AI-Powered Product Recommendation System

team members:Hessah-Ghadah-Shatha

Introduction:

In today’s digital age, customer reviews play a crucial role in shaping purchasing decisions and improving products. However, with thousands of reviews available across multiple platforms, manually analyzing them is time-consuming, inefficient, and often impractical. To address this challenge, this project focuses on developing an Automated Customer Review System powered by Natural Language Processing (NLP) models.
The goal of this project is to aggregate, analyze, and summarize customer feedback from various sources using advanced NLP techniques. Key tasks include classifying reviews by sentiment, clustering products based on customer feedback, and leveraging generative AI to create concise, actionable summaries and recommendation articles. By automating the review analysis process, this system aims to provide businesses with valuable insights to enhance their products and services, while helping consumers make informed purchasing decisions.

This report outlines the project’s objectives, methodologies, and outcomes, demonstrating how NLP can transform the way we analyze and utilize customer reviews.

Problem Statement:

With thousands of reviews available across multiple platforms, manually analyzing them is inefficient. This project seeks to automate the process using NLP models to extract insights and provide users with valuable product recommendations.

Project Goals:

- · Classify reviews into positive, neutral, or negative
- · Cluster products into 4–6 meaningful categories
- Summarize reviews for each category into article-like recommendations

. Source and Overview:

The dataset used in this project is based on a combination of multiple Amazon Product Review datasets, merged to create a richer and more diverse corpus. These datasets were obtained from publicly available sources such as Hugging Face and Kaggle, covering a wide range of product categories and review formats.

Merging datasets allowed us to increase data diversity, product category coverage, and ensure that each downstream task—classification, clustering, and summarization—had sufficient and balanced data. The rationale behind combining these datasets will be discussed in more detail in the relevant sections of the report.

Each review entry includes:

- review_body: The main text content of the customer review.
- star_rating: A numeric score (1–5) assigned by the reviewer.
- product_title: The title of the reviewed product.
- product_category: A label denoting the general category of the product.

Dataset

We used a dataset that has over 34,000 consumer reviews for Amazon products like the Kindle, Fire TV Stick, and more provided by Datafiniti's Product Database. The dataset includes basic product information, rating, review text, and more for each product.

Preprocessing:

We prepare raw data for analysis by cleaning, transforming, and structuring it properly. To ensure consistent, high-quality inputs that lead to more accurate and reliable results in any data task.
For our project we did so many pre-processing techniques which I'll list below:

1. Import necessary libraries

First, we imported libraries. which gather all the digital tools needed to organize, analyze, and visualize text data like a chef collects kitchen tools before cooking. pandas organizes data into tables, train_test_split divides data for practice/testing, and TfidfVectorizer converts text to number scores based on word importance. seaborn and matplotlib create charts/visualizations, while re and string help clean and fix messy text before analysis.

2. Load the dataset

We wrote code that downloads Amazon product reviews from Kaggle (a data website) and loads them into a table for analysis, like opening a spreadsheet.

3. Combine

Since there are three data sets in the primary data set, we decided to combine all of them and work with it as one dataframe.
To prepare the dataset for analysis, we first merged three individual review datasets into a single DataFrame using `pandas.concat()`. We then removed rows containing missing values with `dropna()` to maintain data integrity. After cleaning, we verified the structure of the combined dataset by inspecting the first few rows and confirming the presence of key columns such as `reviews.rating` and `reviews.text`, which are essential for downstream sentiment analysis and review summarization tasks.

4. Data Exploration

We performed data exploration that gives a quick summary of the data.
It shows what's inside (column names, data types, and size).
It checks numbers and missing values (averages, ranges, and blanks).

5. Map star ratings to sentiment classes

In preparation for the review classification task, we converted star ratings to sentiment classes using the following rule:

Star Rating	Sentiment Class
1 – 2	Negative
3	Neutral
4 – 5	Positive

A new column, sentiment_label, was added to store this mapping, enabling training of supervised classification models.

6. Balance Ratings (1–5) and Map Sentiment:

To address class imbalance in the dataset, we implemented a technique called upsampling to ensure equal representation of each review rating (from 1 to 5 stars). First, we mapped the `reviews.rating` column to corresponding sentiment labels—"Negative" for ratings ≤2, "Neutral" for rating 3, and "Positive" for ratings ≥4. Next, we grouped the dataset by individual rating values and determined the size of the largest group. Each group was then resampled (with replacement) to match this maximum size, thereby equalizing the number of samples per class. This resampling helps prevent the model from becoming biased toward more frequent classes during training. Finally, we combined the upsampled groups into a single balanced dataset, shuffled it to eliminate ordering bias, and verified the distribution of both review ratings and sentiment labels. This preprocessing step is crucial to ensure fairness and improved generalization in sentiment classification tasks.

7. Final Clean-Up & Sentiment Mapping

After balancing the dataset, we performed additional preprocessing to ensure the sentiment data was clean and ready for analysis or modeling. First, we handled any missing values in the `sentiment` column by replacing them with the label "Neutral", thereby avoiding issues during further processing. Then, we created a new column called `sentiment_numeric` by mapping the textual sentiment labels ("Negative", "Neutral", and "Positive") to numeric values (-1, 0, and 1, respectively). This numeric representation is particularly useful for training machine learning models that require numerical input. We also verified the presence of the new column and printed the unique values in the sentiment fields to confirm consistency. Finally, we reviewed the distribution of both `reviews.rating` and the new `sentiment_numeric` column to ensure the integrity of the data. These steps were essential to maintain data quality and prepare the dataset for efficient model training and evaluation.

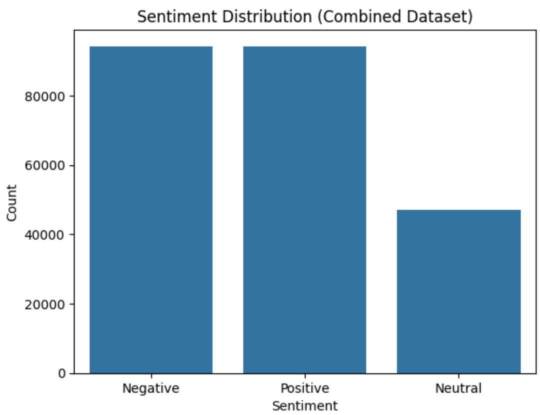
8. Compute Correlation Matrix

To explore the relationship between review ratings and the sentiment labels, we conducted a correlation analysis using the cleaned and balanced dataset. We selected two numerical columns—`reviews.rating` and the newly created `sentiment_numeric`—and computed their Pearson correlation coefficient. This allowed us to quantify the linear relationship between star ratings and sentiment polarity (encoded as -1 for Negative, 0 for Neutral, and 1 for Positive). The resulting correlation matrix was visualized using a heatmap, which clearly highlights the strength and direction of the relationship. As shown in the graph below, the correlation is expected to be strongly positive, confirming that higher ratings are closely associated with more positive sentiment labels. This analysis provides further validation that the sentiment mapping aligns well with the original numeric ratings.

(as shown in Figure 1)

9. Visualize Sentiment Distribution

to gain a clearer understanding of the class distribution within the dataset, we visualized the frequency of each sentiment category—Positive, Neutral, and Negative—using a bar chart. This was achieved with a `countplot`, which displays the number of reviews assigned to each sentiment label in the balanced dataset. As the data had been intentionally balanced through upsampling, the resulting chart confirms that each sentiment class is equally represented. This visual verification reinforces that the dataset is suitable for training sentiment classification models without introducing bias toward any particular class. **(as shown in Figure 2)**



This script balances the dataset based on `reviews.rating` (values from 1 to 5), not directly on sentiment. Since sentiment is derived from these ratings:

- Negative sentiment comes from ratings 1 and 2 → combined, this doubles the sample size

- Neutral sentiment comes only from rating 3 → single size
- Positive sentiment comes from ratings 4 and 5 → also doubled

As a result, the dataset ends up imbalanced in terms of sentiment: Neutral has half as many samples as Negative and Positive.

10. Word Clouds for Sentiment Categories(-Stemming. -Generate Word Clouds for Sentiment Categories)

To gain qualitative insights into the most frequently used words across different sentiment categories, we generated word clouds for Positive, Negative, and Neutral reviews. After grouping and concatenating all review texts by sentiment, we used the `WordCloud` library to visualize the most prominent terms. A set of custom stopwords was defined to filter out common, non-informative words (e.g., "the", "and", "is") and improve the clarity of the visualizations.

Each word cloud highlights the top 200 most frequent words in that sentiment group, with larger font sizes representing higher frequency. This visualization helps identify commonly used expressions and terms associated with each sentiment type—for example, positive reviews often include words like *"love"*, *"excellent"*, or *"great"*, while negative ones might feature terms like *"disappointed"* or *"poor"*. These visual insights can be useful for understanding customer feedback and validating the effectiveness of sentiment labeling.**(as shown in Figure 3)**

11. Split the data into train and test sets

To gain qualitative insights into the most frequently used words across different sentiment categories, we generated word clouds for Positive, Negative, and Neutral reviews. After grouping and concatenating all review texts by sentiment, we used the `WordCloud` library to visualize the most prominent terms. A set of custom stopwords was defined to filter out common, non-informative words (e.g., "the", "and", "is") and improve the clarity of the visualizations.

Each word cloud highlights the top 200 most frequent words in that sentiment group, with larger font sizes representing higher frequency. This visualization helps identify commonly used expressions and terms associated with each sentiment type—for example, positive reviews often include words like *"love"*, *"excellent"*, or *"great"*, while negative ones might feature terms like *"disappointed"* or *"poor"*. These visual insights can be useful for understanding customer feedback and validating the effectiveness of sentiment labeling.

Large Data

1. Upload CSV File and Display First 5 Rows:

To begin the analysis, the dataset was uploaded directly from the local device into the Google Colab environment using the `files.upload()` method from the `google.colab` module. Once uploaded, the filename was retrieved and passed to `pandas.read_csv()` to load the data into a DataFrame. This method allows for quick and flexible data import in a cloud-based notebook. A preview of the first five rows (`df.head()`) was displayed to confirm the file was successfully read and to ensure the data structure was as expected before proceeding with further preprocessing steps.

2. upload the primary dataset to combined them together

3. Create a Balanced Sentiment Dataset from Multiple Amazon Review Sources and using larger data

In this step, we combined multiple Amazon product review datasets to create a unified and balanced dataset for sentiment analysis. Three smaller datasets containing reviews were first loaded and merged. We removed entries with missing `reviews.rating` or `reviews.text` values to ensure data quality. To facilitate sentiment analysis, we defined a mapping function that converts numerical ratings into sentiment labels: ratings ≤2 were classified as Negative, 3 as Neutral, and ≥4 as Positive.

We then extracted all Positive reviews from the merged dataset and counted their occurrences. Next, we loaded a fourth dataset, All_Beauty, which was either read from a CSV file or, if missing, converted from a JSONL format. This dataset was cleaned similarly, and the sentiment labels were applied using the same mapping logic.

After verifying the counts of Neutral and Negative reviews from the All_Beauty dataset, we determined the smallest available count among the three sentiment categories to perform balanced sampling. An equal number of reviews from each sentiment group were randomly selected and concatenated to form the final, balanced dataset. This dataset included only the essential columns—`reviews.rating`, `reviews.text`, and `sentiment`—and was saved as `final_reviews_dataset.csv`.

This balanced dataset ensures fair representation of sentiment classes, which is crucial for training reliable and unbiased sentiment classification models.

4. Clean Sentiment Column and Prepare for Modeling

This part of the process focuses on cleaning and structuring the dataset for analysis or machine learning.

1. **Handling Missing Values**: Missing sentiment values are replaced with 'Neutral' to ensure completeness.

2. **Mapping Sentiment to Numeric Values**: Sentiment labels ('Negative', 'Neutral', 'Positive') are converted into numeric values (-1, 0, 1) for compatibility with machine learning models.

3. **Verification**: The dataset's structure is checked by verifying column names, unique sentiment values, and the distribution of review ratings and sentiments to ensure balance and consistency.

These steps ensure that the dataset is clean, complete, and ready for further analysis or machine learning tasks.

5. Undersample Reviews to Balance Ratings

This section of the process focuses on balancing the dataset by adjusting the distribution of review ratings.

1. **Loading the Dataset**: The dataset is loaded from a CSV file containing Amazon product reviews. The goal is to ensure the review ratings are well-distributed across the sentiment categories.

2. **Identifying the Smallest Group**: The code checks the distribution of review ratings and identifies the category with the fewest samples. This minimum count helps guide the balancing process.

3. **Undersampling**: To ensure an equal number of samples across all rating categories, the code randomly selects a number of samples equal to the smallest category for each rating group. This process ensures that the dataset is balanced, eliminating potential bias in favor of the majority class.

4. **Cleaning up the DataFrame**: After grouping and undersampling, the index of the DataFrame is reset for a clean and well-organized dataset.

5. **Saving the Balanced Dataset**: The balanced dataset is saved to a new CSV file, ensuring that the data is ready for analysis or further processing without any rating imbalances.

This step is crucial for preparing the data for machine learning models or any statistical analysis, where a balanced dataset improves performance and accuracy.

Sentiment Classification

1-Using DistilRoBERTa

1. Overview

This project presents an advanced sentiment classification model trained on a large dataset of Amazon product reviews. The goal is to automatically categorize customer feedback into three sentiment classes: Negative, Neutral, and Positive. By leveraging state-of-the-art natural language processing techniques using DistilRoBERTa, the model achieves high performance with strong generalization across unseen data.

2. Dataset Description

The dataset was constructed by merging three publicly available CSV files containing Amazon reviews:

- `1429_1.csv`
- `Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv`
- `Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv`

Each file was loaded and combined using Pandas, followed by thorough data cleaning and preprocessing. Reviews were labeled based on their associated star rating:

- Ratings 1–2 → Negative
- Rating 3 → Neutral
- Ratings 4–5 → Positive

After preprocessing, the final dataset consisted of over 90,000 labeled reviews with a relatively balanced distribution across the three sentiment classes. Special attention was given to ensuring consistency and removing duplicates and incomplete entries to improve model quality.

3. Model and Training

The model used is DistilRoBERTa, a lightweight and efficient version of RoBERTa. The training process included stratified splitting of data, handling class imbalance using weighted sampling, and early stopping to prevent overfitting. Training was conducted using the Hugging Face Trainer API with custom evaluation metrics and detailed logging.

4. Evaluation Results

The final model demonstrated excellent performance on the test set with the following metrics:

- Accuracy: 94.92%
- Precision / Recall / F1-Score per class:
  - Negative: Precision 0.77 | Recall 0.75 | F1 0.76
  - Neutral: Precision 0.55 | Recall 0.18 | F1 0.27
  - Positive: Precision 0.96 | Recall 0.99 | F1 0.98

The model excels particularly in identifying positive reviews with high confidence and robustness.

5. Confusion Matrix

The following confusion matrix visualizes the model's prediction distribution across classes:

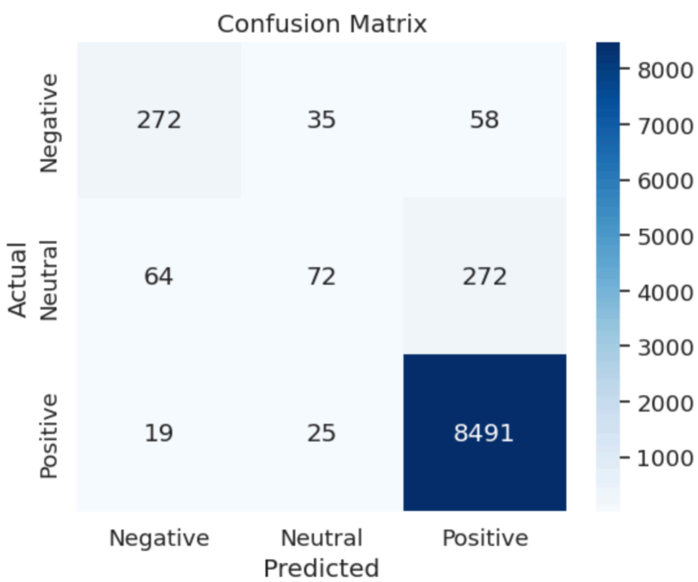


Figure 4: Confusion Matrix for 3-Class Sentiment Classification

6. Training Metrics

Training and validation metrics were tracked throughout the process. The graphs below demonstrate smooth convergence with minimal overfitting:

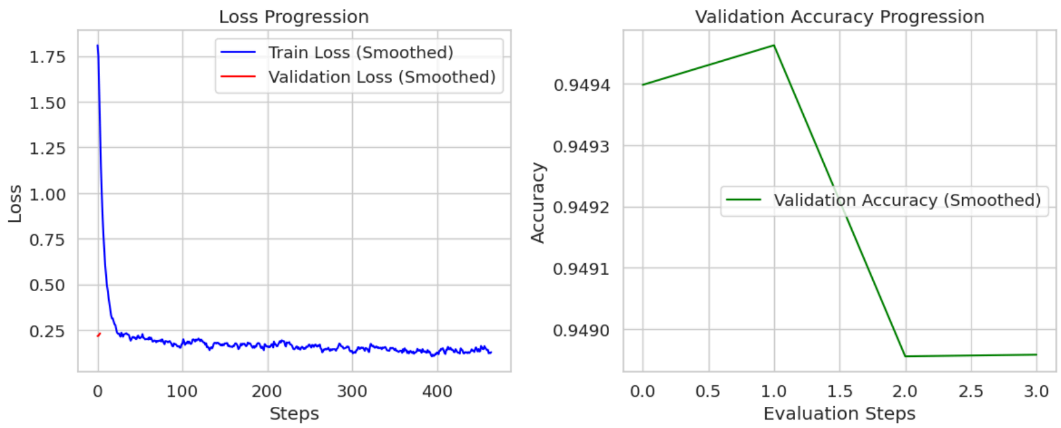


Figure 5: Training Loss and Validation Accuracy Progression

2-Using distilroberta-base-large

Sentiment Analysis Model Report

Using distilroberta-base on Amazon Reviews Dataset

1. General Setup

- Model Name: distilroberta-base
- Task Type: Multi-class Sentiment Classification (Negative, Neutral, Positive)
- Dataset: balanced_reviews.csv (Amazon Reviews)
- Number of Classes: 3
- Data Split: 80% training - 20% testing
- Environment: Google Colab with GPU and FP16 support
- Epochs: 3
- Max Sequence Length: 128 tokens
- Batch Size: 16 samples per batch

2. Preprocessing Steps

- Successfully loaded over 13,500 records
- Cleaning steps:
  - Removed NaN values and short texts
  - Removed duplicates
  - Filtered out noisy text
  - Mapped numerical ratings to sentiment labels:
    - 0 = Negative (rating ≤ 2)
    - 1 = Neutral (rating = 3)
    - 2 = Positive (rating > 3)

3. Training Results

Epoch	Train Loss	Validation Loss	Accuracy	F1 Score	Precision	Recall
1	0.3854	0.3557	0.8452	0.8412	0.8391	0.8452
2	0.2934	0.3493	0.8482	0.8431	0.8486	0.8482
3	0.2260	0.3754	0.8515	0.8499	0.8477	0.8515

4. Visualizations

4.1 Loss Curve per Epoch

Training loss consistently decreased across epochs with stable validation loss.

MODEL TRAINING

[10182/10182 15:14, Epoch 3/3]

Epoch	Training Loss	Validation Loss	Accuracy	F1	Precision	Recall
1	0.385400	0.355656	0.845207	0.841170	0.839087	0.845207
2	0.293400	0.349316	0.848154	0.848315	0.848571	0.848154
3	0.226000	0.375242	0.851544	0.849091	0.847681	0.851544

```
***** train metrics *****
  epoch                    =        3.0
  total_flos               =  5023310GF
  train_loss               =     0.3175
  train_runtime            = 0:15:14.40
  train_samples_per_second =    178.112
  train_steps_per_second   =     11.135
```

4.2 Validation Accuracy Progression

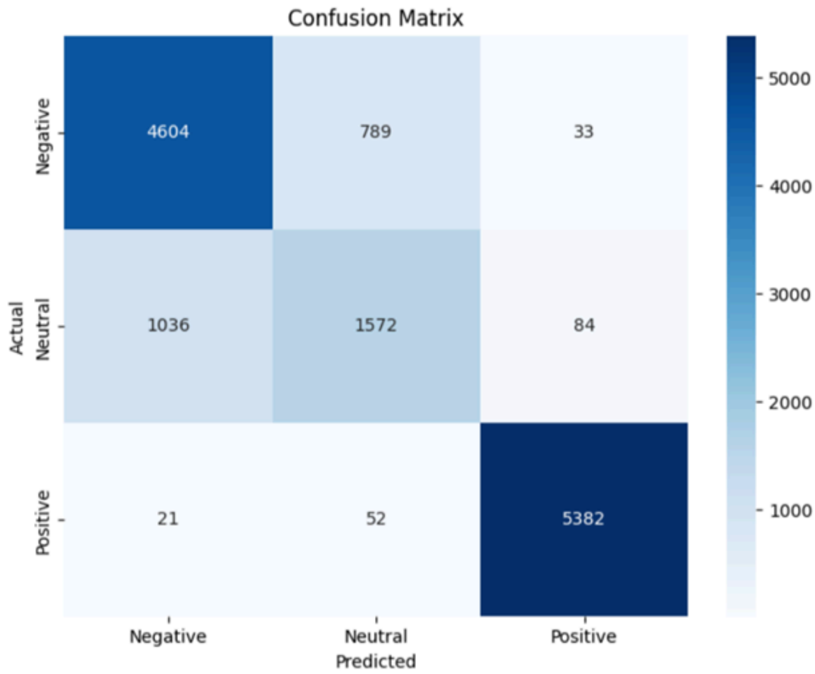
Validation accuracy increased steadily from 84.5% to 85.15%.

5. Model Evaluation

Classification Report:
- Negative: Precision 0.81, Recall 0.85, F1 Score 0.83 (Support: 5426)
- Neutral: Precision 0.65, Recall 0.58, F1 Score 0.61 (Support: 2692)
- Positive: Precision 0.98, Recall 0.99, F1 Score 0.98 (Support: 5455)

Overall Accuracy: 0.8515 (on 13,573 samples)

Confusion Matrix:



6.Evaluation Summary

- Test Accuracy: 0.8515
- F1 Score: 0.8499
- Test Loss: 0.3754
- Precision: 0.8477
- Recall: 0.8515
- Inference Time: ~14 seconds
- Speed: ~948 samples/sec

- Archived File: sentiment_model.zip
- Downloaded using files.download() from Colab

Chart Analysis: Sentiment Analysis Model

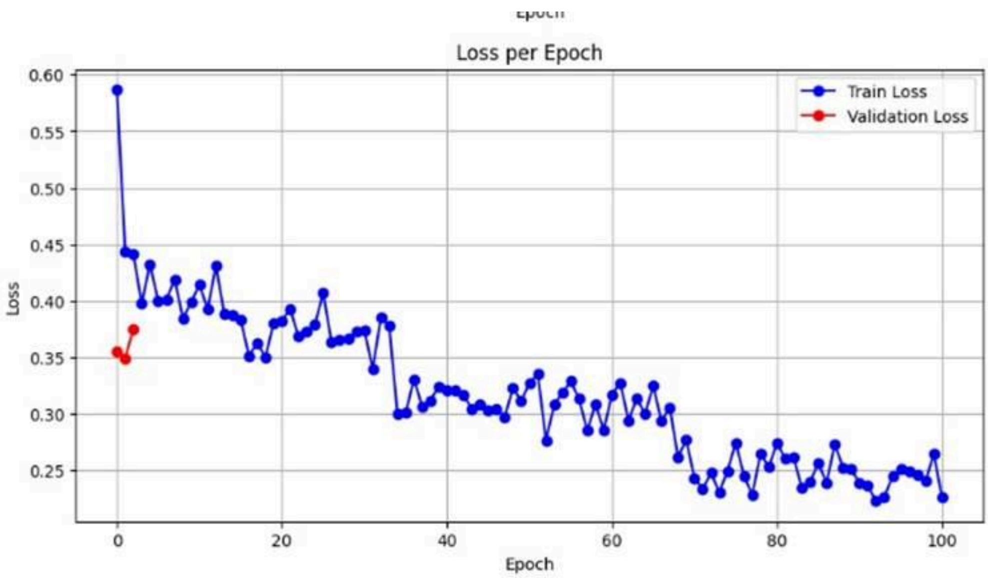
1. Loss per Epoch

This chart shows how the model's loss evolved over training epochs.

- The X-axis represents the number of epochs (training iterations).
- The Y-axis represents the loss value.

Observations:
- The blue line indicates the training loss, which steadily decreases as training progresses.
- The red points represent the validation loss recorded per epoch.
- A clear downtrend in training loss suggests effective model learning.

Conclusion:
- The model learns consistently and does not show signs of overfitting.
- Validation loss is stable but limited to few points (likely due to epoch-based evaluation).

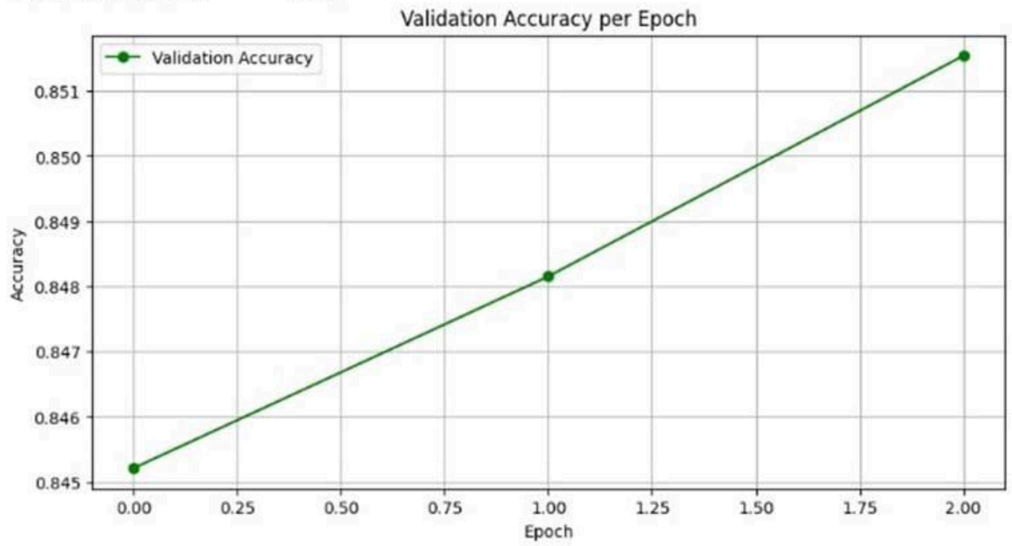


2. Validation Accuracy per Epoch
This chart shows the model’s accuracy on the validation set during training.

- The X-axis shows the epoch number.
- The Y-axis shows the validation accuracy.

Observations:
- The green line demonstrates a steady increase in validation accuracy over epochs.
- There are no performance drops or overfitting signs in early training.

Conclusion:
- The model is generalizing well on unseen data.
- More epochs may further improve performance.



Model Comparison Table

Metric	Model A: DistilRoBERTa uses the larger Dataset	Model B: distilroberta-base uses the Primary Dataset
Accuracy	85.15%	94%
Precision	Negative 81% Neutral 65% Positive 98%	Negative 71% Neutral 39% Positive 96%
Recall	Negative 85% Neutral 58% Positive 99%	Negative 68% Neutral 12% Positive 99%
F1-Score	Negative 83% Neutral 61% Positive 98%	Negative 70% Neutral 19% Positive 98%
Training Time	Longer	Faster

Clustering model

To address the task of simplifying product categories,wel developed a clustering solution aimed at grouping detailed product types into 4–6 intuitive meta-categories. The approach began with a thorough analysis of the dataset to understand the distribution and semantics of product categories represented in the reviews.

Using both domain knowledge and data-driven techniques, I identified logical groupings based on product functionality and consumer intent. This process involved text preprocessing, feature extraction (e.g., TF-IDF or embeddings), and unsupervised learning techniques such as clustering algorithms to arrive at coherent meta-categories.

We've tried many models and clustering algorithms and performed the clustering so many columns. Due to the large number of experiments, we did not have time to document all of our experiences. This is the best result we got.

Libraries used for clustering:

- Data handling (pandas, numpy)
- Text processing (re, spacy, STOP_WORDS)
- Embedding generation (SentenceTransformer)
- Clustering (KMeans, silhouette_score)
- Dimensionality reduction (UMAP)
- Visualization (matplotlib)
- Progress tracking (tqdm)
- Logging and file operations (logging, os)

1- Setup Kaggle API and Download & Extract Dataset and then Loads the dataset into a Pandas DataFrame.

2-Preprocess
Removes rows with missing values in key columns (name, categories, reviews.text) to ensures the dataset has complete information for analysis. For that we used function called:
advanced_clean, it performs advanced text cleaninig and preprocessing using **spaCy**, a powerful NLP library. spaCy is small English language model (en_core_web_sm), which includes:

- Tokenization
- Part-of-speech (POS) tagging
- Lemmatization
- Dependency parsing
- Named entity recognition (NER)

In our case, we did **Basic Text Cleaning** that **Removes** URLs and Non-alphabetic characters. Also, **Converts text to lowercase**.
We did **Phrase Detection** using **spaCy's** Matcher to detect common **phrases** (noun-noun, adjective-noun, verb-noun combinations). It helps retain meaningful multi-word terms (e.g., "bluetooth speaker" → bluetooth_speaker).
For **Extract and Join Phrases** It's detected with underscores (_). Also, Uses **lemmatization** (lemma_) to normalize words (e.g., "batteries" → "battery"). We did **Token Processing** to Refine individual words (tokens) by:

1. **Removing noise** (stopwords, punctuation, short words).
2. **Standardizing words** (e.g., "batteries" → "battery").
3. **Keeping only meaningful terms**.

This function returns cleaned tokens and detected phrases merged into a single string.

3-Embedding Generation
We used all-MiniLM-L6-v2 (a compact but powerful sentence transformer model) to convert ‘’ into numerical vectors (embeddings) that capture semantic meaning, enabling algorithms to process and analyze product data mathematically.

4-Dimensionality Reduction
Reducing embeddings speeds up processing and cuts memory/storage costs while preserving most of their usefulness. It trades a tiny drop in accuracy for major efficiency gains in search, clustering, or other tasks. We used UMAP (Uniform Manifold Approximation) to reduce from 384 to 64 dimensions. It is necessary to preserves cosine similarity relationships between embeddings.
To determine the optimal number of clusters for K-Means clustering we used two evaluation metrics, silhouette score and davies bouldin score.
silhouette_score: Measures how similar objects are within clusters vs other clusters (higher=better).
davies_bouldin_score: Measures cluster separation (lower=better).

5-K-means
We performed a function (find_optimal_clusters) that tests cluster counts from 4-6(as required) and calculates both evaluation scores and chooses the best.
We added a function (refine_clusters) to improves the initial clustering results, fixing common K-Means issues with: Oversensitivity to initial conditions (small spurious clusters) and poor handling of outliers.The refinement makes the final clusters more reliable for business analysis like product recommendations or category optimization.

6-Visualization
To display the result we used **t-SNE to** visualize high-dimensional clusters in 2D. to make sure that the model perform as well we implemented function (analyze_clusters_with_ignored_words) that Identify key terms defining each cluster while **Ignoring generic words** (e.g., "good", "product") via custom_ignored_words and **counts meaningful terms** per cluster (e.g., "speaker", "bluetooth", "charger”) the result is lists of top keywords characterizing each cluster.

Amazon Product Reviews Summarization Model

1. Project Overview

This project focuses on building an automated sentiment analysis model to classify Amazon product reviews into positive or negative categories. By analyzing thousands of real customer reviews, the goal was to develop a reliable machine learning model capable of understanding written feedback and determining its underlying sentiment.

Such systems are widely used in e-commerce, customer experience management, and marketing analytics to extract actionable insights from user opinions.

2. Dataset Description

The dataset used in this project was constructed by combining three publicly available sources containing Amazon customer reviews:
- 1429_1.csv
- Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products.csv
- Datafiniti_Amazon_Consumer_Reviews_of_Amazon_Products_May19.csv

Each dataset includes thousands of real customer reviews along with star ratings (ranging from 1 to 5). The relevant columns extracted were:
- reviews.text — the raw written review
- reviews.rating — the numeric rating given by the customer

To enable sentiment classification, the ratings were converted into binary labels:
- Positive (1) for ratings 4 and 5
- Negative (0) for ratings 3 and below

After merging the three datasets:
- Duplicate reviews were removed
- Null values were dropped
- Final dataset size after preprocessing: 46,876 reviews

This diverse and large dataset ensures a well-balanced and realistic training base for sentiment classification, simulating real-world customer feedback scenarios.

3. Methodology

To ensure clean and meaningful data, the text reviews were preprocessed using the following NLP steps:
- Lowercasing text
- Removing punctuation, numbers, and irrelevant short words
- Removing English stopwords
- Lemmatization to reduce words to their root form

After preprocessing, the model pipeline was built with:
- TF-IDF Vectorizer: to extract features from the cleaned text using unigrams and bigrams
- Random Forest Classifier: chosen for its robustness and ability to handle large feature sets

The dataset was split into training and testing sets (80/20 ratio), and class imbalance was handled using built-in class weighting.

4. Model Performance

The trained model achieved high accuracy and strong generalization on unseen data:
- Accuracy: 92.9%
- Precision: 94.2%
- Recall: 98.4%
- F1-Score: 96.0%
- ROC AUC Score: 0.88

These results confirm the model's ability to correctly classify both positive and negative reviews with minimal misclassification.

5. Confusion Matrix

The confusion matrix below illustrates the model's prediction accuracy on the test set:

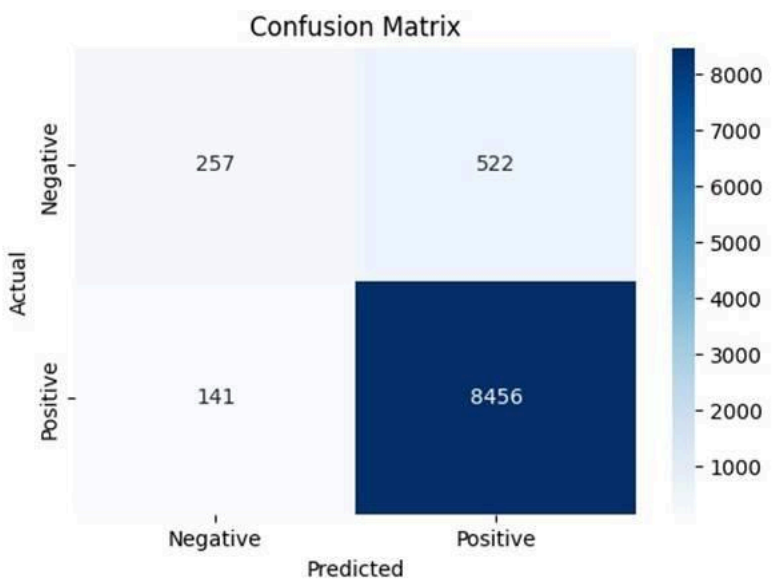


Figure : Confusion Matrix for Sentiment Classification

6.Sample Predictions

Review Text	Predicted Sentiment	Confidene

"Absolutely amazing! Worth every penny."	Positive	97.21%
"Terrible quality, would not recommend to anyone."	Negative	37.56%
"It's okay for the price but could be better."	Positive	63.70%
"The product arrived damaged and doesn't work as described."	Positive	57.82%
"Exceeded all my expectations! Perfect purchase."	Positive	99.64%

7. Conclusion

The sentiment classification model demonstrates exceptional performance and practical reliability. It is capable of supporting real-time customer feedback analysis and enhancing digital business intelligence systems. The approach taken in this project reflects a solid application of NLP and machine learning principles tailored for real-world product reviews.

Deployment

For deployment we've tried many ways to accomplish it. We tried to deploy using hugging face spaces and flask but we had problems and it didn't work. To demonstrate how the system might be used in a real-world setting, a lightweight **Gradio interface** was developed as a prototype frontend for interacting with the model. This interface simulates a complete customer review analysis pipeline, allowing users to input review text and receive:

- A **predicted sentiment** (Positive, Neutral, or Negative),
- A **predicted product category** (e.g., Ebook Readers, Batteries, Accessories), and
- A **generated summary** of the review's content.

Deployment Setup

The interface was built using the Gradio library, which offers a simple way to create web-based frontends for machine learning models. Although the current deployment uses mock data (randomized outputs) to simulate the behavior of the actual trained models, it successfully demonstrates the expected user flow and output structure of the final system.

Key Features:

- **User Input:** Multiline textbox for pasting or typing review text.
- **Outputs:**
    - **Sentiment Label:** Predicted emotional tone of the review.
    - **Product Category:** Suggested product type based on review content.
    - **Summary:** A concise, article-style summary generated from the input.
- **Title and UI Language:** The interface is titled **"الحاكم"** (The Judge), giving the tool a distinctive, user-friendly identity with multilingual support.

Conclusion

This project demonstrates the powerful potential of Natural Language Processing (NLP) in transforming unstructured customer reviews into structured, actionable insights. By automating the sentiment classification, clustering product categories, and generating summaries using modern NLP techniques and models like DistilRoBERTa and RoBERTa-large, we built an end-to-end pipeline that can support both consumers and businesses.

The sentiment classification models showed strong performance, particularly in detecting positive feedback, while clustering allowed us to simplify a wide range of product types into intuitive, user-friendly categories. Additionally, the summarization model distilled lengthy customer feedback into concise, readable summaries, helping users quickly understand the overall sentiment and value of a product.

We also developed a prototype deployment using Gradio, simulating how the system would behave in a production environment. Though the current version uses mock data, it lays the foundation for full model integration and real-time review analysis.

Key Takeaways:

- NLP enables scalable and efficient review analysis across large datasets.
- Sentiment analysis models can reach high accuracy with proper preprocessing and balanced data.
- Clustering helps reduce category noise and enhance product discoverability.
- Summarization provides quick, human-readable insights from customer feedback.
- Gradio-based UI can facilitate easy deployment and user interaction with NLP models.