure 4-12: Cleaning the reviews text. 20
Figure 4-13: Show visualisation heatmap for reviews dataset after cleaning. 21
Figure 4-14: Save cleaned reviews dataset as csv file after cleaning. 21
Figure 4-15: Data Structure for Product Metadata Dataset. 22
Figure 4-16: Checking missing values. 22
Figure 4-17: Checking duplicate records. 22
Figure 4-18: Data handling by dropping unnecessary columns from metadata dataset. 23
Figure 4-19: Data cleaning for detail column of product metadata. 23
Figure 4-20: Handling Missing Values in Product Metadata Dataset. 23
Figure 4-21: Removing Duplicate Rows from Product Metadata Dataset. 23
Figure 4-22: Convert datatype for product metadata dataset. 24
Figure 4-23: Filtering and Renaming Columns in the Product Metadata Dataset. 24
Figure 4-24: Final Verification of Cleaned Product Metadata Dataset. 24
Figure 4-25: Show length of rows and columns from product metadata after cleaning. 25
Figure 4-26: Check datatypes for product metadata. 25
Figure 4-27: Save cleaned product metadata dataset as csv file after cleaning. 25
Figure 4-28: Merging Cleaned Reviews and Metadata Datasets. 26
Figure 4-29: Verifying Row and Column Count After Merging Datasets. 26
Figure 4-30: Sentiment Classification Using TextBlob. 28
Figure 4-31: Sentiment Label Preview and Distribution in the Dataset. 28
Figure 4-32: Extracting Polarity Scores from Cleaned Review Text. 29
Figure 4-33: Preview of Cleaned Review Text with Sentiment Labels and Polarity Scores. . 29
Figure 4-34: Saving the Final Merged Dataset with Sentiment and Polarity. 29
Figure 5-1: Import Require Libraries. 30
Figure 5-2: Summary of Dataset Structure and Null Value Check. 31
Figure 5-3: Heatmap Visualization of Missing Values in the Dataset. 31

Figure 5-4: Correlation Matrix of Price, Rating Number, Average Rating, and Polarity. 32
Figure 5-5: Heatmap Visualization of Correlation Among Key Numerical Features. 32
Figure 5-6: Distribution of Ratings. 33
Figure 5-7: Sentiment Label Distribution. 34
Figure 5-8: Average Rating per Sentiment. 34
Figure 5-9: Sentiment Polarity vs Product Price. 35
Figure 5-10: Number of Ratings per Sentiment Group. 35
Figure 5-11: Top 10 Most Positive Products. 36
Figure 5-12: Top 10 Most Negative Products. 36
Figure 5-13: Top 10 Most Neutral Products. 37
Figure 5-14: Sentiment Distribution Pie Chart. 37
Figure 5-15: Sentiment Count by Product Category. 38
Figure 5-16: Dataset Loading and Structure Overview. 39
Figure 5-17: Product-Level KPI Aggregation. 39
Figure 5-18: Category-Level KPI Aggregation. 40
Figure 5-19: Correlation Matrix of Key Performance Indicators (KPIs). 40
Figure 5-20: Correlation Matrix of Price, Rating, Polarity, and Number of Reviews. 41
Figure 5-21: Top 10 Products by Sentiment Polarity. 42
Figure 5-22: Top 10 Products by Customer Rating. 42
Figure 5-23: Top 10 Sentiment Products — Comparison of Rating vs. Polarity. 43
Figure 5-24: Price vs Sentiment Polarity at Product Level. 43
Figure 5-25: List of Weak Products. 44
Figure 5-26: Number of Reviews vs Average Sentiment Polarity. 45
Figure 5-27: Top 10 Products by Review Volume. 46
Figure 5-28: Sentiment Polarity vs. Number of Reviews (Sales Proxy). 46
Figure 5-29: Identification of Hidden Gem Products Using Price, Rating, and Sentiment Polarity Filters. 47
Figure 5-30: Identification of Overpriced Low-Quality Products Based on Price, Rating, and Negative Sentiment. 47
Figure 5-31: Text Cleaning and Sentiment Label Preparation. 48
Figure 5-32: TF-IDF Vectorization. 48
Figure 5-33: Train-Test Data Split and Distribution. 48
Figure 5-34: Logistic Regression Classification Report and Confusion Matrix. 49
Figure 5-35: Naïve Bayes Classification Report and Confusion Matrix. 50
Figure 5-36: Support Vector Machine (SVM) Classification Report and Confusion Matrix. 50
Figure 5-37: Model Serialization using joblib. 51
Figure 6-1: Model Evaluation Metrics Table (Accuracy & F1 Score). 51
Figure 6-2: Bar Chart Comparing Model Accuracy. 52
Figure 6-3: Bar Chart Comparing Model F1 Score. 52
Figure 6-4: Classification Report – Tuned Logistic Regression. 53
Figure 6-5: Classification Report – Tuned Naive Bayes. 53

Figure 6-6: Classification Report – Tuned SVM. 54
Figure 6-7: Model Evaluation Metrics Table after Tuning (Accuracy & F1 Score). 54
Figure 6-8: Bar Plot of Model Accuracy Scores Comparison After Hyperparameter Tuning. 55
Figure 6-9: Bar Plot of Model F1 Scores Comparison After Hyperparameter Tuning..... 55
Figure 6-10: Model Accuracy and F1 Score Comparison Before and after Hyperparameter Tuning. 56
Figure 7-1: Dashboard Top Section — Product Search, KPI Tiles, Sentiment and Rating Charts. 59
Figure 7-2: Dashboard Bottom Section — Helpful Votes, Price Trends, Store Analytics. 59
Figure 7-3: Interactive Product Search Interface with KPI Summary and Quick Access to Most Reviewed Product..... 60
Figure 7-4: Detailed Sentiment View of Selected Product with Verified Purchase Status, User Review, and Descriptive Metadata..... 60
Figure 7-5: Most Reviewed Products — Ratings, Sentiment Scores, and Review Volumes Table. 61

List of Tables

Table 4-1: Review Dataset Fields 15
Table 4-2: Metadata Dataset Fields 16
Table 7-1: Summary of Technologies Used in the Sentiment-Driven Product Recommendation System 58

1. Introduction

E-commerce platforms' rapid expansion has increased the importance of sentiment research in comprehending consumer behaviour. The purpose of this research project is to examine how sentiment analysis, specifically a subset of natural language processing (NLP), can be used to improve product recommendations and generate business insights from customer-written reviews. Platforms may better comprehend user input and enhance product development, marketing tactics, and recommendation systems by assessing customer sentiment, whether it be positive, negative, or neutral (Hake et al., 2025). This project suggests creating a visual sales insight dashboard to assist a sentiment-driven product recommendation engine.

To evaluate text reviews and provide sentiment scores, the system uses natural language processing (S and Ramathmika, 2019). High-performing products are identified, and product recommendations are made more personalized by combining these sentiment scores with product reviews and other metadata.

To build this system, several supervised learning algorithms—such as Logistic Regression, Naive Bayes, and Support Vector Machine (SVM) these are implemented to evaluate product review data, predict sentiment polarity, and enhance the recommendation process. Evaluation metrics including accuracy and F1-score are used to compare model performance. Sentiment analysis, machine learning, and visualization are combined to help e-commerce companies and customers make better decisions by bridging the gap between unstructured review data and actionable product suggestions.

The aim of this project is to examine the application of sentiment analysis in review evaluations to make product recommendations and generate actionable insights to identify which product category has the highest performance.

Objectives

- To research how machine learning algorithms are applied in recommendation systems for e-commerce platforms.
- To conduct a role of sentiment analysis in enhancing e-commerce product recommendation systems.
- To calculate the average rating for each product to determine which categories receive the highest ratings and to evaluate review data by analysing textual sentiments
- To find the correlation among various features to assess whether changes in product prices affect product ratings.

- To perform sentiment analysis to determine the count and percentage of positive, negative, and neutral sentiments, and visualize sentiment and sales insights through an interactive JavaScript dashboard.
- To analyse the performance of the implemented machine learning models using appropriate evaluation metrics to determine the most effective algorithm for sentiment classification.
- To extract sales insights from structured review data (price, rating, discount).
- To reflect on the project outcomes and future opportunities for extending the system, including potential integration with real-time recommendation engines.

This objective will be achieved by applying machine learning strategies of exceptionally high effectiveness.

Research Questions

The system is developed, implemented, and evaluated in response to the following research questions:

1. How can sentiment analysis of customer product reviews be used to generate accurate and personalized product recommendations in an e-commerce environment?
2. Which machine learning models are the most effective for classifying sentiment in customer reviews for the purpose of recommendation and insight generation?
3. What actionable sales insights can be extracted by combining sentiment scores with other review features such as ratings, price, and helpfulness votes?
4. How can an interactive dashboard be designed to effectively visualize sentiment trends and product performance for end users and business stakeholders?
5. To what extent can sentiment-based review analysis support more informed decision-making for both consumers and online retailers?
6. How does sentiment distribution vary across different product categories, and what does this reveal about customer preferences?
7. How do different machine learning models perform under varying levels of review text complexity and feature sets?
8. Which features—such as sentiment polarity, price, or verified purchase—have the strongest impact on product recommendation accuracy?
9. To what extent does a sentiment-driven dashboard improve product discoverability and sales strategy insights for stakeholders?

2. Literature Review and Theoretical Background

This literature review focuses on the key techniques and technologies relevant to developing a sentiment-driven product recommendation and sales insight system for e-commerce. Emphasizing the efficacy and efficiency of algorithms such as Logistic Regression, Naive Bayes and Support Vector Machine (SVM) in categorizing customer evaluations, the study primarily examines machine learning-based sentiment analysis techniques (Shipra Saraswat et al., 2023). Basic natural language processing (NLP) preprocessing techniques that prepare text data for analysis and interpretation include lemmatization, tokenization, stop word removal, and TF-IDF vectorization. The study also looks at interactive dashboard frameworks and visualization tools for clearly and easily displaying analytical results. The choice of interpretable models and interactive visualizations for creating a successful, scalable recommendation system is supported by this review's critical analysis of present systems and identification of research issues.

2.1 Machine Learning Algorithm for Sentiment classification

Machine Learning (ML) is a subset of artificial intelligence (AI) that enables systems to automatically learn from data, identify patterns, and make decisions or predictions without being explicitly programmed (Overview of Machine Learning, 2024). In the context of this project, ML is used to improve the accuracy and relevance of product recommendations and to analyse consumer sentiment for better business insights. Through machine learning, the computers will learn to do intelligent jobs such as predicting for the future or decision making. Machine learning algorithms are mainly categorized into four types: supervised learning, unsupervised learning, and reinforcement learning (Gürcan, 2018).

This study implemented and evaluated several supervised ML algorithms, including Logistic Regression, Naïve Bayes and Support Vector Machine (SVM) (Venkatesh and Ranjitha, 2018). Each algorithm offers unique advantages and is selected based on its suitability for text-based sentiment analysis tasks.

2.1.1 Logistic Regression

A linear model for classifying data into binary or multiclass categories is called logistic regression. Using the sigmoid function, it calculates the likelihood that a given input is a member of a particular class. Its efficacy on large, sparse datasets makes it a good combination with TF-IDF vectorization for text categorization. Its computational efficiency and interpretability are excellent.

- **Advantages**: Simple, fast to train, suitable for linearly separable data.
- **Limitations**: Cannot capture complex relationships without feature engineering or polynomial terms.

2.1.2 Naïve Bayes

Naïve Bayes classifiers are based on Bayes' Theorem and assume independence among features. They work particularly well for text classification because of the natural structure of word frequencies in documents. Naïve Bayes is fast, scalable, and effective with limited data.

- **Advantages**: High speed, effective for small datasets, easy to implement.
- **Limitations**: Assumption of feature independence can reduce accuracy in real-world text.

2.1.3 Support Vector Machine

A strong classifier, the Support Vector Machine (SVM) creates hyperplanes to divide data into discrete classes. It works well in high-dimensional spaces, which makes it appropriate for text categorization in situations with sparse feature spaces.

- **Advantages**: High accuracy works well in high-dimensional settings, robust to overfitting.
- **Limitations**: Training can be slow on large datasets, and results are less interpretable.

These models evaluated the accuracy and F1-score to identify the best-performing algorithm for sentiment classification. By comparing many models, the project guaranteed forecast resilience and clarified the trade-offs between accuracy, interpretability, and complexity.

2.2 Sentiment Analysis Techniques in E-commerce

One area of natural language processing (NLP) that deals with identifying the emotional tone of words is sentiment analysis, often known as opinion mining. It is essential for figuring out how customers feel about things in the context of e-commerce. Businesses can decide on products, marketing, and consumer engagement tactics with knowledge whether reviews are categorized as positive, negative, or neutral (Morton et al., 2014). Methods based on machine learning that train models on labelled data provide for greater flexibility and precision. This field has made extensive use of supervised machine learning methods such as Logistic

Regression, Naïve Bayes, and Support Vector Machine (SVM). Deep learning models such as Recurrent Neural Networks (RNNs) and transformers like BERT are recent innovations that improve contextual comprehension but require a large amount of computing power.

2.2.1 Lexicon-based approaches

Lexicon-based approaches use predefined lists of words annotated with sentiment scores. Tools like VADER and TextBlob are popular due to their ease of use and ability to handle simple polarity detection. However, they may struggle with contextual nuances, sarcasm, and domain-specific language. These methods are often used in real-time applications due to their speed.

2.2.2 Machine learning-based sentiment analysis

Machine learning-based sentiment analysis involves training classifiers on labelled review datasets. Algorithms like Naïve Bayes, Logistic Regression, and SVM learn patterns from data and provide higher accuracy than lexicon methods when sufficient data is available. These methods require feature engineering (e.g., TF-IDF) and preprocessing to convert text into a usable format (Al-Sulaiti et al., 2021).

2.2.3 Deep learning-based methods

Deep learning-based methods such as Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and transformer models like BERT offer even greater contextual understanding by learning semantic relationships. These models can handle complex language structures but are computationally expensive and require large datasets libraries (Siham Bouguern and Mouna Aizi, 2024).

In this project, machine learning-based sentiment analysis is used due to its balance between performance, interpretability, and computational cost. This approach enables the system to classify reviews accurately and provide meaningful recommendations and insights .

2.3 Natural Language Processing in Text Analysis

Preprocessing is an essential step for text-based machine learning systems that converts unstructured textual input into a format suitable for model training. Tokenization divides the text into discrete words or tokens, usually starts the process. The next step is stop word removal, which involves eliminating common but uninformative words, such as "and," "the," and "is," to reduce noise (Gupta et al., 2022). The dataset is then made more consistent by

applying lemmatization or stemming, which reduces words to their base or root forms. The analysed text was transformed into numerical characteristics using TF-IDF vectorization, which assigns weights to terms based on their significance across the entire corpus of documents. These combined steps standardize the text data, reduce dimensionality, and enhance the overall efficiency and effectiveness of the machine learning models (Shin, Na and Elmqvist, 2025).

2.4 Visualization and Dashboard Tools in Sentiment Applications

Visual analytics enables stakeholders to quickly interpret complex data. Tools like Tableau and Power BI offer drag-and-drop interfaces but have limited customization for real-time, sentiment-based recommendation systems (Few, 2006; Tableau, 2021). JavaScript libraries like D3.js and Chart.js enable the creation of more flexible and dynamic dashboards (Shin, Na and Elmqvist, 2025). In this project, JavaScript was used to build interactive visualizations that display review sentiment, product ratings, and pricing trends, helping users explore sales insights visually.

2.5 Research Gaps and Opportunities

Although sentiment analysis and machine learning-based review classification have been widely studied in the context of e-commerce, several critical limitations remain unaddressed. Most existing systems prioritize classification accuracy but overlook the importance of delivering insights in an interpretable and actionable format. Many utilize intricate deep learning models, which are effective but opaque, making them inappropriate for users or companies that need to be able to explain their decision-making processes. Moreover, sentiment analysis and interactive visualization dashboards have little in common. The applicability of current systems for dynamic, data-driven product recommendation strategies is limited by their inability to investigate product sentiment in real-time or through user-driven methods. Additionally, it is not common practice to identify product categories based on their highest performance or to inform product suggestions by evaluating information such as sentiment scores, review counts, and average ratings.

These gaps present opportunities for improvement. Interpretable machine learning models and interactive JavaScript-based dashboards (such as those created using Chart.js or D3.js) can be utilized to develop a system that offers both visual clarity and analytical accuracy.

Incorporating review numbers into recommendation algorithms also provides end users with more detailed and tailored insights. To enhance e-commerce decision-making, this project develops a sentiment-driven recommendation system that integrates machine learning categorization with a user-friendly, dynamic interface.

Our literature review, which was pertinent to our research, examined key techniques in sentiment analysis, machine learning algorithms, and NLP preprocessing. In addition to ML-based sentiment classification using SVM, Naïve Bayes, and Logistic Regression, the review endorsed the use of TF-IDF feature extraction. It was also emphasized how crucial data visualization is for offering insightful information. In addition to aligning with project goals, the selected techniques provide a solid foundation for creating a sentiment-driven recommendation and sales intelligence system that is effective.

3. Proposed system

This research project is to assess the application of sentiment analysis in customer-written reviews to provide product suggestions on e-commerce platforms (Jeidy Panduro-Ramirez, 2024). Through the utilization of sentiment analysis, which is a subset of natural language processing (NLP), e-commerce platforms can extract substantial insights from the textual comments of customers, identify areas in which further development is required, and modify product proposals better to meet the requirements of their customers (Jayakody and Kumara, 2021). This project will develop a sentiment-driven product recommendation and sales insight for an e-commerce system. The collected insights will be used to categorize product reviews as positive, negative, or neutral by using machine learning algorithms. In order to improve recommendation algorithms and find high-performing products, the sentiment scores that are produced will be merged with product ratings and categories. Furthermore, this study will employ several machine learning techniques for sentiment categorization and evaluate their effectiveness to identify the most effective models.

This project will focus on the dataset provided by Amazon's product reviews for businesses. A Natural Language processing library in Python is used to fetch the algorithm. To perform sentiment analysis of product reviews obtained from Amazon's product dataset, the reviews are classified as having positive, negative, or neutral sentiment using Supervised Learning techniques, which are primarily used for product classification and sentiment-

enhanced recommendations. Structured data was used to train these algorithms, which mapped inputs such as sentiment polarity, average rating, and price to known outputs, including product popularity and suggestion ranking.

The proposed approach utilizes sentiment analysis based on natural language processing and machine learning to enhance product suggestions and provide valuable sales statistics. Figure 3-1 illustrates the system's architecture.

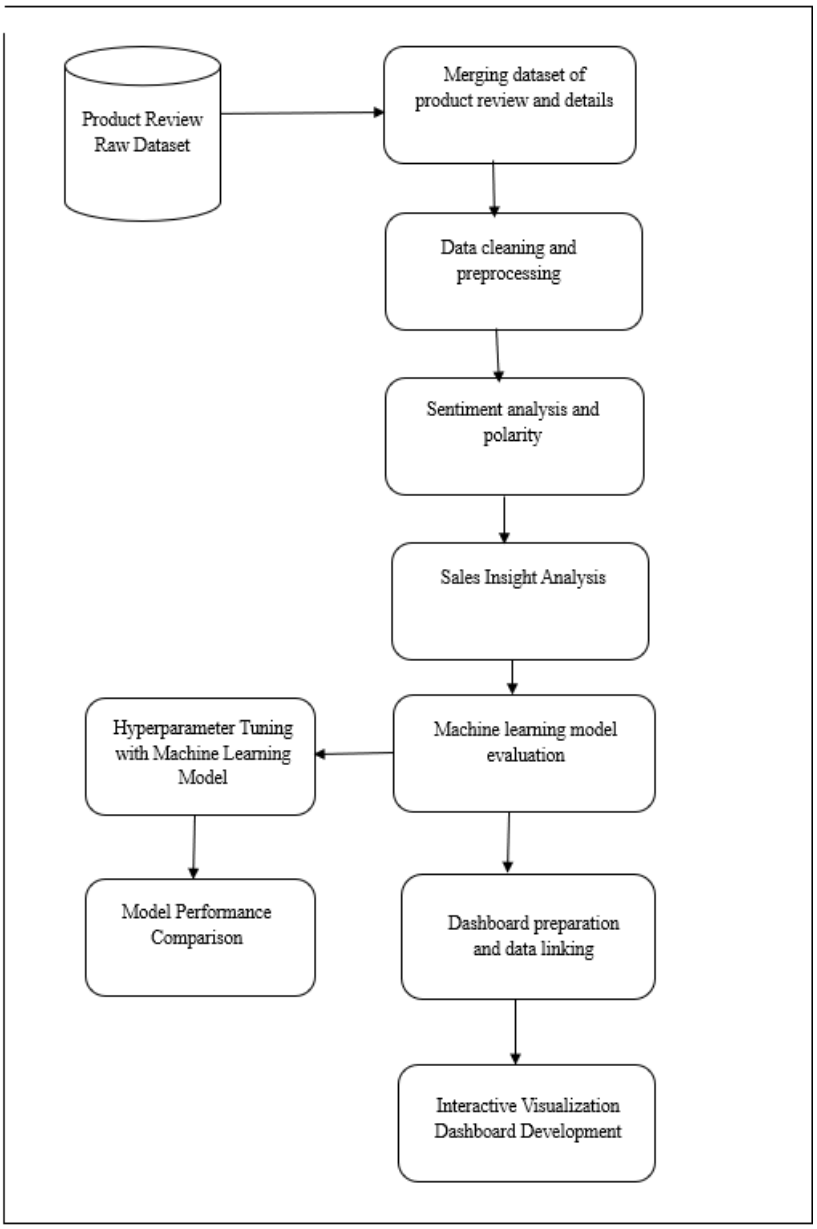

Figure 3-1: Workflow Architecture of the Sentiment-Driven Product Recommendation and Sales Insight System.

The core components and workflow of the system are as follows:

1. **Data Acquisition and Integration**

   The system uses product metadata and user reviews of each product from Amazon product review databases. When we analyse these datasets, this unstructured information is combined to create a single structure that aligns product details with reviews.

2. **Data Preprocessing and Text Cleaning**

   The datasets are carefully pre-processed before the model is developed. This includes handling missing data, eliminating stop words, converting text to lowercase, filtering out noise and punctuation, and applying normalization and tokenization procedures. The review texts are made clean, consistent, and prepared for sentiment analysis in this step.

3. **Sentiment Analysis and Labelling**

   A natural language processing program called TextBlob is used to analyse the sentiment polarity of each review. The polarity ratings are used to categorize reviews into three sentiment categories: positive, neutral, and negative. Subjective textual feedback is converted into measurable data points through this classification.

4. **Sales Insight Extraction**

   Sentiment scores are aggregated with product ratings, review counts, average rating and price data to uncover patterns such as top-rated or highly reviewed products, sentiment trends, and customer satisfaction.

5. **Machine Learning Model Development**

   Supervised machine learning models, including Support Vector Machine (SVM), Naïve Bayes, and Logistic Regression, are trained to automatically classify sentiment from text, thereby improving the system's recommendation capabilities. The review texts' vectorized TF-IDF (Term Frequency–Inverse Document Frequency) characteristics are used in these models.

6. Hyperparameter Tuning and Model Evaluation

   To optimize performance, GridSearchCV is used to adjust model parameters. The efficacy of the models is assessed using their accuracy and F1-score. Logistic Regression outperformed SVM and Naïve Bayes, according to experimental data, with an accuracy of 90.15% and an F1-score of 0.9008, outperforming both.

7. Interactive Visualization Dashboard Development

   The research's final product is an interactive dashboard created. This dashboard provides real-time visualization of sentiment distribution, product popularity, price-performance comparison, and ranking information. Users can view dynamic charts, apply filters, and analyse sentiment at the product level to inform data-driven decision-making.

This pipeline ensures a complete system from raw data ingestion to insightful, interactive visualization, offering both technical rigor and practical usability for e-commerce decision-making (Adarsh, Sreedevi and Deepa, 2023).

4. Methodology

This methodology utilizes machine learning and sentiment analysis to provide a thorough procedure for analysing consumer feedback and making informed product recommendations. In our approach, sentiment data is extracted from written customer reviews, and each review is classified as either positive, neutral, or negative, and the polarity of the sentiment is measured. Machine learning models designed to produce product recommendations are then updated to incorporate these properties (R.V. Karthik, Ganapathy, and Kannan, 2018). Delivering intelligent recommendations that improve the e-commerce decision-making process and customer experience is made possible by this methodology.

4.1 Data Description

High-quality and rich datasets are critical to the efficacy of this sentiment-driven recommendation system. Due to its extensive information, wide variety of products, and wealth of consumer reviews, an Amazon e-commerce product review dataset was selected for this study. This dataset provides both structured and unstructured data, making it suitable for sentiment analysis, machine learning, and business intelligence applications.

The Amazon Reviews 2023 repository run by UC San Diego provided the dataset used in this research. The Health and Personal Care category was selected due to its high level of consumer interaction and applicability to tailored product suggestions. This dataset comprises two files of Amazon Product Reviews and metadata, as illustrated in the Table (4-1 and 4-2), as per the details listed on Amazon's official website for the Amazon Reviews ’23. Data source: Introduction - Amazon Reviews'23

Field	Type	Explanation
rating	float	Rating of the product (from 1.0 to 5.0).
title	str	Title of the user review.
text	str	Text body of the user review.
asin	str	ID of the product.
parent_asin	str	Parent ID of the product.
user_id	str	ID of the reviewer
verified_purchase	boolean	User purchase verification
helpful_vote	int	Helpful votes of the review

Table 4-1: Review Dataset Fields

Field	Type	Explanation
main_category	str	Main category (i.e., domain) of the product.
title	str	Name of the product.
average_rating	float	Rating of the product shown on the product page.
rating_number	int	Number of ratings in the product.
description	list	Description of the product.
price	float	Price in US dollars (at time of crawling).
store	str	Store name of the product.
categories	list	Hierarchical categories of the product.
details	dict	Product details, including materials, brand, sizes, etc.
parent_asin	str	Parent ID of the product.

Table 4-2: Metadata Dataset Fields

In this step, the dataset is imported from the data folder. This folder contains two dataset files like review_Health_and_Personal_Care.csv and meta_Health_and_Personal_Care.csv.

```python
# Load product reviews dataset file
reviews = pd.read_csv('data/raw/review_Health_and_Personal_Care.csv')

# Load product metadata dataset file
metadata = pd.read_csv('data/raw/meta_Health_and_Personal_Care.csv')
```

Figure 4-1: Upload dataset.

```python
# Get the dimension of the data for review dataset
review = reviews.shape

print(f"Number of rows for reviews dataset: {review[0]} \n Number of columns for reviews dataset: {review[1]} ")

# Get the dimension of the data for metadata dataset
meta = metadata.shape

print(f"Number of rows for metadata dataset: {meta[0]} \n Number of columns for metadata dataset: {meta[1]} ")
```

✓ 0.0s Python

```
Number of rows for reviews dataset: 494121
 Number of columns for reviews dataset: 10
Number of rows for metadata dataset: 60293
 Number of columns for metadata dataset: 14
```

Figure 4-2: Getting the dimension of dataset.

This project utilized Amazon's product reviews and metadata dataset, which consists of rows and columns as shown in Figure 4-2 before data cleaning. The column names and column values are shown in the first few rows using the head() method. Then, the data structure and types are checked using the info() method.

```python
# explore the datasets

# View the first few rows
reviews.head()
# Check the structure and data types
reviews.info()

# View the first few rows
metadata.head()
# Check the structure and data types
metadata.info()

reviews['rating'].value_counts()
```

✓ 0.3s Python

Figure 4-3: Initial Exploration of Review and Metadata Datasets.

4.2 Data Preprocessing

To prepare raw data for analysis and modelling, data preprocessing is a crucial step. Both the review and metadata datasets underwent the preparation steps. The following steps are summarized: renaming fields, handling missing values, eliminating duplicates, normalizing text, reducing the number of columns, and converting data types. The datasets were made clean, consistent, and appropriate for creating a sentiment-driven recommendation system thanks to these preparation procedures.

Make sure there are no duplicate rows or missing values in the dataset before modelling it. Missing and duplicate data may result in biased model training and impact evaluations, as some data items are overrepresented. A methodical methodology compares all feature values across rows to find and eliminate duplicates. Duplicate entries and missing records are removed to prevent redundancy and guarantee data integrity.

4.2.1 Cleaning the Reviews Dataset

4.2.1.1 Handling Missing and Duplicate Values

The reviews dataset was initially examined for duplicates and missing entries to guarantee data quality. The following things were done:

Only the title and text columns had null values, as indicated by a missing value check.

- To visually identify missing data points, a heatmap was created using the Seaborn library.
- Rows with missing titles or text were removed since these are essential for sentiment analysis.
- Duplicate rows were detected and removed using pandas. duplicated() function.

```python
# Check for missing values
print("Missing values per column:\n", reviews.isnull().sum())



# Visualize missing values
sns.heatmap(reviews.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```

✓ 4.9s Python

```
Missing values per column:
 rating                 0
title                125
text                 141
images                 0
asin                   0
parent_asin            0
user_id                0
timestamp              0
helpful_vote           0
verified_purchase      0
dtype: int64
```

Figure 4-4: Detection and Visualization of Missing Values.

```python
# Check duplicated row
# Identify duplicate rows
duplicates = reviews[reviews.duplicated()]

# Display the duplicate rows
print(duplicates)

# Count the number of duplicate rows
num_duplicates = reviews.duplicated().sum()
print(f"Number of duplicate rows: {num_duplicates}")
```

Python

Figure 4-5: Test the number of duplicate entries.

Columns such as images and timestamp were dropped because they were either irrelevant or not used in the current stage of analysis.

```python
# Drop unnecessary columns with any missing value
reviews = reviews.drop(columns=["images", "timestamp"])  # 'images' is empty, 'timestamp' not used for now
print(f"Well done, dropping for unnecessary columns")
```
✓ 0.1s Python

Figure 4-6: Data handle by dropping unnecessary columns from reviews dataset.

```python
# Drop Missing reviews
reviews.dropna(subset=['text'], inplace=True)

# Drop misssing title
reviews.dropna(subset=['title'], inplace=True)

print(f"Missing value handling is complete.")
```
✓ 0.2s Python

Figure 4-7: Data handle by dropping missing reviews entries.

```python
# Remove duplicate rows
cleaned_reviews_df = reviews.drop_duplicates()
print(f"Removing duplicate value is complete.")
```
✓ 1.2s Python

Figure 4-8: Data handle by removing duplicate reviews rows.

```python
# Double-check
print(f"Missing values now: {cleaned_reviews_df.isnull().sum()}")
print(f"Number of duplicate rows now: {cleaned_reviews_df.duplicated().sum()}")
```
✓ 1.2s Python

```
Missing values now: rating              0
title                0
text                 0
asin                 0
parent_asin          0
user_id              0
helpful_vote         0
verified_purchase    0
dtype: int64
Number of duplicate rows now: 0
```

Figure 4-9: Checking reviews dataset after cleaning.

The structure of the reviews dataset was carefully verified to guarantee consistency and preparedness for further processing following the completion of the first data cleaning stages. We double-checked the row and column counts to be sure that no important data had been inadvertently lost during the cleaning process. Next, we looked at the data type of each column to make sure that values were categorized correctly, such as objects for textual data and integers for numerical ratings. Avoiding data type inconsistencies during analysis or model training requires this verification step. A heatmap was also created one last time to visually verify that there were no more missing values in the dataset. After the integrity of the cleaned data was confirmed, the dataset was exported and saved in CSV format. Accessing and reusing it for subsequent tasks like feature extraction, model construction, and visualization is made simpler at this level.

```python
# show length of rows and columns from dataset after cleaning
print(f"Rows length: {len(cleaned_reviews_df)}")
print(f"Columns length: {cleaned_reviews_df.columns}")
```
✓ 0.0s Python

```
Rows length: 488776
Columns length: Index(['rating', 'title', 'text', 'asin', 'parent_asin', 'user_id',
       'helpful_vote', 'verified_purchase'],
      dtype='object')
```

Figure 4-10: Show length of rows and columns for reviews dataset after cleaning.

```python
# Check column data types
print(cleaned_reviews_df.dtypes)
```
✓ 0.0s Python

```
rating                int64
title                object
text                 object
asin                 object
parent_asin          object
user_id              object
helpful_vote          int64
verified_purchase      bool
dtype: object
```

Figure 4-11: Checking column datatypes.

To get the review texts ready for sentiment analysis, the raw textual data underwent a thorough preparation step. The review text was systematically cleaned up using a specially written Python script. During analysis, this function first changes all text to lowercase to ensure uniformity and remove case sensitivity issues. It then eliminated all punctuation, special symbols, and numeric digits that would have introduced noise into the model or impacted the precision of text-based calculations using regular expressions. This cleaning process improved the reviews' applicability for further sentiment analysis tasks, including feature-based product rating, keyword extraction, and polarity score, by standardizing the reviews.

```python
# Cleaning the reviews text
import re

def clean_text(text):
    text = str(text).lower()
    text = re.sub(r"[^a-z0-9\s]", "", text)  # remove punctuation and symbols
    return text

cleaned_reviews_df['cleaned_text'] = cleaned_reviews_df['text'].apply(clean_text)
```
✓ 3.4s Python

Figure 4-12: Cleaning the reviews text.

```python
#Checking for null values
cleaned_reviews_df.isnull().sum()

# Visualize missing values
sns.heatmap(reviews.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```

✓ 3.9s Python

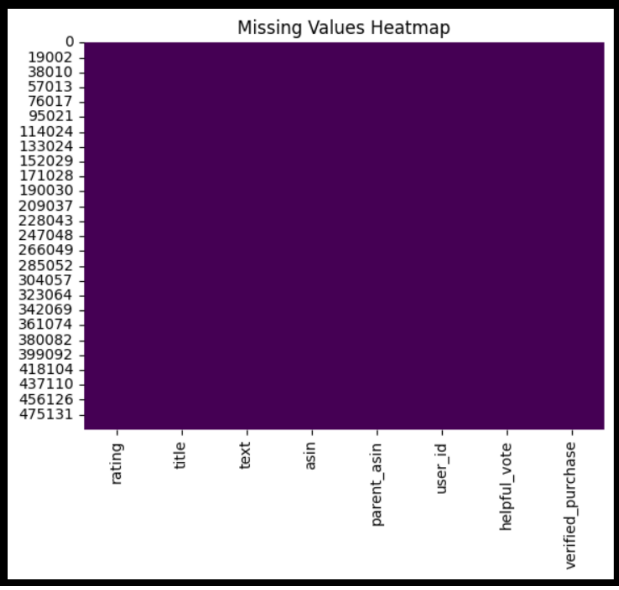

Figure 4-13: Show visualisation heatmap for reviews dataset after cleaning.

```python
# save cleaned review dataset
reviews.to_csv("data/cleaned_reviews.csv", index=False)
```

✓ 4.9s Python

Figure 4-14: Save cleaned reviews dataset as csv file after cleaning.

The last step was to preserve the cleaned data for later use, once all preprocessing and data cleaning procedures on the reviews dataset were complete. This was accomplished by exporting the Data Frame to a CSV file using Pandas' to_csv() function. The file, named cleaned_reviews.csv, was saved in the specified directory.

4.2.2 Cleaning the Product Metadata Dataset

The data cleaning techniques applied to the product metadata dataset are described in this section to ensure its suitability for trustworthy analysis and modeling.

The Pandas library's metadata.info() function was used to obtain an overview of the dataset's structure. Important metadata, such as the total number of entries, column names, data types,

and the number of non-null values in each column, were supplied by this function.

```python
# Check structure
metadata.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 60293 entries, 0 to 60292
Data columns (total 14 columns):
 #   Column           Non-Null Count  Dtype
---  ------           --------------  -----
 0   main_category    60293 non-null  object
 1   title            60288 non-null  object
 2   average_rating   60293 non-null  float64
 3   rating_number    60293 non-null  int64
 4   features         60293 non-null  object
 5   description      60293 non-null  object
 6   price            10536 non-null  float64
 7   images           60293 non-null  object
 8   videos           60293 non-null  object
 9   store            57938 non-null  object
 10  categories       60293 non-null  object
 11  details          60293 non-null  object
 12  parent_asin      60293 non-null  object
 13  bought_together  0 non-null      float64
dtypes: float64(3), int64(1), object(10)
memory usage: 6.4+ MB
```

Figure 4-15: Data Structure for Product Metadata Dataset.

4.2.2.1 Data Cleaning and Transformation

This process began by identifying and visualizing missing values and duplicate entries using both Pandas and Seaborn. This duplicate and missing records were detected and quantified using the duplicated() and isnull() functions, providing insight into potential redundancy in the dataset.

```python
# Check for missing values
metadata.isnull().sum()
print("Missing values per column:\n", metadata.isnull().sum())

# Visualize missing values
sns.heatmap(metadata.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap for metadata")
plt.show()
```

✓ 1.1s Python

Figure 4-16: Checking missing values.

```python
# Check duplicated row
# Identify duplicate rows
duplicates = metadata[metadata.duplicated()]

# Display the duplicate rows
print(duplicates)

# Count the number of duplicate rows
num_duplicates = metadata.duplicated().sum()
print(f"Number of duplicate rows: {num_duplicates}")
```

✓ 1.1s Python

Figure 4-17: Checking duplicate records.

```python
# Drop unnecessary column for metadata
metadata_df = metadata.drop(columns=[
    'features', 'description', 'images', 'videos', 'categories', 'bought_together'
])

print(f"Dropping unnecessary columns is complete.")
```

✓ 0.0s Python

Figure 4-18: Data handling by dropping unnecessary columns from metadata dataset.

Regular expressions were used to eliminate extraneous characters from the details column in preparation for further examination.

```python
import re

# Strip {, }, and keys
def clean_details(text):
    return re.sub(r"[{}']", "", re.sub(r"\s*\w+:\s*", "", str(text)))

metadata_df['details'] = metadata_df['details'].apply(clean_details)
```

✓ 2.1s Python

Figure 4-19: Data cleaning for detail column of product metadata.

To retain important data, further data handling procedures included eliminating rows with missing entries and using the column median to fill in missing price values. 'Unknown' was also used as a placeholder string in the shop field to ensure consistency. Finally, all remaining duplicate rows were removed, resulting in a cleaner and more structured dataset ready for downstream tasks such as analysis, modelling, or integration.

```python
# Drop and Fill for missing value data
# Drop rows where essential info is missing
metadata_df = metadata_df.dropna(subset=['title'])

# Fill missing price with median (to avoid dropping useful rows)
metadata_df['price'] = metadata_df['price'].fillna(metadata_df['price'].median())

# Optional: fill missing store or title with placeholder
metadata_df['store'] = metadata_df['store'].fillna('Unknown')

print(f"Missing value handling complete: kept all rows with filled price and store.")
```

✓ 0.0s Python

Figure 4-20: Handling Missing Values in Product Metadata Dataset.

```python
# Remove duplicate rows
metadata_df = metadata_df.drop_duplicates()
print(f"Removing duplicate value is complete.")
```

✓ 0.1s Python

Figure 4-21: Removing Duplicate Rows from Product Metadata Dataset.

4.2.2.2 Data Type Conversion and Renaming Columns

Numerical fields such as price, average_rating, and rating_number were converted to float or integer types as appropriate. In addition, column names were renamed for clarity (e.g., title to product_title).

```python
# Convert data type
metadata_df['price'] = pd.to_numeric(metadata_df['price'], errors='coerce')
metadata_df['average_rating'] = pd.to_numeric(metadata_df['average_rating'], errors='coerce')
metadata_df['rating_number'] = pd.to_numeric(metadata_df['rating_number'], errors='coerce')
```

✓ 0.0s Python

Figure 4-22: Convert datatype for product metadata dataset.

```python
# Filter and Rename Columns
metadata_df = metadata_df[[
    'parent_asin', 'main_category', 'title', 'average_rating', 'rating_number', 'price', 'store', 'details'
]]
metadata_df.rename(columns={'title': 'product_title'}, inplace=True)


metadata_df.info()
```

✓ 0.0s Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 60288 entries, 0 to 60292
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   parent_asin     60288 non-null  object
 1   main_category   60288 non-null  object
 2   product_title   60288 non-null  object
 3   average_rating  60288 non-null  float64
 4   rating_number   60288 non-null  int64
 5   price           60288 non-null  float64
 6   store           60288 non-null  object
 7   details         60288 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 4.1+ MB
```

Figure 4-23: Filtering and Renaming Columns in the Product Metadata Dataset.

```python
# Double-check
print(f"Summary of data types: {metadata_df.info()}")
print(f"Missing values now: {metadata_df.isnull().sum()}")
print(f"Number of duplicate rows now: {metadata_df.duplicated().sum()}")
```

✓ 0.1s Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 60288 entries, 0 to 60292
Data columns (total 8 columns):
 #   Column          Non-Null Count  Dtype
---  ------          --------------  -----
 0   parent_asin     60288 non-null  object
 1   main_category   60288 non-null  object
 2   product_title   60288 non-null  object
 3   average_rating  60288 non-null  float64
 4   rating_number   60288 non-null  int64
 5   price           60288 non-null  float64
 6   store           60288 non-null  object
 7   details         60288 non-null  object
dtypes: float64(2), int64(1), object(5)
memory usage: 4.1+ MB
Summary of data types: None
Missing values now: parent_asin       0
main_category     0
product_title     0
average_rating    0
rating_number     0
price             0
store             0
details           0
dtype: int64
Number of duplicate rows now: 0
```

Figure 4-24: Final Verification of Cleaned Product Metadata Dataset.

```python
# show length of rows and columns from dataset after cleaning
print(f"Rows length: {len(metadata_df)}")
print(f"Columns length: {metadata_df.columns}")
```
✓ 0.0s Python

```
Rows length: 60288
Columns length: Index(['parent_asin', 'main_category', 'product_title', 'average_rating',
       'rating_number', 'price', 'store', 'details'],
      dtype='object')
```

Figure 4-25: Show length of rows and columns from product metadata after cleaning.

To make sure the formatting was correct, the column data types were examined. Price, average_rating, rating_number, and other important numeric variables were appropriately transformed to float or integer types, but text-based elements stayed as objects. The dataset's readiness for analysis and modeling was verified by this phase.

```python
# Check column data types
print(metadata_df.dtypes)
```
✓ 0.0s Python

```
parent_asin        object
main_category      object
product_title      object
average_rating    float64
rating_number       int64
price             float64
store              object
details            object
dtype: object
```

Figure 4-26: Check datatypes for product metadata.

The final cleaned product metadata dataset was saved as a CSV file using the to_csv() function. This ensures the preprocessed data is preserved and readily available for future analysis, modelling, or integration tasks, without the need to repeat the cleaning process.

```python
# save cleaned metadata dataset
metadata_df.to_csv("data/cleaned_metadata.csv", index=False)
```
✓ 0.7s Python

Figure 4-27: Save cleaned product metadata dataset as csv file after cleaning.

4.2.3 Merging Dataset

After cleaning the datasets, the next step was to merge the cleaned data to support integrated analysis. These were loaded using pandas.read_csv() function. The merging process was performed using the merge() function in Pandas, with 'parent_asin' as the common key, and an inner join was applied to retain only the records that appeared in both datasets. This ensured that only products with available reviews and metadata were included in the final dataset, improving data relevance and consistency.

```python
import pandas as pd

# Load cleaned files
reviews = pd.read_csv('data/cleaned_reviews_df.csv')
metadata = pd.read_csv('data/cleaned_metadata_df.csv')

# Merge
merged_df = pd.merge(reviews, metadata, on='parent_asin', how='inner')

# Preview
print(merged_df.head())
print(merged_df.shape)

# Save for later steps
merged_df.to_csv('data/merged_cleaned_data.csv', index=False)
```

✓ 16.9s Python

Figure 4-28: Merging Cleaned Reviews and Metadata Datasets.

The resulting DataFrame was previewed using the head() and shape() functions to describe its structure and the number of records. The final merged dataset consisted of 43,892 rows and multiple feature-rich columns, including review information, user feedback, and detailed product metadata. This dataset was then saved as merged_cleaned_data.csv for use in modelling, visualization, and sentiment-based recommendation tasks.

```python
# show length of rows and columns from dataset after cleaning
print(f"Rows length: {len(merged_df)}")
print(f"Columns length: {merged_df.columns}")
```

✓ 0.0s Python

```
Rows length: 488767
Columns length: Index(['rating', 'title', 'text', 'asin', 'parent_asin', 'user_id',
       'helpful_vote', 'verified_purchase', 'cleaned_text', 'main_category',
       'product_title', 'average_rating', 'rating_number', 'price', 'store',
       'details'],
      dtype='object')
```

Figure 4-29: Verifying Row and Column Count After Merging Datasets.

4.3 Sentiment Analysis

Sentiment analysis is essential to this research because it turns unstructured review text into attributes that can be analysed. This step uses TextBlob, an open-source Python natural language processing (NLP) module, to extract sentiment labels and polarity scores.

For cleaned reviews, this project determine a sentiment polarity score using the TextBlob library. With a score ranging from -1.0 (most negative) to +1.0 (most positive), 0.0 indicates a neutral attitude. These polarity values are used to classify the reviews according to preset thresholds:

Positive: a polarity score above +0.1.

Negative: a polarity score below -0.1.

Neutral: a polarity score in the range of -0.1 to +0.1.

By removing ambiguous or less opinionated statements as neutral and capturing strong sentiment expressions, this threshold-based method allows for efficient classification. To cut down on noise and enhance the dataset's sentiment grouping clarity, the specified thresholds were selected.

To convert qualitative review language into quantitative features—which are essential for subsequent tasks like visual analytics, recommendation systems, and predictive modelling—this type of sentiment categorization is used.

4.3.1 Sentiment Classification and Feature Engineering

This step involves defining a sentiment analysis function using a TextBlob object. TextBlob is an open-source Python library for processing textual data. It will provide a simple API for standard natural language processing (NLP) tasks, such as tokenization, sentiment analysis, or part-of-speech tagging. This project will utilize the TextBlob object for sentiment analysis because it can classify each review into positive, neutral, or negative categories, use these scores as numerical features in a recommendation system, and quickly extract polarity scores from each review.

The TextBlob library is used to create a sentiment analysis function, as shown in Figure 4-30. Based on predetermined thresholds, the algorithm assigns a polarity score to each cleaned review text and categorizes it as positive, negative, or neutral. The apply() function is then used to apply the sentiment label to every row in the dataset.

```python
def get_sentiment(cleaned_text):
    # Create a TextBlob object
    blob = TextBlob(str(cleaned_text))
    # Get polarity: [-1.0, 1.0]
    polarity = blob.sentiment.polarity
    # Classify sentiment
    if polarity > 0.1:
        return 'positive'
    elif polarity < -0.1:
        return 'negative'
    else:
        return 'neutral'

# Apply function row by row
cleaned_df['sentiment'] = cleaned_df['cleaned_text'].apply(get_sentiment)
```
✓ 2m 34.7s Python

Figure 4-30: Sentiment Classification Using TextBlob.

Figure 4-31 displays a preview of the sentiment labels assigned to the cleaned review text using the TextBlob-based classifier. It also presents the distribution of sentiment categories—positive, neutral, and negative—highlighting that the majority of reviews are positive, followed by neutral and negative sentiments. This provides an initial understanding of the overall customer sentiment trends in the dataset.

```python
# Preview updated dataframe
cleaned_df[['cleaned_text', 'sentiment']].head(10)

# Check distribution
cleaned_df['sentiment'].value_counts()
```
✓ 0.0s Python

sentiment	# count
positive	313525
neutral	128088
negative	41720

Figure 4-31: Sentiment Label Preview and Distribution in the Dataset.

4.3.2 Defining Sentiment Polarity Score

To enhance the interpretability of customer feedback, sentiment labels and scores were extracted from the cleaned review texts using the TextBlob library. First, a function was defined to compute polarity scores, which measure sentiment on a continuous scale from -1.0 (most negative) to 1.0 (most positive). These scores were applied to each review and rounded to three decimal places for readability. Alongside sentiment labels (positive, neutral, negative), the polarity values provided a more granular understanding of customer opinions. A sample preview of the dataset was generated to confirm the successful mapping of sentiment and polarity to each cleaned review.

```python
def get_polarity(cleaned_text):
    return TextBlob(str(cleaned_text)).sentiment.polarity

cleaned_df['polarity'] = cleaned_df['cleaned_text'].apply(get_polarity)
cleaned_df['polarity'] = cleaned_df['polarity'].round(3)
```
✓ 2m 41.6s Python

Figure 4-32: Extracting Polarity Scores from Cleaned Review Text.

```python
# Preview
cleaned_df[['cleaned_text', 'sentiment', 'polarity']].head(10)
```
✓ 0.0s Python

	cleaned_text	sentiment	polarity
0	this review is more to clarify someon	positive	0.232
1	love these easy multitasking bleach t	positive	0.307
2	i have been suffering a couple montl	neutral	0.061
3	its cheap and it does what i wanted t	neutral	-0.063
4	didnt do a thing for me not saying tl	neutral	0.0
5	i have never been big on vitamins wi	neutral	0.033
6	to be honest my husband did not us	positive	0.346
7	this is a nice basic sound machine i h	positive	0.211
8	this are super colorful great gift givir	positive	0.51
9	i guess the only product i have in my	positive	0.122

Figure 4-33: Preview of Cleaned Review Text with Sentiment Labels and Polarity Scores.

Finally, the fully processed dataset—including text, sentiment, and polarity information—was saved as a CSV file (merged_with_sentiment_and_polarity.csv). This comprehensive dataset serves as the foundation for downstream analysis, such as recommendation modelling and visualization of customer sentiment trends.

```python
# Save
cleaned_df.to_csv('data/merged_with_sentiment_and_polarity.csv', index=False)
print("Saved file: merged_with_sentiment_and_polarity.csv")
```
✓ 11.8s Python

```
Saved file: merged_with_sentiment_and_polarity.csv
```

Figure 4-34: Saving the Final Merged Dataset with Sentiment and Polarity.

5. Implementation

5.1 Exploratory Data Analysis (EDA)

To comprehend the structure, distribution, and integrity of the dataset before modeling, exploratory data analysis, or EDA, is a crucial first step. This section starts with the import of necessary libraries that offer the fundamental capabilities for statistical analysis and data visualization, including Pandas, NumPy, Matplotlib, and Seaborn.

```python
# improt require liraries

import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import re
```

✓ 0.5s Python

Figure 5-1: Import Require Libraries.

A basic structural summary of the cleaned dataset was generated using info() and describe() functions. The output revealed that the dataset contains 483,833 entries and 18 columns, including key features such as rating, title, text, helpful_vote, price, store, sentiment, and polarity. Data types were confirmed to be appropriately set, with integer and float types used for numeric analysis (e.g., rating, rating_number, price, polarity) and object types for categorical or textual information. To assess data completeness, missing value counts were evaluated using isnull().sum(). The results showed that the dataset was fully cleaned, with no missing entries in any column. This validation step is critical to ensure that downstream analysis and modelling tasks are not adversely affected by null values.

```python
# Basic summary
cleaned_df.info()
cleaned_df.describe()

# Check for nulls
cleaned_df.isnull().sum()
```

✓ 0.7s Python

```
<class 'pandas.core.frame.DataFrame'>
Index: 483333 entries, 0 to 488766
Data columns (total 18 columns):
 #   Column             Non-Null Count   Dtype
---  ------             --------------   -----
 0   rating             483333 non-null  int64
 1   title              483333 non-null  object
 2   text               483333 non-null  object
 3   asin               483333 non-null  object
 4   parent_asin        483333 non-null  object
 5   user_id            483333 non-null  object
 6   helpful_vote       483333 non-null  int64
 7   verified_purchase  483333 non-null  bool
 8   cleaned_text       483333 non-null  object
 9   main_category      483333 non-null  object
 10  product_title      483333 non-null  object
 11  average_rating     483333 non-null  float64
 12  rating_number      483333 non-null  int64
 13  price              483333 non-null  float64
 14  store              483333 non-null  object
 15  details            483333 non-null  object
 16  sentiment          483333 non-null  object
 17  polarity           483333 non-null  float64
dtypes: bool(1), float64(3), int64(3), object(11)
memory usage: 66.8+ MB
```

Figure 5-2: Summary of Dataset Structure and Null Value Check.

The heatmap displayed the absence of null values across all dimensions. This visual provides an intuitive understanding of data quality, reinforcing the robustness of the data preparation pipeline. This EDA workflow confirmed that the dataset was well-strength, complete, and ready for advanced analytical techniques such as sentiment trend analysis, feature correlations, and recommendation modelling.

```python
# Visualize missing values
sns.heatmap(cleaned_df.isnull(), cbar=False, cmap="viridis")
plt.title("Missing Values Heatmap")
plt.show()
```

✓ 6.3s Python

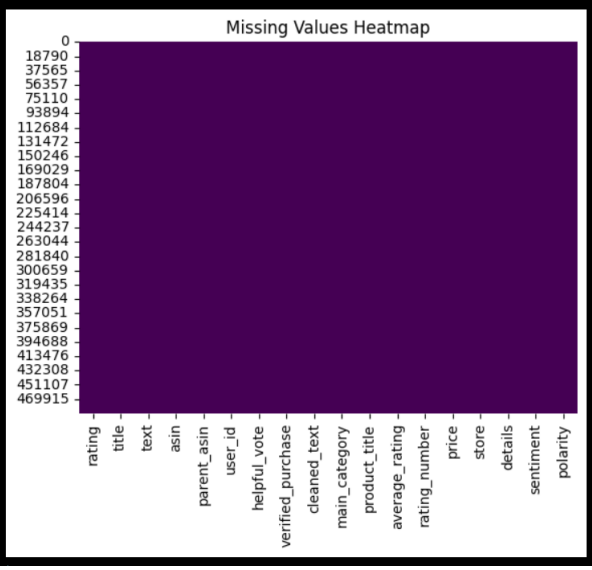

Figure 5-3: Heatmap Visualization of Missing Values in the Dataset.

To understand the linear relationships among core numerical variables — price, rating_number, average_rating, and polarity — a Pearson correlation matrix was computed. This matrix reveals how changes in one variable are associated with changes in another, which is critical for identifying potential predictors in machine learning models and refining business strategies.

```python
import seaborn as sns

# Correlation matrix: Price, Rating, Polarity
corr = cleaned_df[['price', 'rating_number', 'average_rating', 'polarity']].corr()
print("Correlation Matrix:\n", corr)

sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix')
plt.show()
```

✓ 0.3s Python

```
Correlation Matrix:
                   price  rating_number  average_rating  polarity
price           1.000000       0.006941        0.025656 -0.004794
rating_number   0.006941       1.000000        0.210667  0.005069
average_rating  0.025656       0.210667        1.000000  0.187836
polarity       -0.004794       0.005069        0.187836  1.000000
```

Figure 5-4: Correlation Matrix of Price, Rating Number, Average Rating, and Polarity.

The correlation matrix visualizes the linear relationships between four key variables: price, rating_number, average_rating, and polarity. Each value in the heatmap ranges from -1 (perfect negative correlation) to 1 (perfect positive correlation), with colour intensity indicating the strength of correlation.

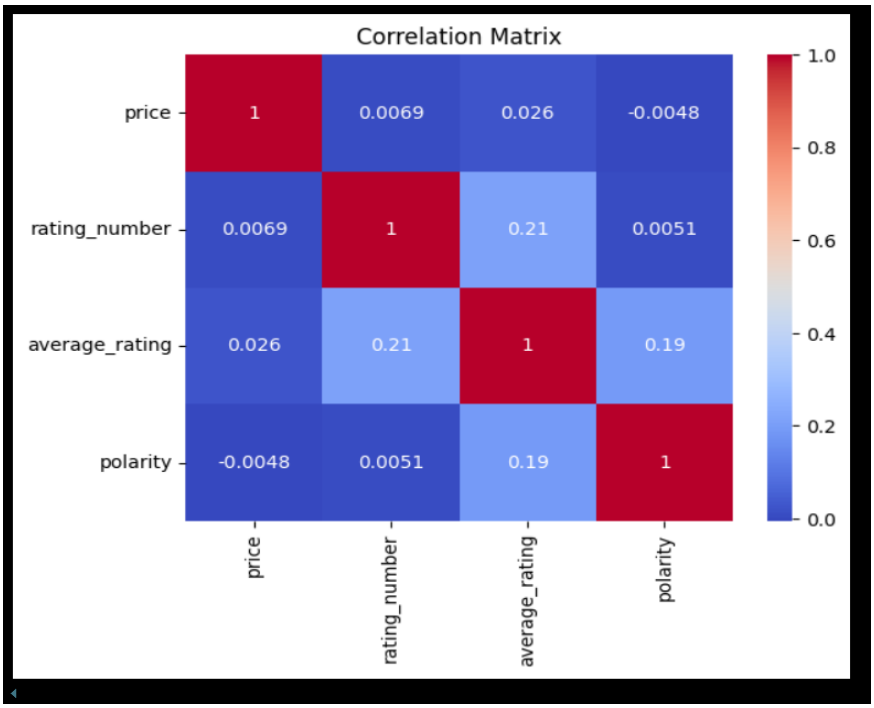

Figure 5-5: Heatmap Visualization of Correlation Among Key Numerical Features.

5.1.1 Data Visualization

This section provides visual sales insights from product reviews by analysing customer sentiments, ratings, and their relationships with product attributes. Through various plots, we examined the distribution of ratings, sentiment labels, average ratings per sentiment, and the relationship between sentiment polarity and product price. These visualizations support a comprehensive understanding of user behaviours and the nature of product feedback. The objective of this visualization is to uncover meaningful patterns that support informed decision-making in e-commerce.

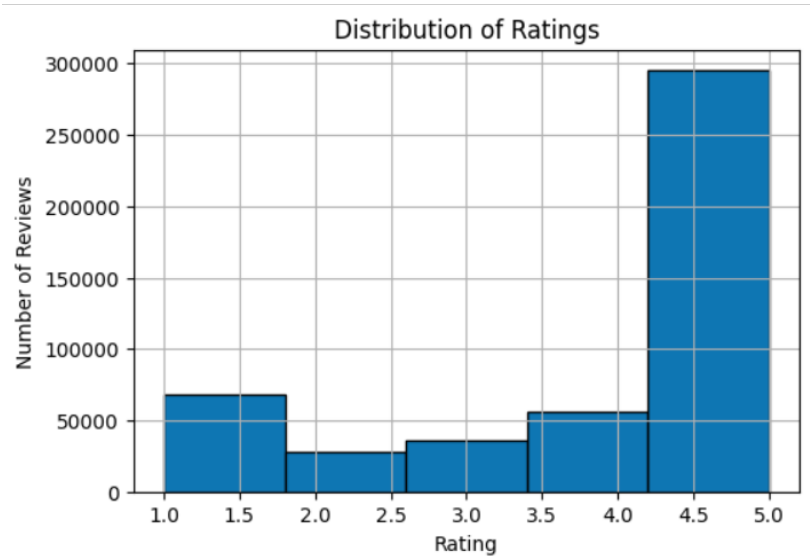

Figure 5-6: Distribution of Ratings.

Figure 5-6 illustrates the distribution of product ratings, revealing a notable skew towards higher ratings, particularly around 5.0. This suggests that the majority of customer feedback in the dataset is generally favourable.

In Figure 5-7, the distribution of sentiment labels shows a predominance of positive sentiments, followed by neutral, with negative reviews forming the smallest group. This distribution aligns with the rating trend and reinforces the positive bias in the data.

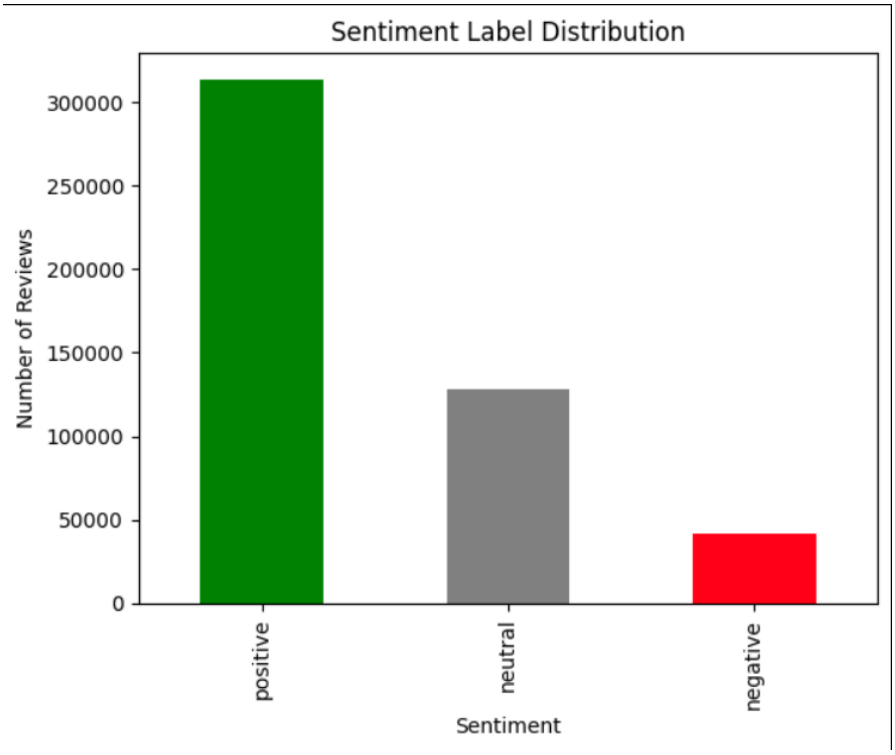

Figure 5-7: Sentiment Label Distribution.

To further analyse sentiment patterns, Figure 5-8 displays the average rating per sentiment group using a box plot. It confirms that reviews labelled as positive are associated with higher average ratings, whereas negative sentiments tend to have lower ratings.

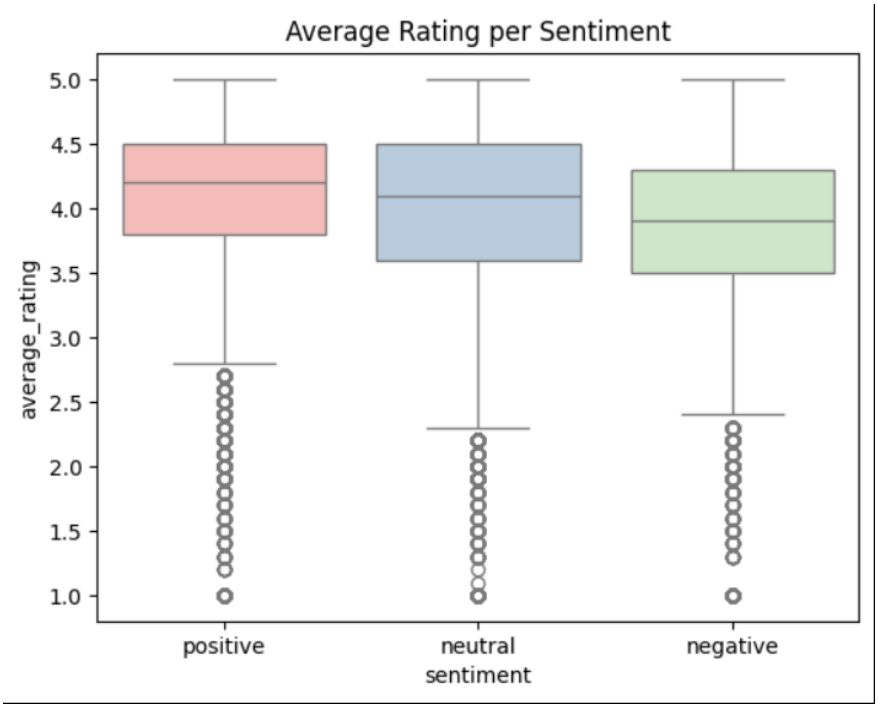

Figure 5-8: Average Rating per Sentiment.

A scatter plot between price and sentiment polarity provided insight into whether sentiment varies across product price ranges as illustrated Figure in 5-9. The absence of a strong correlation indicates that customer sentiment is more likely influenced by product quality and user experience rather than pricing alone.

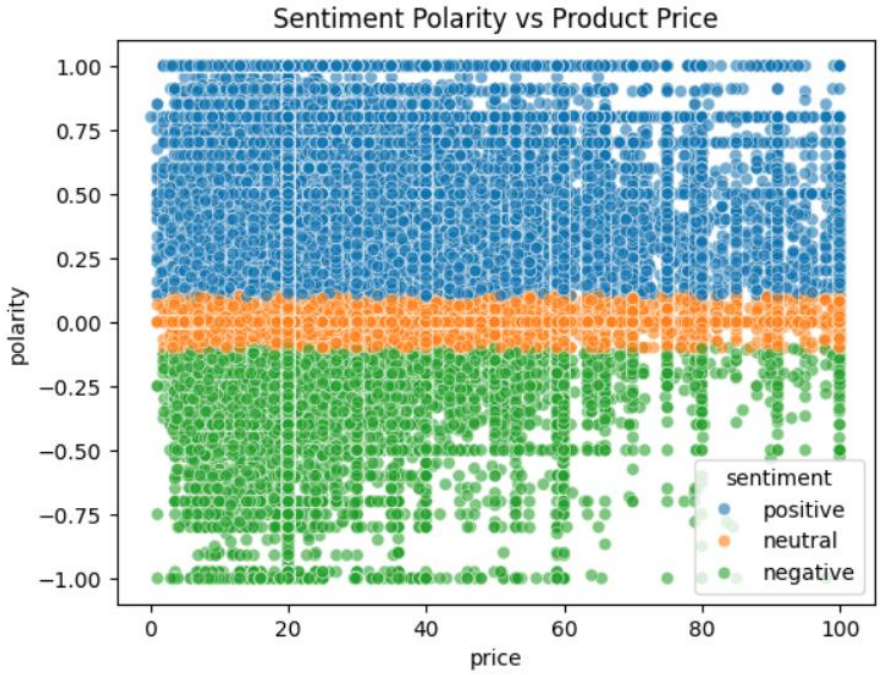

Figure 5-9: Sentiment Polarity vs Product Price.

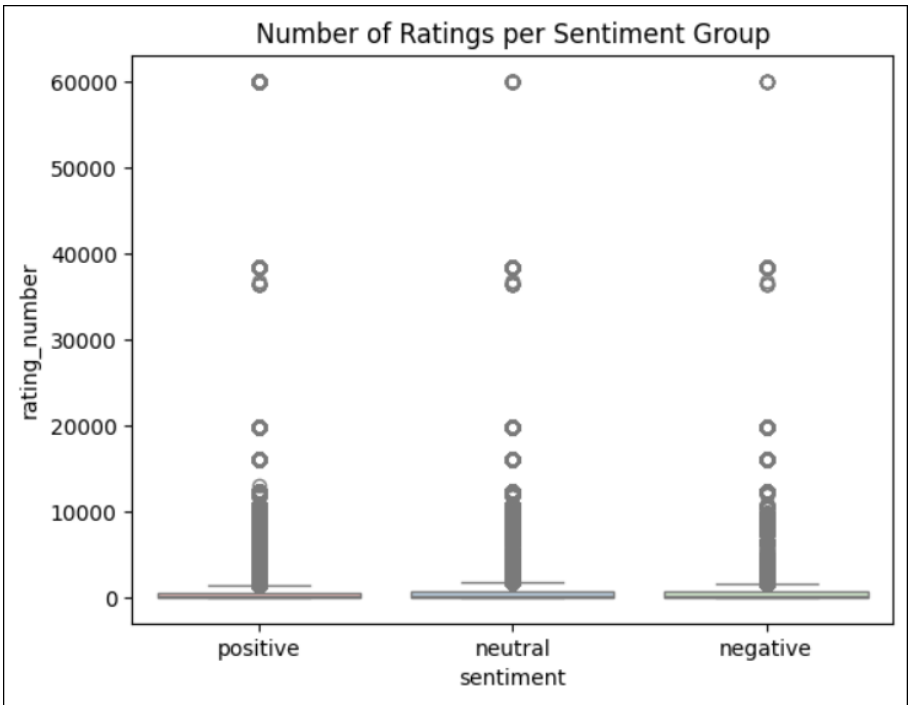

Figure 5-10: Number of Ratings per Sentiment Group.

Positive evaluations typically draw a higher volume of input, which could be a sign of a product's popularity and satisfaction. Figure 5-10 shows the number of ratings among sentiment groups.

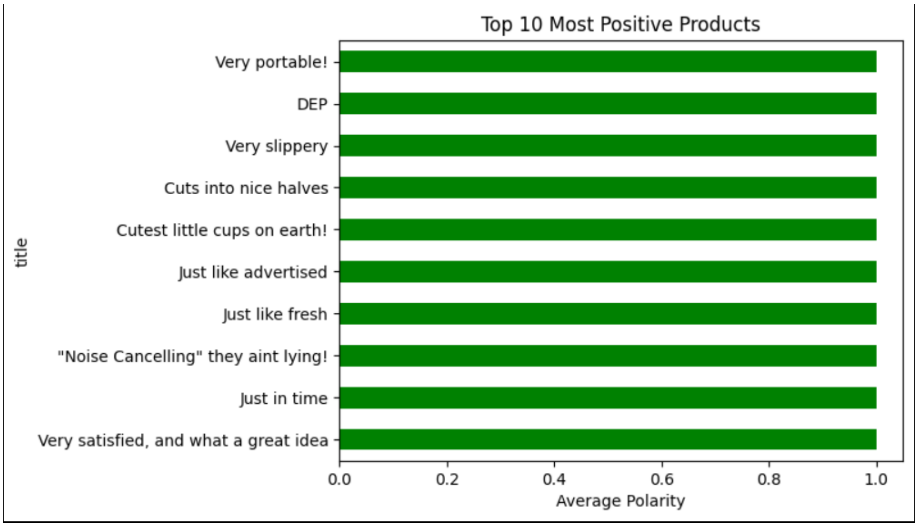

Figure 5-11: Top 10 Most Positive Products.

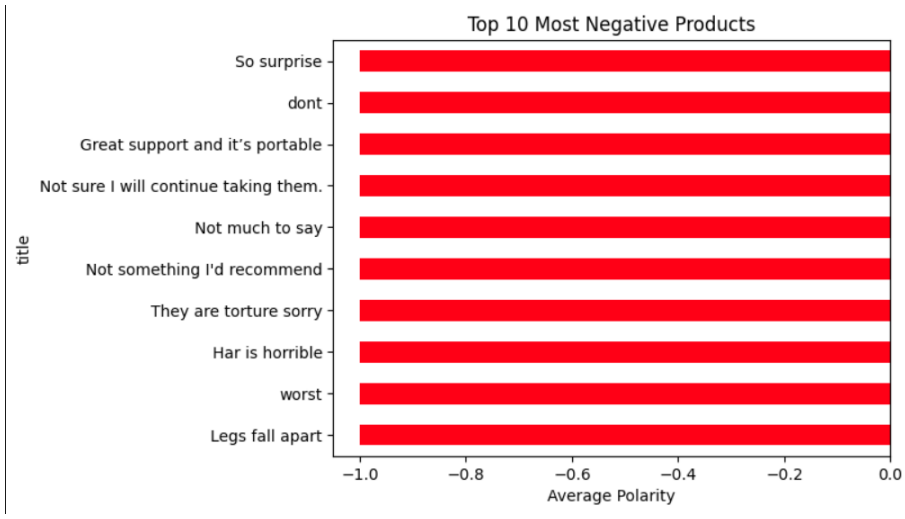

Figure 5-12: Top 10 Most Negative Products.

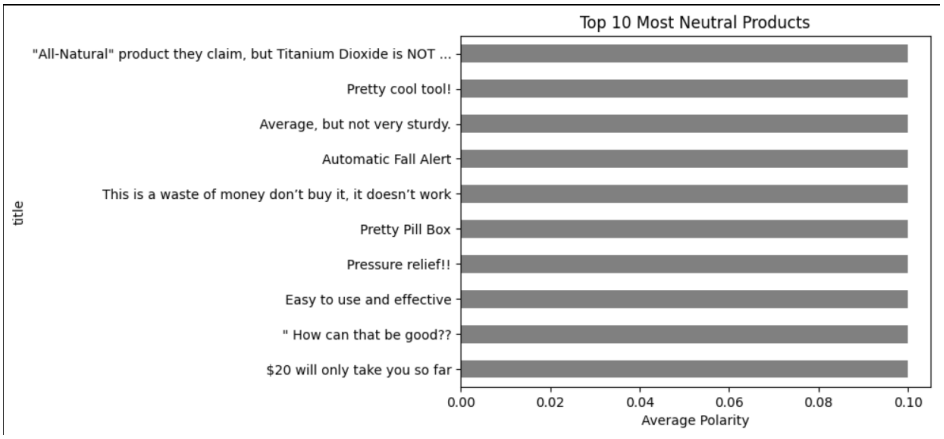

Figure 5-13: Top 10 Most Neutral Products.

The top 10 products in the positive, negative, and neutral sentiment categories are shown in Figures 5-11 through 5-13 according to average sentiment polarity. These graphs are useful for determining which products customers respond to the most favourably or unfavourably, as well as those that get mixed reviews. Product enhancements and marketing plans can be informed by this data.

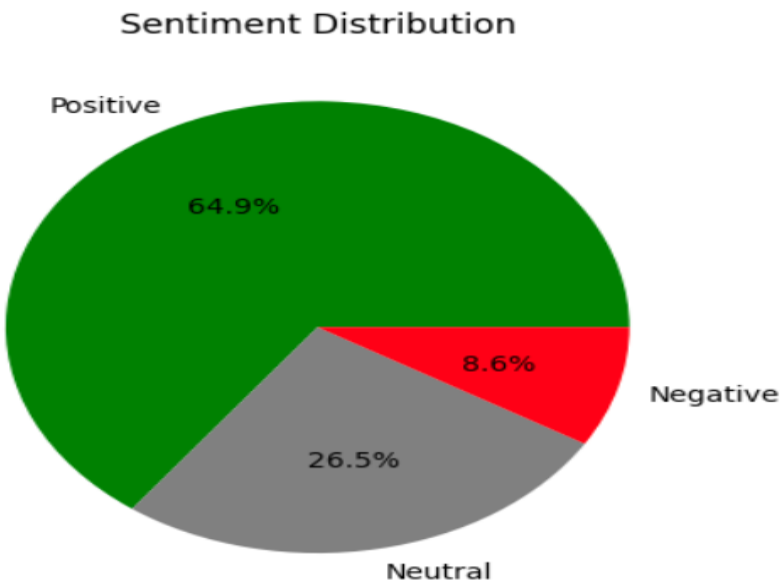

Figure 5-14: Sentiment Distribution Pie Chart.

Figure 5-14 presents a pie chart summarizing the sentiment distribution, which reinforces the earlier finding that the dataset is primarily dominated by positive feedback.

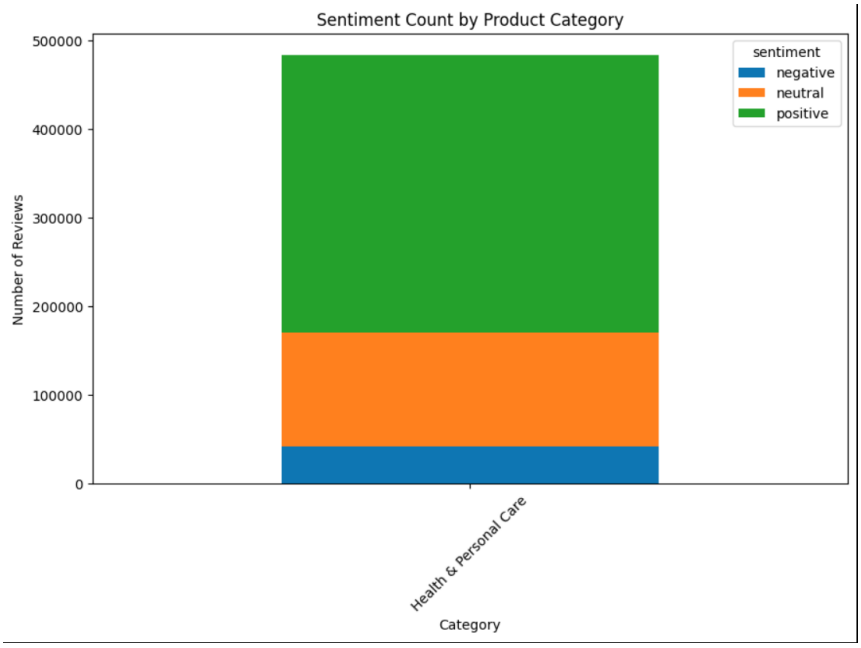

Figure 5-15: Sentiment Count by Product Category.

A grouped bar chart displayed the sentiment distribution across major product categories (Health & Personal Care), helping understand how customer sentiment differs depending on the product domain in Figure 5-15. This is particularly useful for identifying product categories with the highest proportion of negative feedback, which may require improvement.

All of these visualizations provide a thorough understanding of user activity, sentiment patterns, and product perception, which is essential for generating valuable insights and providing recommendation systems in e-commerce platforms.

5.2 Sales Insight Exploration

To extract meaningful insights that inform both product strategy and customer engagement, we conducted a comprehensive exploration of key sales indicators derived from sentiment-enriched review data. The goal of this section is to understand how product price, customer rating, and sentiment polarity relate to consumer engagement, measured through the number of reviews.

This sales insight analysis integrates multiple KPIs—sentiment polarity, price, rating, and volume—to reveal a multidimensional view of product performance. By combining statistical correlation, top-performer detection, and visual diagnostics, it provides a roadmap for:

- Identifying marketing candidates,
- Flagging underperformers,
- Prioritizing high-potential products, and
- Optimizing inventory and promotion strategies.

These insights empower data-driven decisions to enhance customer satisfaction, profitability, and operational efficiency in the e-commerce platform.

The merged dataset containing product-level attributes and sentiment scores was loaded for analysis. We computed KPIs at both the product (parent_asin) and category (main_category) levels, aggregating average price, average rating, mean sentiment polarity, and review counts. This setup enabled a robust analysis of performance patterns across individual items and broader product segments.

```python
import pandas as pd
import seaborn as sns
import matplotlib.pyplot as plt

# Load the merged dataset
df = pd.read_csv('data/merged_with_sentiment_and_polarity.csv')
df.head()
df.info()
print(df.columns.tolist())
```

Python

Figure 5-16: Dataset Loading and Structure Overview.

```python
# Group by product ID
product_kpis = df.groupby('parent_asin').agg({
    'price': 'mean',
    'rating': 'mean',
    'polarity': 'mean',
    'user_id': 'count'
}).reset_index().rename(columns={'user_id': 'num_reviews'})
print(product_kpis.head())
```

✓ 0.2s Python

```
  parent_asin  price  rating  polarity  num_reviews
0  098834002X  19.99     5.0     0.320            1
1  099109462X  19.99     5.0     0.325            1
2  099113351X  19.99     4.5     0.174            4
3  111896229X  19.99     5.0     0.395            1
4  1259211282  16.00     3.5     0.195            2
```

Figure 5-17: Product-Level KPI Aggregation.

This following figure summarizes key metrics aggregated by main_category. It presents the average rating, average sentiment polarity, number of unique products, and total number of reviews per category. Such category-level insights help identify high-performing sectors and guide marketing or promotional focus, with *Health & Personal Care* standing out prominently.

```python
# Group by category
category_kpis = df.groupby('main_category').agg({
    'rating': 'mean',
    'polarity': 'mean',
    'parent_asin': pd.Series.nunique,
    'user_id': 'count'
}).reset_index().rename(columns={'parent_asin': 'num_products', 'user_id': 'num_reviews'})
print(category_kpis.sort_values(by='polarity', ascending=False))
```

✓ 0.2s Python

```
            main_category    rating  polarity  num_products  num_reviews
0  Health & Personal Care  3.995341  0.248463         59247       483333
```

Figure 5-18: Category-Level KPI Aggregation.

The Health & Personal Care category, the largest in the dataset, averaged a strong 3.99 star rating and a relatively high sentiment polarity (0.248). With over **59,000 products** and nearly **483,000 reviews**, this category presents immense potential for customer satisfaction optimization and targeted marketing.

5.2.1 Correlation Analysis of KPIs

A correlation heatmap (**Figure 4.6**) was generated to quantify relationships between numeric KPIs such as price, rating, polarity, and review count. The analysis revealed a strong positive correlation between rating and polarity ($r \approx 0.52$), affirming that reviews with higher ratings tend to express more positive sentiment. Interestingly, price and number of reviews showed negligible correlations with sentiment and rating, reinforcing that customer satisfaction is largely independent of price point.

```python
# Correlation of numeric KPIs
corr = product_kpis[['price', 'rating', 'polarity', 'num_reviews']].corr()
print(corr)
sns.heatmap(corr, annot=True, cmap='coolwarm')
plt.title('Correlation Matrix: Price, Rating, Polarity, Num Reviews')
plt.show()
```

✓ 0.2s Python

```
                price    rating  polarity  num_reviews
price        1.000000  0.004114 -0.005185    -0.001156
rating       0.004114  1.000000  0.516910     0.009465
polarity    -0.005185  0.516910  1.000000    -0.000953
num_reviews -0.001156  0.009465 -0.000953     1.000000
```

Figure 5-19: Correlation Matrix of Key Performance Indicators (KPIs).

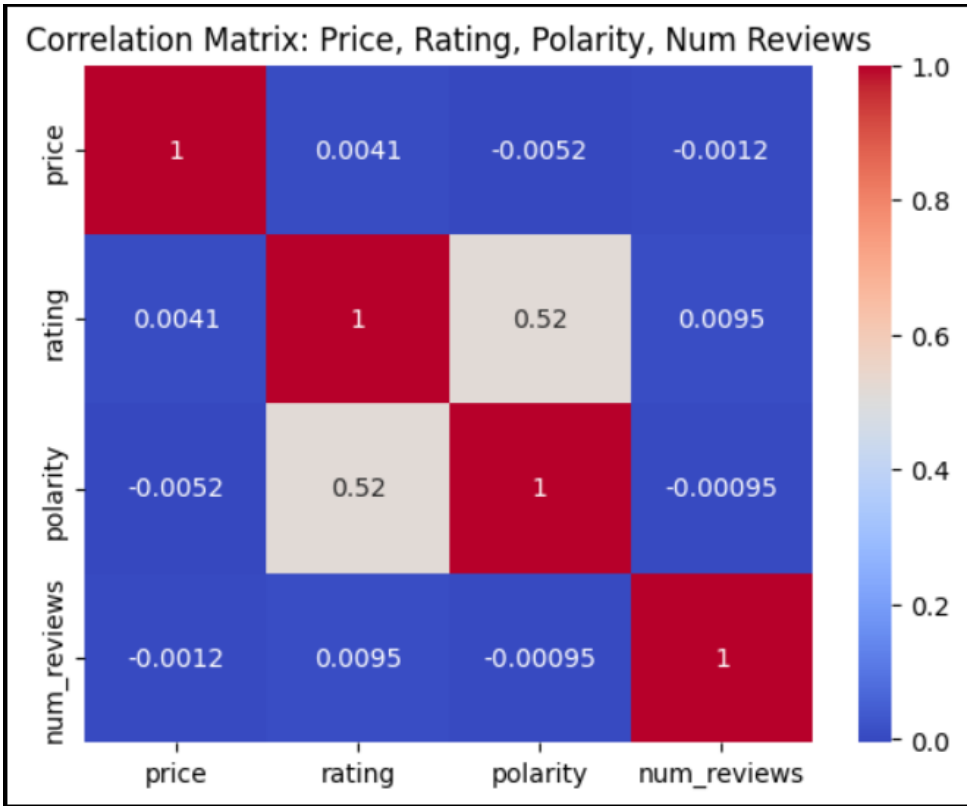

Figure 5-20: Correlation Matrix of Price, Rating, Polarity, and Number of Reviews.

A correlation heatmap was generated to assess linear relationships between four key numerical metrics: **Price**, **Rating**, **Sentiment Polarity**, and **Number of Reviews**. **Rating and Polarity** showed a moderately strong positive correlation (~0.52), confirming that higher-rated products tend to have more positive sentiment. **Price** had negligible correlation with other metrics, suggesting pricing may not directly influence customer sentiment or engagement. **Number of Reviews** was weakly correlated with other metrics, indicating review volume may be driven more by popularity or external factors than product quality alone.

5.2.2 Top Performing Products by Sentiment and Rating

This table displays the top 10 products with the highest average sentiment polarity, reflecting extremely positive customer feedback. All selected products have a polarity score of 1.0 and a perfect 5.0 rating, although each has only one review. While sentiment is high, low review counts suggest these products may be new or niche offerings with limited visibility.

```python
# Top products by polarity
top_sentiment_products = product_kpis.sort_values(by='polarity', ascending=False).head(10)
print(top_sentiment_products)
```

✓ 0.0s Open 'top_sentiment_products' in Data Wrangler Python

```
       parent_asin  price  rating  polarity  num_reviews
11388   B00LOYAGA0  19.99     5.0       1.0            1
18835   B01BV36SB4  19.99     5.0       1.0            1
40222   B07TV2HYHK  19.99     5.0       1.0            1
5604    B006MW89LW  19.99     5.0       1.0            1
14145   B00VMTGK2K  19.99     5.0       1.0            1
2890    B002F5EPNG  19.99     5.0       1.0            1
49574   B08GL1MQMJ  42.89     5.0       1.0            1
845     B000GCS9FU  19.99     5.0       1.0            1
14076   B00VBG5FE8  19.99     5.0       1.0            1
32296   B07C7YD4QD  19.99     5.0       1.0            1
```

Figure 5-21: Top 10 Products by Sentiment Polarity.

This figure presents the products with the highest customer ratings, all scoring a perfect 5.0. Despite their excellent ratings, each product has only one review, which may indicate limited market exposure. Polarity scores vary, suggesting differing sentiment richness across these highly-rated items. The price range spans from budget-friendly to premium, offering insight into customer satisfaction across pricing tiers.

```python
# Top products by rating
top_rating_products = product_kpis.sort_values(by='rating', ascending=False).head(10)
print(top_rating_products)
```

✓ 0.0s Open 'top_rating_products' in Data Wrangler Python

```
       parent_asin  price  rating  polarity  num_reviews
14      156533535X  55.20     5.0     0.258            1
59246   B0CHQ69MCN  11.98     5.0     0.682            1
0       098834002X  19.99     5.0     0.320            1
1       099109462X  19.99     5.0     0.325            1
15      1569350450  19.99     5.0     0.263            2
3       111896229X  19.99     5.0     0.395            1
16      1579183468  20.95     5.0     0.235            2
17      158256048X  39.75     5.0     0.500            1
6       1323471022  19.99     5.0     1.000            1
7       1402512945  19.99     5.0     0.223            1
```

Figure 5-22: Top 10 Products by Customer Rating.

To identify products resonating most positively with customers, we examined the top 10 items ranked by sentiment polarity and average rating. As shown in **Figure 5-22**, these products received perfect ratings (5.0) and the highest possible sentiment score (1.0), even though most had only one review. This suggests initial customer impressions are extremely favourable and may indicate strong product-market fit.

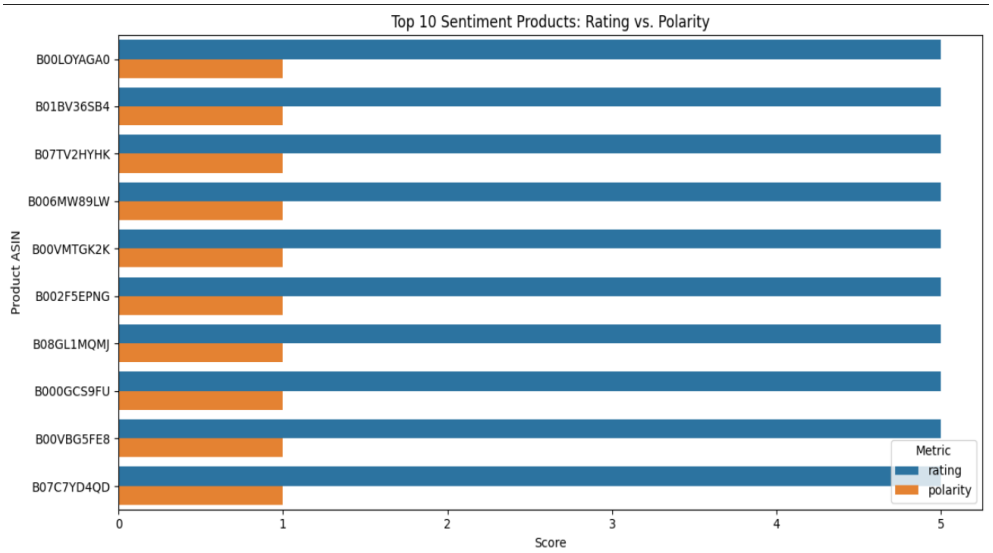

Figure 5-23: Top 10 Sentiment Products — Comparison of Rating vs. Polarity.

This bar chart compares rating and polarity for each of the top 10 high-sentiment products. While high scores across both metrics indicate customer satisfaction, the low review volume implies untapped potential. These products are ideal candidates for promotional amplification.

5.2.3 Sentiment vs. Price Relationship

To explore whether customer sentiment is influenced by pricing, we plotted average price against average sentiment polarity for all products (see **Figure 4.8**).

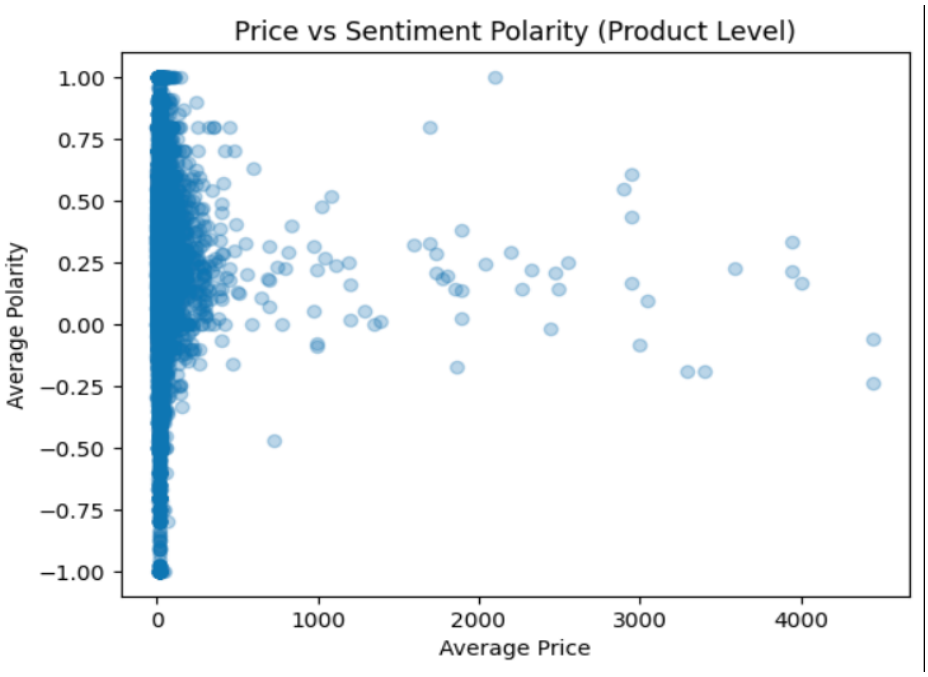

Figure 5-24: Price vs Sentiment Polarity at Product Level.

The scatter plot shows that the majority of products are priced under $100, yet display a wide range of sentiment polarity. No clear correlation is evident, which suggests that positive sentiment is not necessarily tied to premium pricing. This opens up opportunities to promote lower-priced products that customers genuinely appreciate.

5.2.4 Outlier Product Detection

Outlier detection in product performance helps identify anomalies that can influence business decisions—both positively and negatively. In this section, we leverage sentiment polarity, review volume, and product ratings to detect **weak products**, **top-reviewed products**, and understand overall sentiment-review dynamics.

We define *weak products* as those that are **overpriced**, have **low sentiment polarity**, and **low ratings**. These are potential candidates for delisting, price revision, or quality improvement. The filtering logic applied selected products meeting the following conditions:

- Sentiment polarity < 0
- Rating < 3
- Price > overall average

This logic isolates products that not only fail to satisfy customers but also cost more than the average product—potentially harming both reputation and conversion rates.

```python
weak_products = product_kpis[
    (product_kpis['polarity'] < 0) &
    (product_kpis['rating'] < 3) &
    (product_kpis['price'] > product_kpis['price'].mean())
]

print(weak_products)
```

✓ 0.0s Open 'weak_products' in Data Wrangler Python

```
      parent_asin   price    rating  polarity  num_reviews
281    B0001KHN2M  164.95  2.666667 -0.064667            3
793    B000FQTJXI   35.40  1.000000 -0.333000            1
886    B000GSXRFG  159.99  1.000000 -0.015500            2
1245   B000P4S2GK   82.21  1.000000 -0.078000            1
1642   B000YL95CE  259.99  2.500000 -0.100000            2
...           ...     ...       ...       ...          ...
58412  B0BB94Q7NB   29.99  1.833333 -0.010333            6
58607  B0BGPS4K7Z  139.90  2.000000 -0.250000            1
58840  B0BMF5KVB7   39.99  1.000000 -0.250000            1
59005  B0BRYL5JPW   49.00  1.000000 -0.312000            1
59153  B0BZW5K6YK   35.99  1.000000 -0.032000            1

[150 rows x 5 columns]
```

Figure 5-25: List of Weak Products.

The DataFrame output shows products with a price above the mean and negative sentiment, with ratings close to the lowest possible value. These items merit immediate review by business stakeholders.

The volume of reviews provides an indirect measure of sales or customer interaction. To examine how sentiment affects customer engagement, we visualized the relationship between **number of reviews** and **average sentiment polarity**.

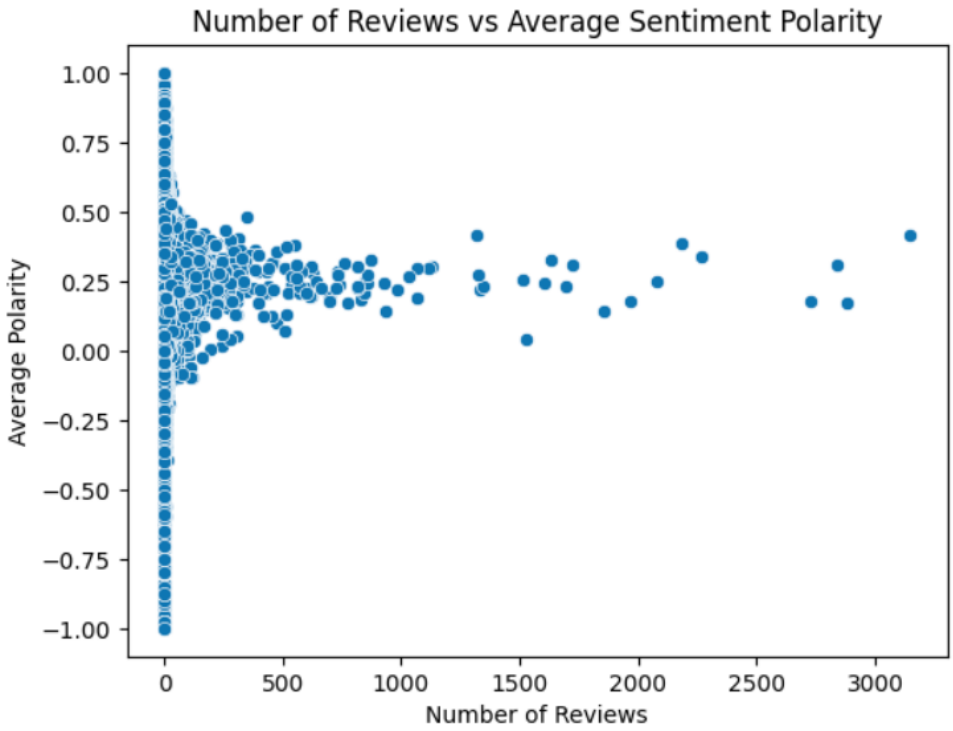

Figure 5-26: Number of Reviews vs Average Sentiment Polarity.

This scatter plot indicates that while most products with 1–100 reviews show polarity values cantered around 0.25, a few outliers exist with high polarity but low review counts. This suggests a small group of very satisfied customers, pointing to high-potential "hidden gems."

we identified products with the highest number of reviews as a proxy for strong sales and engagement.

```python
top_volume = product_kpis.sort_values(by='num_reviews', ascending=False).head(10)
print(top_volume[['parent_asin', 'num_reviews', 'rating', 'polarity']])
```

✓ 0.0s Python

```
       parent_asin  num_reviews    rating  polarity
59163  B0C1HHXQ94         3146  4.684361  0.417860
53261  B08X9LB1WC         2882  3.839348  0.169833
58027  B0B4328BFW         2839  4.304685  0.307128
19085  B01CKMP406         2727  4.048405  0.176765
4166   B0047VWYSO         2269  4.663288  0.339629
4234   B004C7MTLA         2182  4.733731  0.389283
299    B0001ZWPI4         2075  4.264578  0.248958
5806   B0077L8YFI         1969  3.180802  0.180807
12184  B00O0CK2UM         1854  3.398598  0.141674
7712   B00BR1FSU8         1722  4.361208  0.306693
```

Figure 5-27: Top 10 Products by Review Volume.

This list highlights the most-reviewed items, several of which also maintain favourable polarity scores. These products represent established, well-performing items in the catalog, deserving continued support and promotion.

To reverse the relationship, we also plotted **sentiment polarity** against **number of reviews**, helping us understand how sentiment contributes to product engagement levels.

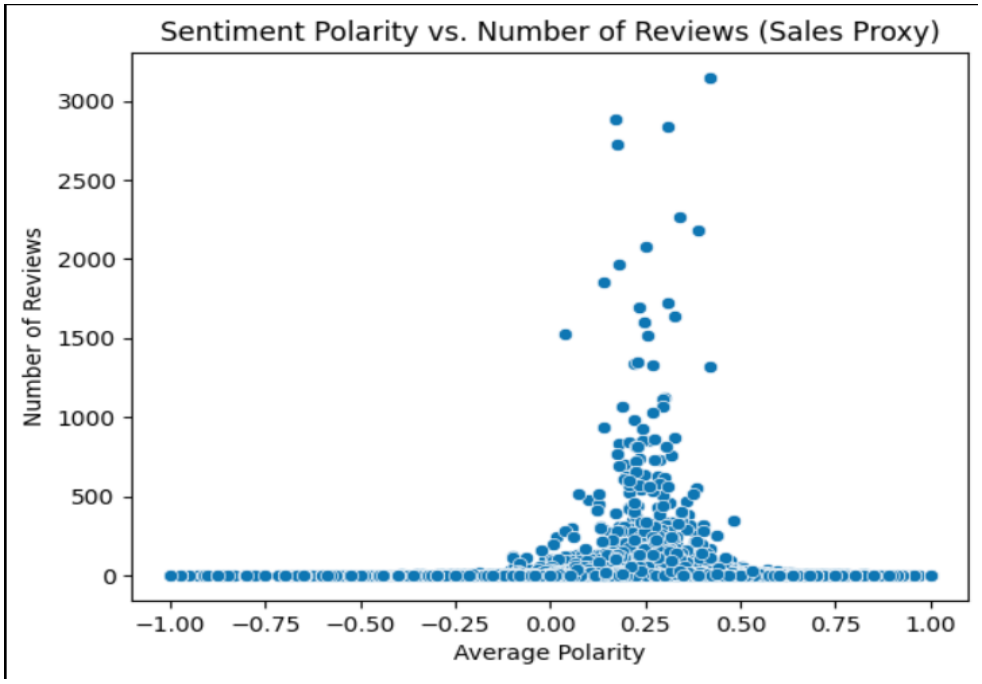

Figure 5-28: Sentiment Polarity vs. Number of Reviews (Sales Proxy).

The distribution is bell-shaped, peaking around a polarity of 0.25. This suggests that moderately positive sentiment aligns with higher review volume. Interestingly, extreme positive polarity values (e.g., close to 1.0) are mostly associated with fewer reviews, potentially due to niche or newer products.

5.2.5 Strategic Product Segmentation

Two special product segments were extracted:

1. **Hidden Gems**: Low-priced products with high ratings and strong positive sentiment — ideal for promotions. These are affordable, loved by customers, and could gain more traction with visibility boosts.

2. **Overpriced Poor Performers**: Expensive products receiving low ratings and negative sentiment — these may require quality improvement or pricing reconsideration.

```python
hidden_gems = product_kpis[
    (product_kpis['price'] < product_kpis['price'].mean()) &
    (product_kpis['rating'] > 4.0) &
    (product_kpis['polarity'] > 0.5)
]

print(hidden_gems)
```

✓ 0.0s Open 'hidden_gems' in Data Wrangler Python

```
      parent_asin  price    rating  polarity  num_reviews
6      1323471022  19.99  5.000000  1.000000            1
28     1617013110  19.99  5.000000  0.577500            2
41     193354337X  19.99  5.000000  0.700000            1
46     193548236X  19.99  5.000000  0.514750            4
51     1938579992  19.99  5.000000  0.600000            1
...           ...    ...       ...       ...          ...
59117  B0BX72SWGF  17.80  5.000000  0.512000            1
59178  B0C2PXG21N  19.99  5.000000  0.635333            3
59231  B0C9QGJPPS  19.99  5.000000  0.800000            1
59237  B0CC1JMCJG  10.99  4.866667  0.530467           30
59246  B0CHQ69MCN  11.98  5.000000  0.682000            1

[6011 rows x 5 columns]
```

Figure 5-29: Identification of Hidden Gem Products Using Price, Rating, and Sentiment Polarity Filters.

```python
overpriced_bad = product_kpis[
    (product_kpis['price'] > product_kpis['price'].mean()) &
    (product_kpis['rating'] < 3.0) &
    (product_kpis['polarity'] < 0)
]

print(overpriced_bad)
```

✓ 0.0s Open 'overpriced_bad' in Data Wrangler Python

```
      parent_asin   price    rating  polarity  num_reviews
281    B0001KHN2M  164.95  2.666667 -0.064667            3
793    B000FQTJXI   35.40  1.000000 -0.333000            1
886    B000GSXRFG  159.99  1.000000 -0.015500            2
1245   B000P4S2GK   82.21  1.000000 -0.078000            1
1642   B000YL95CE  259.99  2.500000 -0.100000            2
...           ...     ...       ...       ...          ...
58412  B0BB94Q7NB   29.99  1.833333 -0.010333            6
58607  B0BGPS4K7Z  139.90  2.000000 -0.250000            1
58840  B0BMF5KVB7   39.99  1.000000 -0.250000            1
59005  B0BRYL5JPW   49.00  1.000000 -0.312000            1
59153  B0BZW5K6YK   35.99  1.000000 -0.032000            1

[150 rows x 5 columns]
```

Figure 5-30: Identification of Overpriced Low-Quality Products Based on Price, Rating, and Negative Sentiment.

This dual insight helps identify not only high-value products worth investing in, but also underperformers that could erode brand reputation if not addressed.

5.3 Machine Learning Model Implementation

This section presents the implementation of three machine learning algorithms used to classify customer sentiment: Logistic Regression, Naïve Bayes, and Support Vector Machine (SVM). The overall workflow includes preprocessing the textual data using TF-IDF, splitting the dataset, training the models, and evaluating them using accuracy, F1 score, and classification reports (Zahoor and Rohilla, 2020).

The sentiment classification task begins with preparing the text and label columns. The cleaned_text column serves as input features while the sentiment column is the target label comprising three classes: **positive**, **negative**, and **neutral**.

```python
# Use cleaned_text and TextBlob sentiment label for now
X = final_df['cleaned_text']
y = final_df['sentiment']   # Must be 'positive', 'negative', 'neutral'
```
✓ 0.0s Python

Figure 5-31: Text Cleaning and Sentiment Label Preparation.

Next, the text data is transformed using **TF-IDF Vectorization**, converting it into a numerical format suitable for model training:

```python
tfidf = TfidfVectorizer()
X_tfidf = tfidf.fit_transform(X)

print(X_tfidf.shape)  # (rows, features)
```
✓ 19.1s Python

```
(483333, 151164)
```

Figure 5-32: TF-IDF Vectorization.

After feature transformation, the dataset is split into training and testing sets with an 80:20 ratio using train_test_split:

```python
from sklearn.model_selection import train_test_split

X_train, X_test, y_train, y_test = train_test_split(X_tfidf, y, test_size=0.2, random_state=42)

print(X_train.shape, X_test.shape)

print(y_test.value_counts())
```
✓ 0.1s Python

```
(386666, 151164) (96667, 151164)
sentiment
positive    62615
neutral     25724
negative     8328
Name: count, dtype: int64
```

Figure 5-33: Train-Test Data Split and Distribution.

5.3.1 Logistic Regression Algorithm

The first classification model implemented is Logistic Regression. The model was trained on the TF-IDF features and evaluated on the test set. The classification report and confusion matrix were used to assess performance.

```python
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix,f1_score

lr = LogisticRegression(max_iter=1000)
lr.fit(X_train, y_train)

lr_pred = lr.predict(X_test)

print("Logistic Regression Report")
print(classification_report(y_test, lr_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, lr_pred))
print("Accuracy:", accuracy_score(y_test, lr_pred))
```

✓ 53.4s Python

```
Logistic Regression Report
              precision    recall  f1-score   support

    negative       0.82      0.57      0.68      8328
     neutral       0.79      0.85      0.82     25724
    positive       0.95      0.95      0.95     62615

    accuracy                           0.89     96667
   macro avg       0.85      0.79      0.82     96667
weighted avg       0.89      0.89      0.89     96667

Confusion Matrix:
 [[ 4781  3162   385]
 [  851 21975  2898]
 [  193  2683 59739]]
Accuracy: 0.8947727766455977
```

Figure 5-34: Logistic Regression Classification Report and Confusion Matrix.

This figure displays the performance of the Logistic Regression model, including precision, recall, F1-score, and support for each sentiment class. The confusion matrix at the bottom shows the actual vs. predicted classifications. The model achieved a high accuracy of **0.8948**, with particularly strong performance for the **positive** class.

5.3.2 Naïve Bayes Algorithm

Naïve Bayes was the second model evaluated. Though commonly used for text classification due to its simplicity and speed, it struggled to match Logistic Regression in performance.

```python
from sklearn.naive_bayes import MultinomialNB

nb = MultinomialNB()
nb.fit(X_train, y_train)
nb_pred = nb.predict(X_test)

print("Naive Bayes Report")
print(classification_report(y_test, nb_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, nb_pred))
print("Accuracy:", accuracy_score(y_test, nb_pred))
```
✓ 3.2s Python

```
Naive Bayes Report
              precision    recall  f1-score   support

    negative       0.98      0.03      0.05      8328
     neutral       0.61      0.08      0.14     25724
    positive       0.67      1.00      0.80     62615

    accuracy                           0.67     96667
   macro avg       0.76      0.37      0.33     96667
weighted avg       0.68      0.67      0.56     96667

Confusion Matrix:
 [[  214  1110  7004]
 [    2  2035 23687]
 [    2   165 62448]]
Accuracy: 0.6692770024930949
```

Figure 5-35: Naïve Bayes Classification Report and Confusion Matrix.

In figure 5-35, the Naive Bayes model achieved an accuracy of 66.93%, with strong precision for negative sentiment but very low recall, indicating it struggled to identify negative samples. Its performance was more balanced for neutral and positive classes, but overall, it underperformed compared to other models.

5.3.3 Support Vector Machine

SVM with a linear kernel was applied using LinearSVC, known for its robustness in high-dimensional spaces like TF-IDF vectors. The model produced strong and consistent performance across all sentiment classes.

```python
from sklearn.svm import LinearSVC
from sklearn.metrics import classification_report, confusion_matrix, accuracy_score,f1_score

# Train SVM
svm = LinearSVC()
svm.fit(X_train, y_train)

# Predict
svm_pred = svm.predict(X_test)

# Evaluate
print("SVM Results:\n", classification_report(y_test, svm_pred))
print("Confusion Matrix:\n", confusion_matrix(y_test, svm_pred))
print("Accuracy:", accuracy_score(y_test, svm_pred))
```
✓ 56.3s Python

```
SVM Results:
              precision    recall  f1-score   support

    negative       0.74      0.57      0.65      8328
     neutral       0.78      0.79      0.78     25724
    positive       0.93      0.95      0.94     62615

    accuracy                           0.88     96667
   macro avg       0.82      0.77      0.79     96667
weighted avg       0.87      0.88      0.87     96667

Confusion Matrix:
 [[ 4771  3137   420]
 [ 1464 20269  3991]
 [  221  2647 59747]]
Accuracy: 0.8771038720556136
```

Figure 5-36: Support Vector Machine (SVM) Classification Report and Confusion Matrix.

Figure 5-36 shows the implementation and evaluation of the Support Vector Machine (SVM) model using LinearSVC. The model achieved an **accuracy of 0.8771**. It performed well in identifying **positive sentiment** (F1-score: 0.94) and **neutral sentiment** (F1-score: 0.79), but was slightly weaker in detecting **negative sentiment** (F1-score: 0.65). Overall, the SVM model demonstrated strong performance for sentiment classification in the dataset.

The trained models were serialized and saved using joblib for future reuse without retraining:

```python
joblib.dump(lr, 'models/sentiment_logreg.pkl')
joblib.dump(nb, 'models/sentiment_naivebayes.pkl')
joblib.dump(svm, 'models/sentiment_svm.pkl')
```

✓ 0.0s Python

Figure 5-37: Model Serialization using joblib..

6 Experimentation

The experimentation phase aimed to evaluate and compare the performance of multiple machine learning algorithms before and after applying hyperparameter tuning. The models tested included Logistic Regression, Naive Bayes, and Support Vector Machine (SVM), using both default settings and optimized configurations via GridSearchCV.

The evaluation was based on **Accuracy** and **F1 Score**, two widely accepted metrics for classification tasks. Accuracy provides the overall correctness, while F1 Score balances precision and recall, which is critical for imbalanced datasets in sentiment classification.

6.1 Model Performance of Machine Learning

Before tuning, all models were trained and evaluated using accuracy and F1 score as performance metrics. Logistic Regression emerged as the top performer with an accuracy of **0.8947** and an F1 score of **0.8926**, followed closely by SVM. Naive Bayes showed comparatively lower performance due to its simplicity and assumptions about data distribution.

	Model	Accuracy	F1 Score
0	Logistic Regression	0.894773	0.892619
1	Naive Bayes	0.669277	0.561027
2	SVM	0.877104	0.874498

Figure 6-1: Model Evaluation Metrics Table (Accuracy & F1 Score).

These results were visualized using bar plots for better clarity and comparison, highlighting the performance differences across the models.

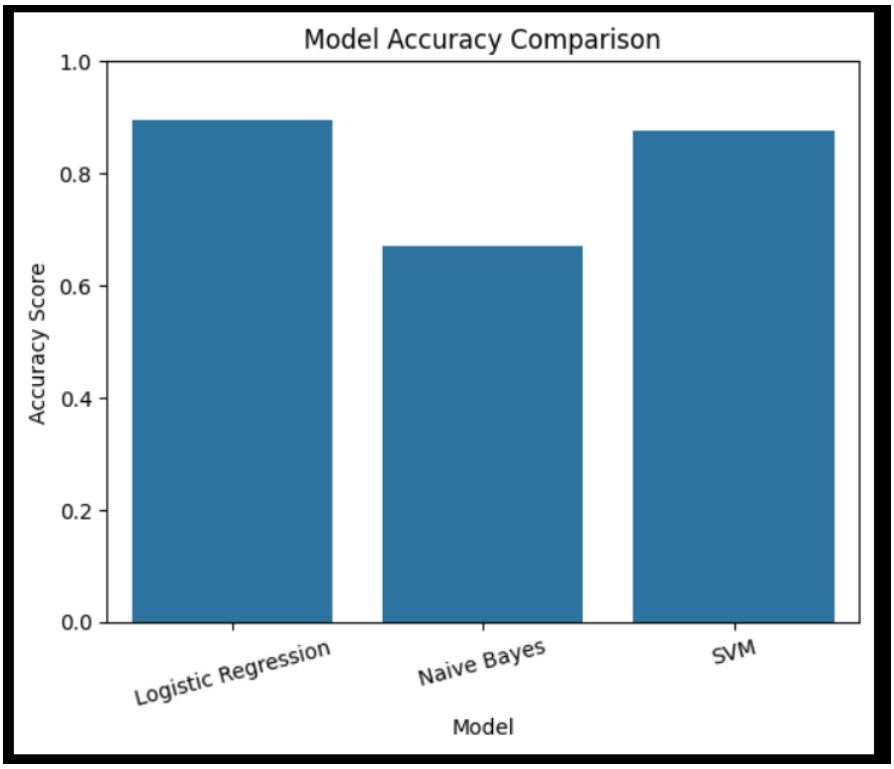

Figure 6-2: Bar Chart Comparing Model Accuracy.

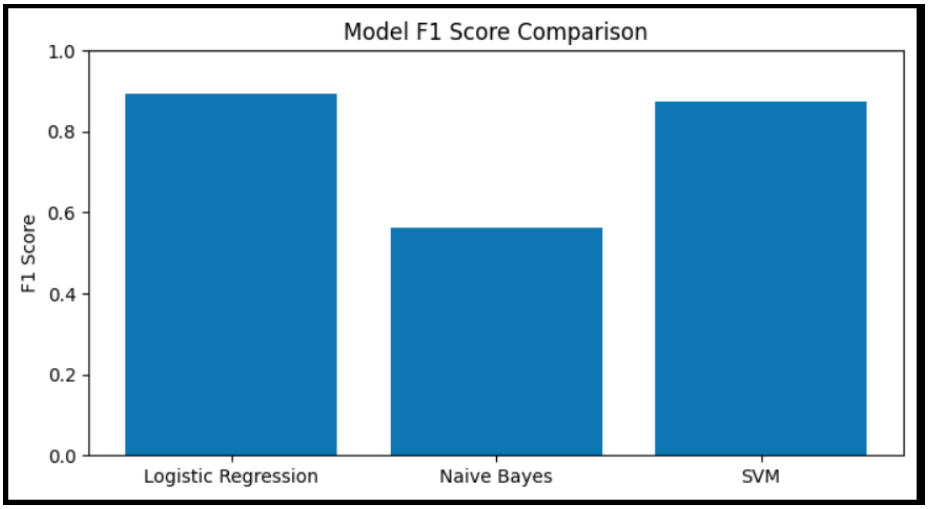

Figure 6-3: Bar Chart Comparing Model F1 Score.

6.2 Hyperparameter Tuning with GridSearchCV

Hyperparameter tuning plays a critical role in optimizing machine learning models, directly impacting their accuracy and generalization capabilities. To enhance performance, hyperparameter tuning was conducted for each algorithm using GridSearchCV (Hamzah

Dabool et al., 2024). Hyperparameter tuning is the process of finding the best settings for a ML model to improve its performance. Since hyperparameters are chosen before training, tuning involves testing different values to see which combination works best. The following key parameters were tuned:

- **Logistic Regression**: C, penalty, solver
- **Naive Bayes**: alpha
- **SVM**: C

```
--- Tuned Logistic Regression Evaluation ---
Accuracy: 0.9015279257657732
F1 Score: 0.9008471719428678

Classification Report:
              precision    recall  f1-score   support

    negative       0.81      0.66      0.73      8328
     neutral       0.81      0.86      0.83     25724
    positive       0.95      0.95      0.95     62615

    accuracy                           0.90     96667
   macro avg       0.86      0.82      0.84     96667
weighted avg       0.90      0.90      0.90     96667
```

Figure 6-4: Classification Report – Tuned Logistic Regression.

```
--- Tuned Naive Bayes Evaluation ---
Accuracy: 0.7544353295333465
F1 Score: 0.7459318318430289

Classification Report:
              precision    recall  f1-score   support

    negative       0.53      0.32      0.39      8328
     neutral       0.58      0.58      0.58     25724
    positive       0.84      0.89      0.86     62615

    accuracy                           0.75     96667
   macro avg       0.65      0.59      0.61     96667
weighted avg       0.74      0.75      0.75     96667
```

Figure 6-5: Classification Report – Tuned Naive Bayes.

```
--- Tuned SVM Evaluation ---
Accuracy: 0.8817590284171434
F1 Score: 0.8787750472823906

Classification Report:
              precision    recall  f1-score   support

    negative       0.74      0.56      0.64      8328
     neutral       0.79      0.80      0.79     25724
    positive       0.93      0.96      0.95     62615

    accuracy                           0.88     96667
   macro avg       0.82      0.77      0.79     96667
weighted avg       0.88      0.88      0.88     96667
```

Figure 6-6: Classification Report – Tuned SVM.

After tuning, significant improvements were observed, particularly for Logistic Regression and Naive Bayes.

	Model	Accuracy	F1 Score
0	Logistic Regression	0.901528	0.900847
1	Naive Bayes	0.754435	0.745932
2	SVM	0.881759	0.878775

Figure 6-7: Model Evaluation Metrics Table after Tuning (Accuracy & F1 Score).

6.2.1 Model Accuracy and F1 Score Comparison After Hyperparameter Tuning

The updated evaluation metrics, including **precision**, **recall**, **f1-score**, and **support**, were reported using the classification report. Logistic Regression again outperformed others after tuning, demonstrating its robustness and suitability for sentiment-driven product recommendations.

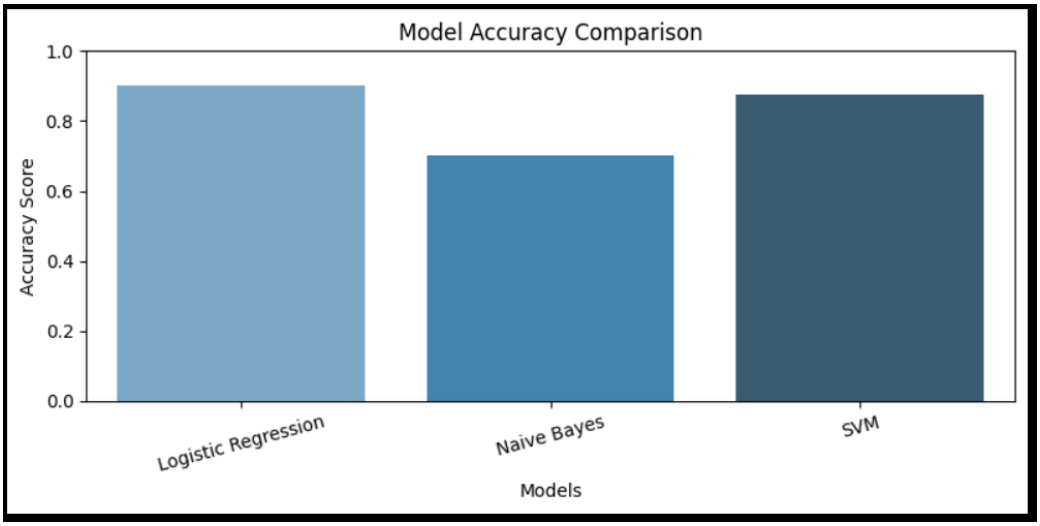

Figure 6-8: Bar Plot of Model Accuracy Scores Comparison After Hyperparameter Tuning.

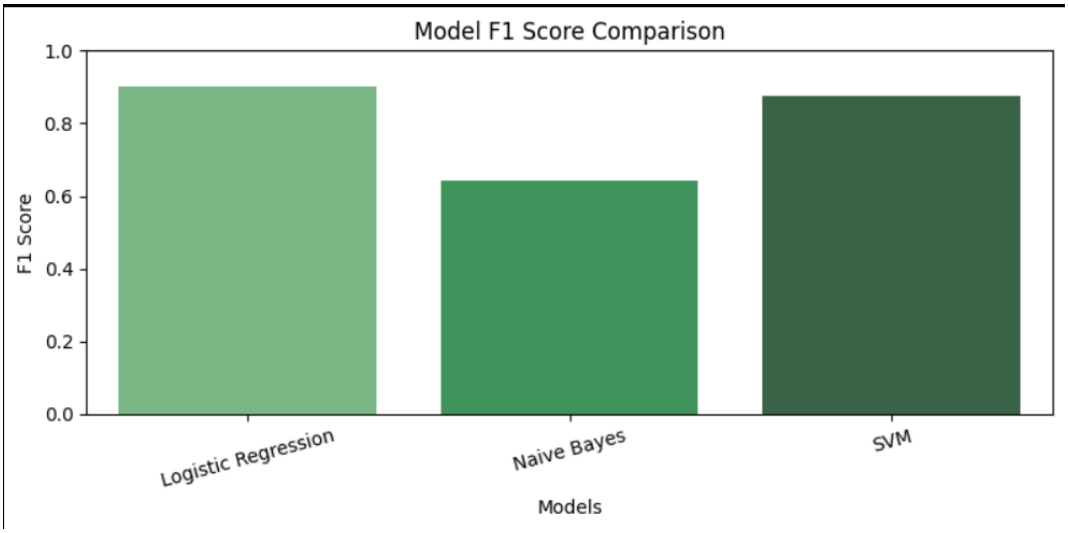

Figure 6-9: Bar Plot of Model F1 Scores Comparison After Hyperparameter Tuning.

From the visualizations and tables above, it is evident that **Logistic Regression** consistently outperformed the other models both before and after tuning, showing the highest accuracy and F1 score. **Naive Bayes**, despite tuning, remained the least performant due to its simplicity and assumptions that may not suit complex sentiment patterns. **SVM** showed strong performance close to Logistic Regression, making it a viable alternative.

6.3 Comparative Performance of Models

To summarize the effect of tuning, a comparative analysis was conducted by plotting accuracy and F1 score for all models—before and after tuning. These visualizations clearly show the performance gains achieved through proper hyperparameter optimization.

Model	Accuracy (Before)	Accuracy (After)	F1 Score (Before)	F1 Score (After)
Logistic Regression	0.894773	0.901528	0.892619	0.900847
Naive Bayes	0.669277	0.702732	0.561027	0.641555
SVM	0.877104	0.877104	0.874498	0.874498

Figure 6-10: Model Accuracy and F1 Score Comparison Before and after Hyperparameter Tuning.

Overall, the experimentation phase validated the importance of model selection and parameter tuning. Logistic Regression proved to be the most effective model, achieving the highest scores in both accuracy and F1 evaluation metrics, making it the best candidate for the final recommendation engine in this project.

7 Visualization Dashboard

The Visualization Dashboard serves as the interactive front-end interface of this project. It provides users with dynamic, real-time insights derived from sentiment analysis of Amazon product reviews in the Health and Personal Care category. The dashboard enables stakeholders—including product managers, marketers, and analysts—to explore key metrics and identify high-performing products and trends in customer feedback.

To support the analysis and interpretation of sentiment-driven insights from product reviews, a responsive and interactive visualization dashboard was developed using HTML, CSS, JavaScript, Chart.js, and Flask. The dashboard serves as a key interface, enabling users to visually explore customer sentiment, rating distributions, product performance, and store-level analytics derived from Amazon's Health and Personal Care product reviews.

The primary objectives of the dashboard are:

- To provide clear and interpretable visual representations of sentiment analysis results.
- To support real-time exploration of product review metrics.
- To assist decision-makers in identifying top-performing products and potential customer satisfaction issues.
- To ensure accessibility and responsiveness across devices.

The dashboard bridges the gap between data analysis and business decision-making by providing stakeholders with real-time, visual access to the processed review metrics.

7.1 Dashboard Component

The dashboard is composed of the following core visual and functional components:

- **Product Search Bar**: Allows users to filter and view data for individual products.
- **Interactive link:** Display the most reviewed product's insights with a single click
- **KPI Metrics Tiles**: Show statistics such as total reviews, number of positive/neutral/negative reviews, and number of 5-star products.
- **Sentiment Distribution Pie Chart**: Displays the proportion of positive, neutral, and negative sentiments.
- **Rating Distribution Bar Chart**: Summarizes how user ratings are spread across a 1–5 star scale.

- **Verified vs Unverified Review Pie Chart**: Highlights the proportion of verified purchases among reviews.
- **Helpful Votes Distribution Chart**: Illustrates how many reviews were marked as helpful.
- **Top Product Review Counts**: Displays the top 5 products with the highest positive and negative review volumes.
- **Scatter Plot – Price vs Rating**: Examines the relationship between product price and customer rating.
- **Average Rating per Price Range**: Groups ratings into price intervals to analyse value-perception.
- **Store-level Sentiment & Review Distribution**: Provides brand-level insights on customer sentiment and engagement.

7.2 Technical Stack

Component	Technology Used
Backend Framework	Python Flask
Frontend	HTML, CSS, JavaScript
Visualization	Chart.js, D3.js
Data Handling	Pandas
Sentiment Engine	TextBlob
Deployment (optional)	Render / PythonAnywhere (if applicable)

Table 7-1: Summary of Technologies Used in the Sentiment-Driven Product Recommendation System.

7.3 Screenshot and Deployment

To demonstrate the implemented system and its user interface, a web-based visualization dashboard was developed and integrated as the front-end component of the project. The dashboard enables users to interactively explore sentiment analysis results, review metrics, and sales insights derived from product review data.

The dashboard layout is divided into two primary sections:

- **Top Section**: Contains the product search functionality, KPI summary tiles (e.g., total reviews, 5-star counts, sentiment breakdown), and core sentiment and rating visualizations.

- **Bottom Section**: Displays detailed insights including helpful votes, top-reviewed products (positive/negative), price vs rating correlation, average rating by price range, and store-level sentiment distribution.

Figure 6.1 illustrates the top section of the dashboard, which provides search and summary functionalities. **Figure 6.2** presents the lower section, where deeper analytics and product-specific insights are visualized. Each chart includes interactive elements such as hover tooltips and export buttons, allowing users to download visuals in image format for further reporting. The dashboard also features a "⭐ Show Most Reviewed Product" shortcut link to quickly display analytics for the product with the highest review volume. The interface is designed with responsive behaviour using CSS media queries, making it accessible on both desktop and mobile devices. The charts are rendered using Chart.js, with real-time data interactions handled through JavaScript and Flask routes.

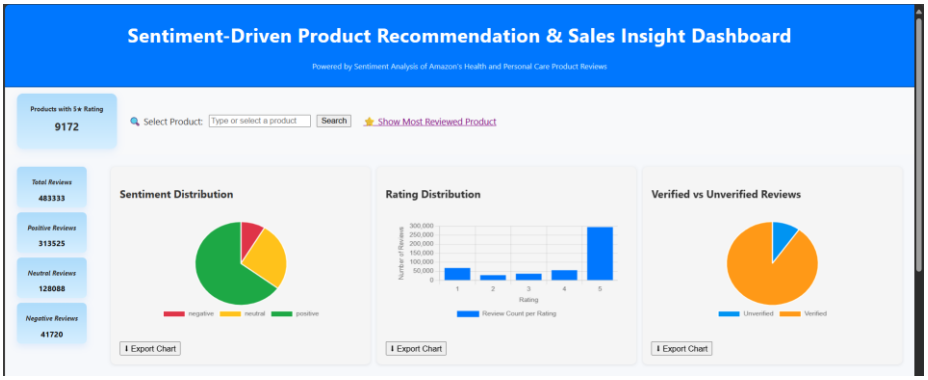

Figure 7-1: Dashboard Top Section — Product Search, KPI Tiles, Sentiment and Rating Charts.

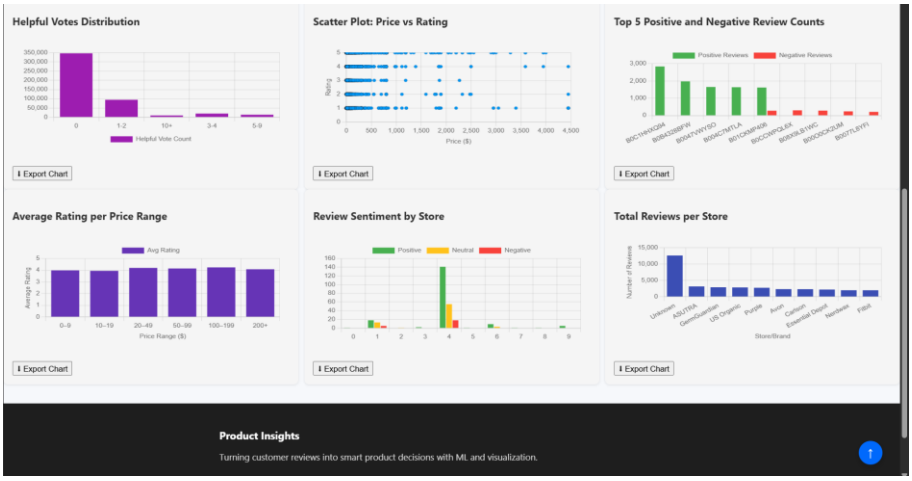

Figure 7-2: Dashboard Bottom Section — Helpful Votes, Price Trends, Store Analytics.

This figure demonstrates the product search interface used to retrieve recommendations for specific products. Users can select a product name from the search box and click the **Search** button to view detailed insights and suggestions based on sentiment analysis. The outcomes of this interaction are further explored in **Figure 6.4**, which presents a comprehensive breakdown of the selected product's sentiment profile and metadata.

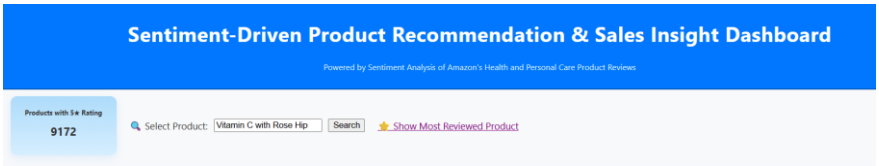

Figure 7-3: Interactive Product Search Interface with KPI Summary and Quick Access to Most Reviewed Product.

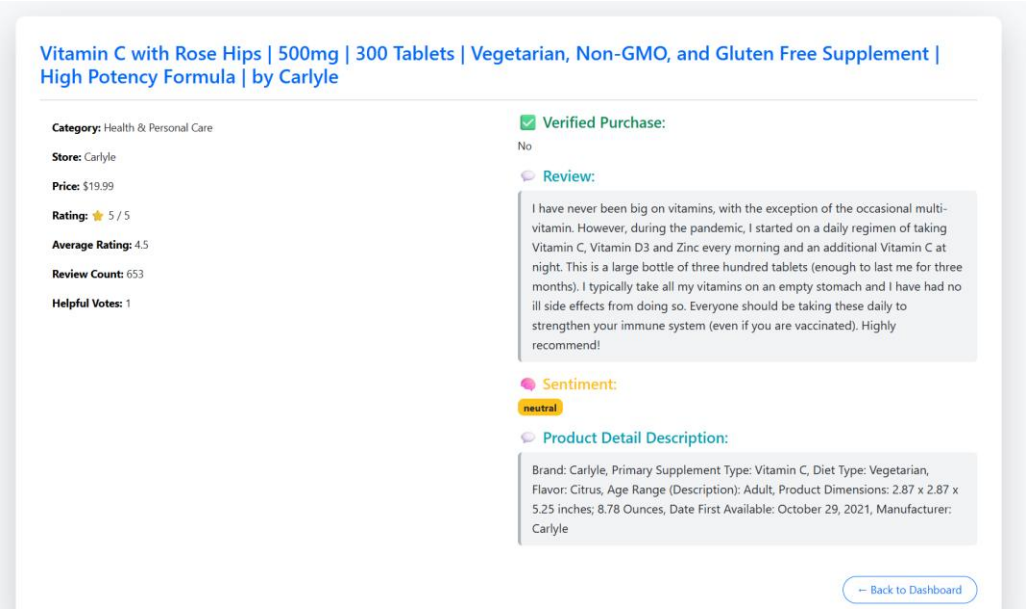

Figure 7-4: Detailed Sentiment View of Selected Product with Verified Purchase Status, User Review, and Descriptive Metadata.

Product ID	Rating	Sentiment	Sentiment Score	Review Count
B08SRPX53Z	4.402	positive	0.271	50944857
B08X9LB1WC	3.839	positive	0.170	35382314
B08KBQNDJC	4.141	positive	0.245	31722214
B004C7MTLA	4.734	positive	0.389	23469592
B08BQZHB9D	3.841	positive	0.218	21524205
B0001ZWPI4	4.265	positive	0.249	19077550
B0047VWYSO	4.663	positive	0.340	17505335
B00BR1FSU8	4.361	positive	0.307	14965902
B08K2MYBZY	4.457	positive	0.268	10196160
B08Z2PMHZ	4.081	positive	0.279	8944917

Figure 7-5: Most Reviewed Products — Ratings, Sentiment Scores, and Review Volumes Table.

Each chart in the dashboard supports interactive exploration through:

- **Tooltip hover effects** to reveal data point details
- **Real-time filtering** by product name via a dropdown/search bar
- **One-click export buttons** to download charts as image files
- **Shortcut navigation** to view the most reviewed product automatically

These features enhance usability and allow users to gain insights dynamically without needing to reload or manipulate raw data manually.

The dashboard is developed using the Flask web framework for the backend, and HTML, CSS, and JavaScript (Chart.js) for the frontend. Flask handles data routing and serves preprocesses sentiment data to the frontend through defined routes. The frontend leverages Chart.js to create responsive, animated visualizations that update in real-time. The visualization dashboard transforms raw sentiment outputs into clear, actionable insights through a highly

interactive and visually consistent interface. It supports strategic decision-making for product improvement and marketing by surfacing trends and feedback patterns at both product and brand levels. The successful integration of machine learning results into this real-time web dashboard highlights the practical value of combining data science with user-friendly design.

8 Future Directions and Recommendations

Although sentiment analysis, machine learning, and data visualization have been effectively integrated in this project to create a system for product suggestion and sales insight, there are a number of areas that might use more development and investigation:

1. **Adoption of Advanced Deep Learning Models**
   Future iterations of the system could explore deep learning architectures such as BERT or RoBERTa, which offer more nuanced understanding of language context and sentiment. These models could significantly improve classification accuracy, particularly in complex review texts.
2. **Real-Time Sentiment Monitoring and Personalization**

   Static datasets are handled by the existing system. To provide dynamic product recommendations and real-time sentiment dashboards customized for each user, future research might concentrate on combining real-time review streams with user behaviour data.
3. **Multilingual and Cross-Domain Expansion**

   The system would become more adaptable and suitable to international platforms if the model were expanded to handle several languages and other e-commerce domains (such as fashion, electronics, and travel).
4. **Incorporating Audio and Video Reviews**

   As voice-based and video reviews become more popular, multimodal sentiment analysis may be used in later iterations to glean information from spoken words, face expressions, and audio tone in addition to written text.
5. **Explainable AI and Model Transparency**

   Incorporating explainability tools such as SHAP or LIME can help visualize the decision-making process of ML models. This would improve transparency and trust,

especially for business users relying on system recommendations for strategic decisions.

6. **Enhanced Dashboard Features**

   Exportable reports, adjustable KPIs, and mobile-responsive design are possible future enhancements. The dashboard would be more useful and useful if it had user filters like sentiment threshold, date range, and product category.

These directions will not only strengthen the technical capabilities of the system but also broaden its practical applications in real-world e-commerce environments.

9 Conclusion

This project effectively integrated machine learning, natural language processing (NLP), and interactive data visualization to address important issues in e-commerce decision-making, resulting in an end-to-end sentiment-driven product recommendation and sales insight system. Analysing user-generated reviews from Amazon's Health & Personal Care category can help customers and business stakeholders by transforming unstructured textual data into organized, usable information. A comparison between the classification models Naive Bayes, Support Vector Machine (SVM), and Logistic Regression showed that Logistic Regression had the highest accuracy and F1-score, particularly after hyperparameter adjustment. This demonstrates how crucial model optimization and selection are to sentiment-based product evaluation.

By using interactive visual dashboards, the project translated raw review data into meaningful visual insights, enabling deeper understanding of customer sentiment and sales impact. This project shows how data science and front-end development can work together harmoniously, illustrating that sentiment analysis and visual storytelling may help close the gap between sophisticated analytics and user understanding. The system empowers smarter purchasing decisions for consumers and provides strategic business insights, showcasing a scalable and impactful solution in the competitive e-commerce landscape.

References

A. Al-Sulaiti, M. Mansour, H. Al-Yafei, S. Aseel, M. Kucukvar and N. C. Onat, "Using Data Analytics and Visualization Dashboard for Engineering, Procurement, and Construction Project's Performance Assessment," *2021 IEEE 8th International Conference on Industrial Engineering and Applications (ICIEA)*, Chengdu, China, 2021, pp. 207-211, doi: 10.1109/ICIEA52957.2021.9436728.

A. Morton, E. Marzban, G. Giannoulis, A. Patel, R. Aparasu and I. A. Kakadiaris, "A Comparison of Supervised Machine Learning Techniques for Predicting Short-Term In-Hospital Length of Stay among Diabetic Patients," *2014 13th International Conference on Machine Learning and Applications*, Detroit, MI, USA, 2014, pp. 428-431, doi: 10.1109/ICMLA.2014.76.

F. Gürcan, "Multi-Class Classification of Turkish Texts with Machine Learning Algorithms," *2018 2nd International Symposium on Multidisciplinary Studies and Innovative Technologies (ISMSIT)*, Ankara, Turkey, 2018, pp. 1-5, doi: 10.1109/ISMSIT.2018.8567307.

H. Dabool, H. Alashwal, H. Alnuaimi, A. Alhouqani, S. Alkaabi and A. Al Ahbabi, "Comparative Analysis of Hyperparameter Tuning Methods in Classification Models For Ensemble Learning," *2024 7th International Conference on Algorithms, Computing and Artificial Intelligence (ACAI)*, Guangzhou, China, 2024, pp. 1-5, doi: 10.1109/ACAI63924.2024.10899492.

H. M. W. T. C. B. Samarakoon, A. U. Ovitigala, G. A. T. T. Wijesinghe, K. M. L. P. Weerasinghe, G. K. B. V. Karunathilake and I. Weerathunga, "Facial Skincare Product Suggestion with Product Popularity and Post Recommendation Care," *2023 4th International Informatics and Software Engineering Conference (IISEC)*, Ankara, Turkiye, 2023, pp. 1-6. Available at: https://doi.org/10.1109/IISEC59749.2023.10391033.

H. S. and R. Ramathmika, "Sentiment Analysis of Yelp Reviews by Machine Learning," *2019 International Conference on Intelligent Computing and Control Systems (ICCS)*, Madurai, India, 2019, pp. 700-704, doi: 10.1109/ICCS45141.2019.9065812.

Hake, J. Pujari, L. S. V and N. K. H. S, "Sentiment Analysis-Based Product Review System for Enhanced Recommendations," 2025 International Conference on Artificial Intelligence and Data Engineering (AIDE), Nitte, India, 2025, pp. 618-623, doi: 10.1109/AIDE64228.2025.10986799.

J. K. Adarsh, V. T. Sreedevi and D. Thangavelusamy, "Product Review System With BERT for Sentiment Analysis and Implementation of Administrative Privileges on Node-RED," in *IEEE Access*, vol. 11, pp. 65968-65976, 2023, doi: 10.1109/ACCESS.2023.3275738.

J. P. U. S. D. Jayakody and B. T. G. S. Kumara, "Sentiment analysis on product reviews on twitter using Machine Learning Approaches," *2021 International Conference on Decision Aid Sciences and Application (DASA)*, Sakheer, Bahrain, 2021, pp. 1056-1061. Available at: https://doi.org/10.1109/DASA53625.2021.9682291.

J. Panduro-Ramirez, "Sentiment Analysis in Customer Reviews for Product Recommendation in E-commerce Using Machine Learning," *2024 International Conference on Advances in Computing, Communication and Applied Informatics (ACCAI)*, Chennai, India, 2024, pp. 1-5. Available at: https://doi.org/10.1109/ACCAI61061.2024.10602027.

Overview of Machine Learning, in *Machine Learning for Industrial Applications* , Wiley, 2024, pp.1-20, doi: 10.1002/9781394268993.ch1.

R. Marappan and S. R, "New Machine Learning Model to Movie Recommender and Sentiment Analysis," *2023 International Conference on Advances in Computation, Communication and Information Technology (ICAICCIT)*, Faridabad, India, 2023, pp. 363-366. Available at: https://doi.org/10.1109/ICACICIT60255.2023.10465888.

R. V. Karthik, S. Ganapathy and A. Kannan, "A Recommendation System for Online Purchase Using Feature and Product Ranking," *2018 Eleventh International Conference on Contemporary Computing (IC3)*, Noida, India, 2018, pp. 1-6. Available at: https://doi.org/10.1109/IC3.2018.8530573.

S. Bouguern and M. Aizi, "Implementation of Interactive Dashboard Visualization for Startup's Inventory Management: Beauty Startup Case Study," *2024 International Conference of the African Federation of Operational Research Societies (AFROS)*, Tlemcen, Algeria, 2024, pp. 1-5, doi: 10.1109/AFROS62115.2024.11036879.

S. Saraswat, S. Bhardwaj, S. Vashistha and R. Kumar, "Sentiment Analysis of Audio Files Using Machine Learning and Textual Classification of Audio Data," *2023 6th International Conference on Information Systems and Computer Networks (ISCON)*, Mathura, India, 2023, pp. 1-5, doi: 10.1109/ISCON57294.2023.10112195.

S. Shin, I. Na and N. Elmqvist, "Drillboards: Adaptive Visualization Dashboards for Dynamic Personalization of Visualization Experiences," in *IEEE Transactions on Visualization and Computer Graphics*, doi: 10.1109/TVCG.2025.3542606.

S. Zahoor and R. Rohilla, "Twitter Sentiment Analysis Using Machine Learning Algorithms: A Case Study," *2020 International Conference on Advances in Computing, Communication & Materials (ICACCM)*, Dehradun, India, 2020, pp. 194-199, doi: 10.1109/ICACCM50413.2020.9213011.

V. Gupta, V. K. Mishra, P. Singhal and A. Kumar, "An Overview of Supervised Machine Learning Algorithm," *2022 11th International Conference on System Modeling & Advancement in Research Trends (SMART)*, Moradabad, India, 2022, pp. 87-92, doi: 10.1109/SMART55829.2022.10047618.

Venkatesh and K. V. Ranjitha, "Classification and Optimization Scheme for Text Data using Machine Learning Naïve Bayes Classifier," *2018 IEEE World Symposium on Communication Engineering (WSCE)*, Singapore, 2018, pp. 33-36, doi: 10.1109/WSCE.2018.8690536.