BAHRAIN POLYTECHNIC بوليتكنك البحرين

Assessment Cover Sheet

Assessment	Project(Individual)		
Assessment	Uncontrolled	Individual	Not must-pass
Due Date	11 December 2025	**Course Code**	IT9002
Course Title	Natural Language Processing		
Internal Moderator's	Dr. Abdelhameed Fawzy		
External Examiner's	Dr.		

Instructions:

1. This cover sheet must be completed (section in blue below) and attached to your assessment before submission in hard copy/soft copy.
2. The time allowed for this assessment is XXX minutes/hours/days.
3. This assessment carries XXX marks distributed to a total of XXX questions assessing CILO X and CILO X.
4. The materials allowed for use in this assessment are XXX, XXX, and XXX.
5. The **use of generative AI tools is strictly prohibited.**
6. References consulted (if any) must be properly acknowledged and cited.
7. The assessment has a total of XXX pages.

Learner ID	**12010745**	**Date Submitted**	08/01/2026
Learner Name	**Israa Tageldin**		
Programme	MSc in AI		
Programme	IT9002		
Lecturer's	Sini Raj Pulari		

By submitting this assessment for marking, I affirm that this assessment is my own work.

Learner _____

Do not write beyond this line. For assessor use

Assessor's Name	Sini Raj Pulari		
Marking Date		**Maks Obtained**	

Comments:

Table of Contents

Stars vs. Sentiment: Comparing Text-Based and Rating-Based Sentiment in Amazon Reviews for Better Customer Insight

Introduction

Online reviews play an important role in shaping consumer purchasing decisions. They usually include both a **star rating** and a **review text**, and together they help buyers evaluate products and reduce uncertainty before making a purchase. Consumers may modify their purchasing preferences after reading product reviews, and this change can impact their level of trust in online e-commerce platforms. Therefore, the reviews information becomes important for both the customers and the companies that provide these services. [1], [2]

As a frequent online shopper myself, I always pay attention to both the star rating and the written review, whether the product is beauty, health, fashion, electronics, or home items. The combination of rating and text usually guides my decision and helps customers better understand the product's quality and performance.
But what happens if the star rating and the review text do not express the same message? This question forms the base of this project, which aims to examine sentiment–rating consistency in customers reviews. [1], [3].

This project investigates the **relationship between star ratings and review text sentiment** in Amazon product reviews. It compares **text-based sentiment** (derived from review content) with **rating-based sentiment** (learned from star labels) to measure agreement and disagreement patterns. Finally, it explores how combining both signals can provide **better customer insight** and support more reliable review interpretation for business use cases such as recommendations.

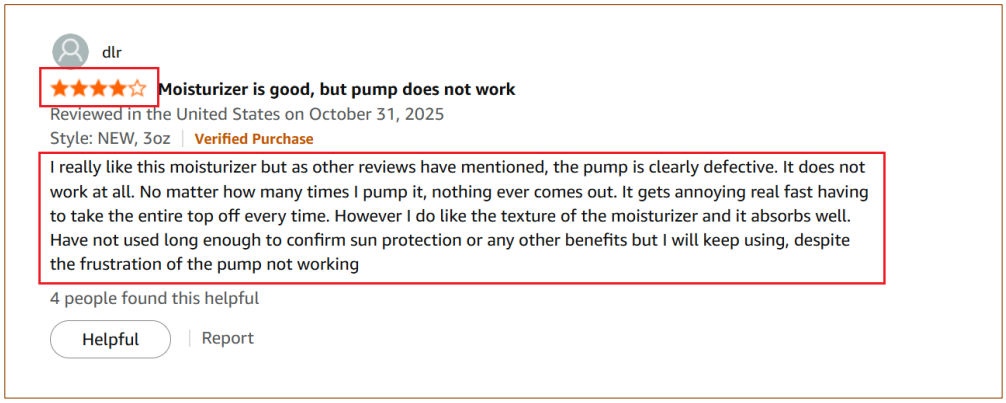

Figure 1. An example of an Amazon product review showing the review text and its star rating.

Task 1 – Problem Statement Formulation and Definition

Motivation

Online retailers depend heavily on customer reviews to shape purchase decisions, pricing strategies, and product visibility. A typical review contains two key signals: a **text** expressing opinion and a **1–5 star rating**. However, studies show that these two components do not always convey the same sentiment, and this issue has been discussed under concepts such as text-rating discrepancy and text-score disagreement. [1], [2]

For example, rating–text inconsistency could occur in several situations:

- Another customer may give **high stars with negative text** due to personal bias, external influence, or attempts to benefit from return or refund policies.
- A customer may leave **low stars with positive text** due to shipping delays, unmet expectations, pricing concerns, promotion tricks, or other factors not directly related to product quality.

Figure 2. Example of a real Amazon product review showing low stars due to shipping [4]

Such sentiment–rating mismatches can reduce the reliability of product evaluations and mislead both customers and businesses. Prior research also highlights that mismatch detection is useful because inconsistent reviews can distort overall ratings and affect user decisions. [3], [5]

Problem Statement

Star ratings are often treated as a direct summary of customer opinion, but they may not match the sentiment expressed in review text. This project examines **alignment and disagreement** between **text-based sentiment** from written reviews and **rating-based sentiment** derived from star ratings in Amazon reviews.

By analyzing areas of agreement and disagreement, the study aims to generate better customer insight and propose a combined view of rating and text signals that can support more reliable review interpretation and recommendation decisions.

Objectives and Expected Result

1. Classify Amazon review text into three sentiment classes (**positive, neutral, negative**) using NLP techniques.
2. Convert star ratings into three sentiment groups (**1–2 = negative, 3 = neutral, 4–5 = positive**) to allow direct comparison.
3. Train machine learning models using rating-based labels and compare their predictions with text-based sentiment results.
4. Compare **text-based sentiment** with **rating-based sentiment** to measure how often they align or differ.
5. Present visual and statistical results showing **agreement rates**, **disagreement patterns**, and examples of mismatched cases **if any**.
6. Discuss whether combining rating and text sentiment could give better customer insight and support more reliable recommendations.

Expected outcome:
This study will examine whether mismatches between star ratings and review text sentiment occur in Amazon reviews, and how often they happen. It will also explore whether combining both the star rating and the text sentiment can provide better customer insight and support more reliable recommendations.

Task 2 - Selection of an Appropriate Data Set (Data Collection)

Dataset Selection

For this project, the chosen business domain is **Amazon E-commerce reviews**, focusing on **Health & Personal Care** products. This category contains rich, emotionally expressive text, making it suitable for text-based sentiment analysis

Since this project compares sentiment from review text with sentiment derived from star ratings, this dataset is suitable for checking whether the two signals agree or differ, while also providing rich text for effective NLP modelling.

Dataset Source & Justification

The dataset was obtained from large-scale Amazon Reviews dataset, collected in 2023 that includes verified customer purchase reviews [6].
The original dataset contained **494,121** Amazon Health & Personal Care product reviews, spanning from *February 2001* to *September 2023*.

To keep the analysis recent and reduce processing load in Google Colab, only reviews from the year **2023** were retained, resulting in a final working dataset of **11,477 reviews**.

Reasons for choosing this dataset:

- Contains real customer opinion review text essential for NLP sentiment analysis.
- Includes both ratings and review text, allowing rating sentiment comparison.
- Focuses on personal care products, where emotional opinions lead to rich polarity.
- Includes verified purchase and helpful votes metadata for richer insights.
- Suitable for classification, prediction, and comparison research.

Dataset Structure, Shape and Overview

The selected dataset was first uploaded to **my GitHub repository**, then loaded into Google Colab using *Pandas* for initial exploration and quality checks. It contains **11,477 reviews** with **10 features**, including numeric ratings, review text, user identifiers, timestamps, and purchase metadata. Figures 3,4 and 5 shows the shape and overview of the dataset.

```
import pandas as pd
#Load dataset from my GitHub repository into datafram
file_path = 'https://raw.githubusercontent.com/israa-tageldin/NLP-Project/refs/heads/main/Amazon_Reviews_Health.jsonl'
df = pd.read_json(file_path, lines=True)

print(f"Successfully loaded {len(df)} reviews.")
df.head()
```

```
Successfully loaded 11477 reviews.
```

	rating	title	text	images	asin	parent_asin	user_id	timestamp	helpful_vote	verified_purchase
0	5	Works	I use these for my car windows when they fog u...	[]	B09JYKTSKR	B09T5H2VKP	AHHFW36BP4VMQWC6V2NTKIXFAA2A	2023-01-17 17:28:21.955	0	True
1	3	One of my go to brushes	This double sided brush is decent for dogs- my...	[{'small_image_url': 'https://m.media-amazon.c...	B08B7238H5	B0963Y93GG	AFV22L7AEKI2LW6HMLRLUKNYVBGQ	2023-02-15 18:59:45.339	0	True
2	3	It's only soluble fiber	I bought this hoping to increase my fiber inta...	[]	B07577L1H8	B07577L1H8	AFTIM4OKTMFNKK4BRL4GNDHMPFSA	2023-02-17 20:14:54.147	0	True
3	5	As expected	As advertised	[]	B007QESMDK	B007QESMDK	AE25NQAZI3725GZIL5FS52ZIKWKQ	2023-01-17 03:19:50.103	0	True
4	5	I love this mask! Very well crafted!	I have been wanting to buy a Henry mask for a ...	[]	B08TY6PZJN	B08ZCLW41P	AFKFGAKYW7CSKWQNZE4CIPW66TJQ	2023-01-26 20:33:24.104	0	True

Figure 3. Dataset loaded into pandas with a preview of the first few records.

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 11477 entries, 0 to 11476
Data columns (total 10 columns):
 #   Column             Non-Null Count  Dtype
---  ------             --------------  -----
 0   rating             11477 non-null  int64
 1   title              11477 non-null  object
 2   text               11477 non-null  object
 3   images             11477 non-null  object
 4   asin               11477 non-null  object
 5   parent_asin        11477 non-null  object
 6   user_id            11477 non-null  object
 7   timestamp          11477 non-null  datetime64[ns]
 8   helpful_vote       11477 non-null  int64
 9   verified_purchase  11477 non-null  bool
dtypes: bool(1), datetime64[ns](1), int64(2), object(6)
memory usage: 818.3+ KB
```

Figure 4. Summary information of the DataFrame

```
df.describe()
```

	rating	timestamp	helpful_vote
count	11477.000000	11477	11477.000000
mean	3.847434	2023-03-13 13:05:16.322029312	0.440795
min	1.000000	2022-12-31 21:13:10.032000	0.000000
25%	3.000000	2023-01-30 16:23:39.623000064	0.000000
50%	5.000000	2023-03-03 23:46:42.276000	0.000000
75%	5.000000	2023-04-07 21:03:54.273999872	0.000000
max	5.000000	2023-09-11 05:03:12.467000	87.000000
std	1.584163	NaN	1.995006

Figure 5. Descriptive statistics of the dataset.

Column Name	Description	Type
rating	User provided star rating (1–5)	int64
title	Review short headline	object
text	Full review text content	object
images	Image URLs if user uploaded any	object
asin	Product identifier	object
parent_asin	Parent product ID	object
user_id	Reviewer ID	object
timestamp	Date/time of review	datetime64
helpful_vote	Number of helpful vote counts	int64
verified_purchase	Whether the user is a verified buyer	bool

Table 1. Dataset columns with descriptions and corresponding data types.

Labels in the Dataset

The dataset includes a numeric label, **rating**, with five classes (1–5 stars). As shown in Figure 6, the ratings are imbalanced, with 5-star reviews forming the largest portion of the dataset. In this project, star ratings will be first converted into **three sentiment groups** (1–2 = Negative, 3 = Neutral, 4–5 = Positive) and used to train **rating-based sentiment models**. In parallel, **text-based sentiment labels** are generated from the review text. These two signals are then compared to measure alignment and disagreement and to explore whether combining rating and text sentiment can provide better customer insight.

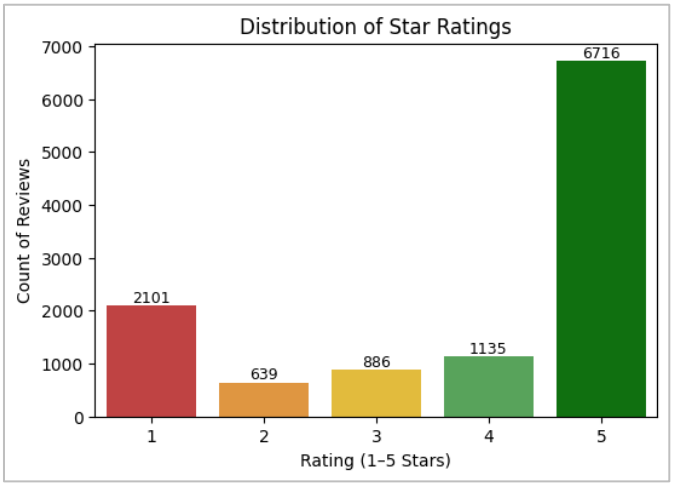

Figure 6. Distribution of star ratings (1–5) across the reviews.

Task 3 - Text Preprocessing

Text preprocessing is a critical stage in NLP because raw review data often contains noise and inconsistencies that can reduce model accuracy. Reviews may include duplicated entries, missing values, HTML tags, and mixed casing, which can distort word patterns and affect sentiment analysis. To prepare the data for both **text-based sentiment extraction** and **rating-based model training**, a structured preprocessing pipeline was applied. This pipeline cleans the review text, removes unnecessary information, performs tokenization and lemmatization to normalize word forms, and finally converts the processed reviews into TF-IDF features for modelling and comparison tasks.

Initial Data Cleaning and Preparation

Before applying NLP transformations, the dataset was cleaned and prepared as follows:

- **Removed unused columns:** Only fields required for analysis were retained.
- **Removed duplicate records:** Duplicate reviews were eliminated.
- **Removed empty reviews:** Rows with missing or empty review content were filtered out to ensure all remaining records contain valid text.
- **Lowercased text:** All review text was converted to lowercase.
- **Removed stopwords:** Common non-informative words (e.g., *the*, *is*, *and*) were removed to reduce noise and enhance the quality of the textual features.
- **Combined title and review text:** The title and text fields were merged into one review field to capture richer context and improve sentiment signals.

These steps produced a clean and consistent text column (*reviews*) ready for NLP processing.

```
Initial shape: (11477, 10)
After keeping useful columns: (11477, 6)
After duplicates removed: (11238, 6)
Final shape after cleaning: (11233, 7)
```

	rating	title	text	timestamp	helpful_vote	verified_purchase	reviews
0	5	Works	I use these for my car windows when they fog u...	2023-01-17 17:28:21.955	0	True	works i use these for my car windows when they...
1	3	One of my go to brushes	This double sided brush is decent for dogs-my...	2023-02-15 18:59:45.339	0	True	one of my go to brushes this double sided brus...
2	3	It's only soluble fiber	I bought this hoping to increase my fiber inta...	2023-02-17 20:14:54.147	0	True	it s only soluble fiber i bought this hoping t...
3	5	As expected	As advertised	2023-01-17 03:19:50.103	0	True	as expected as advertised
4	5	I love this mask! Very well crafted!	I have been wanting to buy a Henry mask for a ...	2023-01-26 20:33:24.104	0	True	i love this mask very well crafted i have been...

Figure 7. Dataset shape before and after data cleaning, with a preview of the first few cleaned records.

Text Representation Techniques

After the initial cleaning stage, the review text was further processed to create a structured representation suitable for both text-based sentiment extraction and rating-based model training. Tokenization, Lemmatization, and TF-IDF feature extraction steps were applied to normalized text to transform it into a structured and standardized format ready for NLP modelling and machine learning classifiers.

1. Tokenization

Tokenization was used to split each cleaned review into individual word tokens. This step transforms unstructured text into a structured sequence of words, enabling consistent word-level processing and preparing the reviews for normalization and feature extraction.

```
df["tokens"] = df["reviews"].apply(word_tokenize)

# Show tokenization sample output
display(df[["reviews", "tokens"]].head(3))
```

	reviews	tokens
0	works use car windows fog work glasses well	[works, use, car, windows, fog, work, glasses,...
1	one go brushes double sided brush decent dogs ...	[one, go, brushes, double, sided, brush, decen...
2	soluble fiber bought hoping increase fiber int...	[soluble, fiber, bought, hoping, increase, fib...

Figure 8. Tokenization process applied to the review text column.

2. Lemmatization

Lemmatization was applied to convert each token into its base dictionary form. This reduces word form variation across the dataset and improves feature consistency by treating different forms of the same word as the same term, which supports more stable sentiment learning.

```
lemmatizer = WordNetLemmatizer()
def lemmatize_tokens(tokens):
    return [lemmatizer.lemmatize(t) for t in tokens]

df["tokens_lemma"] = df["tokens"].apply(lemmatize_tokens)
# Convert back to text (needed for TF-IDF input)
df["review_lemmatized"] = df["tokens_lemma"].apply(lambda x: " ".join(x))
# Show lemmatization sample output
display(df[["reviews", "review_lemmatized"]].head(3))
```

	reviews	review_lemmatized
0	works use car windows fog work glasses well	work use car window fog work glass well
1	one go brushes double sided brush decent dogs ...	one go brush double sided brush decent dog iss...
2	soluble fiber bought hoping increase fiber int...	soluble fiber bought hoping increase fiber int...

Figure 9. Lemmatization process applied to the tokenized review text.

3. TF-IDF Feature Extraction

TF-IDF was used to transform the lemmatized review text into numerical feature vectors suitable for training **rating-based sentiment models**. It assigns higher weights to informative terms that are frequent within a review but less common across the dataset, helping highlight sentiment bearing words and phrases. Figure 10 shows the construction of the TF-IDF feature matrix, including matrix dimensions and sample vocabulary terms learned from the review corpus.

```
[13]
✓ 4s
    tfidf = TfidfVectorizer(max_features=5000, ngram_range=(1, 2), stop_words="english")
    X_tfidf = tfidf.fit_transform(df["review_lemmatized"])

    print("TF-IDF matrix shape:", X_tfidf.shape)

    feature_names = tfidf.get_feature_names_out()
    print("Sample TF-IDF features:", feature_names[:100])

    TF-IDF matrix shape: (11207, 5000)
    Sample TF-IDF features: ['aa' 'ab' 'ability' 'able' 'able use' 'abrasive' 'absolute' 'absolutely'
     'absolutely amazing' 'absolutely love' 'absolutely loved' 'absorb'
     'absorbed' 'absorbent' 'absorbs' 'absorption' 'ac' 'accept' 'access'
     'accessory' 'accident' 'accidentally' 'according' 'accurate' 'ache'
     'ache pain' 'achieve' 'aching' 'acid' 'acid reflux' 'acne' 'act' 'acting'
     'action' 'active' 'activity' 'actual' 'actually' 'actually pretty'
     'actually work' 'acv' 'ad' 'adapter' 'add' 'added' 'adding' 'addition'
     'additional' 'additionally' 'additive' 'address' 'adequate' 'adhere'
     'adhered' 'adhesion' 'adhesive' 'adjust' 'adjustable' 'adjusted'
     'adjustment' 'admit' 'adorable' 'adult' 'advanced' 'advertised'
     'advertisement' 'advertising' 'advise' 'advised' 'ae' 'af' 'affect'
     'afford' 'affordable' 'afraid' 'afternoon' 'aftertaste' 'age'
     'aggressive' 'ago' 'agree' 'agrestis' 'ahead' 'aid' 'aid kit' 'air'
     'air freshener' 'air purifier' 'air quality' 'al' 'alarm' 'alcohol'
     'alert' 'alkaline' 'allergic' 'allergy' 'alleviate' 'allow' 'allowed'
     'allowing']
```

Figure 10. TF-IDF feature matrix shape and sample learned vocabulary.

TF-IDF Terms Plot

A plot of the top TF-IDF terms was generated [Figure 11] to visualize the most influential words and phrases captured by the vectorizer based on overall TF-IDF weight. This visualization provides an interpretable overview of important terms before modeling and supports understanding of how review text is represented numerically.

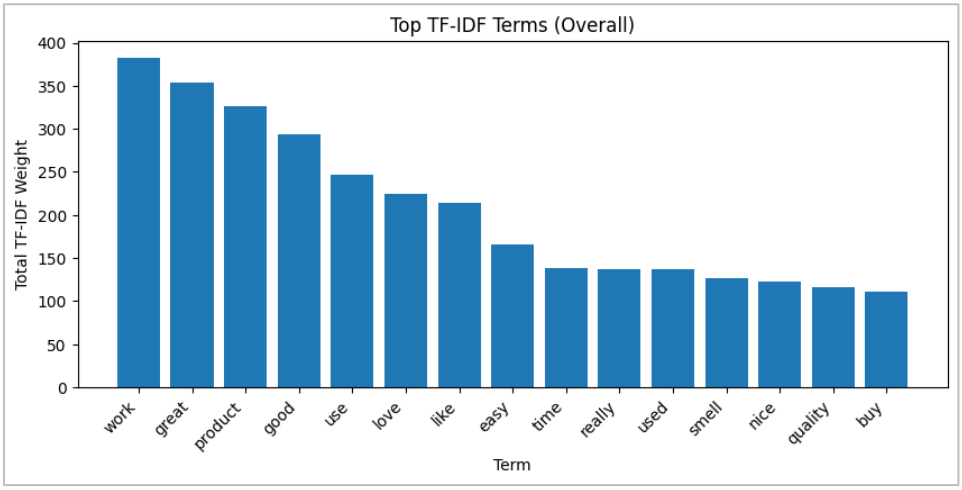

Figure 11. Top TF-IDF weighted terms in the review corpus.

Task 4 – Text Representation

After cleaning and preprocessing the Amazon Health & Personal Care reviews dataset, the next step is to represent the review text in a machine-readable format so it can be analysed and used for sentiment classification.

Different text representation techniques capture different aspects of language. For this project, three representation methods were selected because they are highly relevant to sentiment detection and support the comparison between rating-based and text-based sentiment signals: **Bag of Words, N-grams**, and **Part-of-Speech (POS) tagging**. Together, these techniques capture word frequency patterns, short sentiment phrases such as negations and intensifiers, and linguistic structures that often convey opinion strength and contrast.

1. Bag of Words (BoW)

1. Bag of Words (BoW) + Top Words Plot

```
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
import matplotlib.pyplot as plt

bow = CountVectorizer(max_features=5000, stop_words="english")
X_bow = bow.fit_transform(df["review_lemmatized"])

print("BoW matrix shape:", X_bow.shape)

terms = np.array(bow.get_feature_names_out())
counts = np.asarray(X_bow.sum(axis=0)).ravel()

top_idx = counts.argsort()[-15:][::-1]
top_terms = terms[top_idx]
top_counts = counts[top_idx]

plt.figure(figsize=(10,4))
plt.bar(top_terms, top_counts)
plt.title("Top 15 Most Frequent Words (Bag of Words)")
plt.xlabel("Word")
plt.ylabel("Count")
plt.xticks(rotation=45, ha="right")
plt.show()
```

```
BoW matrix shape: (11207, 5000)
```

Bag of Words represents each review as a vector of word counts, where each feature corresponds to a word in the vocabulary. BoW is useful for customer reviews because sentiment is frequently expressed through specific words such as “good”, “bad”, “effective”, “waste”, and “recommend”. This representation provides a strong baseline for sentiment classification and allows simple interpretation of which words appear most frequently across reviews. In this project, BoW is used to capture general word usage patterns and support later modelling of sentiment from review text.

Figure 12. Construction of the Bag-of-Words matrix

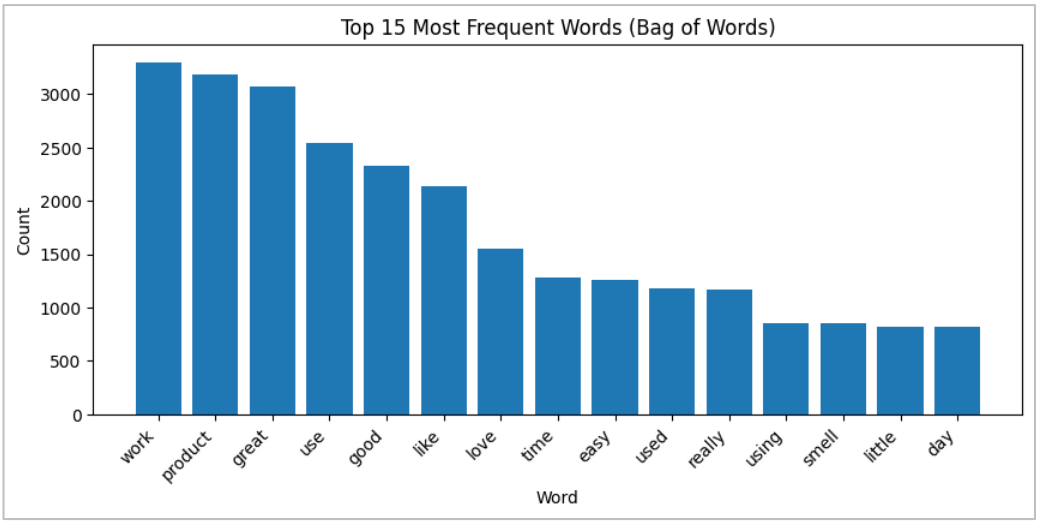

Figure 13. Visualization of the top 15 most frequent words by Bag of Words matrix

2. N-grams (Bigrams)

2. N-grams (Bigrams) + Top Bigrams Plot

```
from sklearn.feature_extraction.text import CountVectorizer
import numpy as np
import matplotlib.pyplot as plt

bigram_vec = CountVectorizer(stop_words="english", ngram_range=(2,2), max_features=5000)
X_bi = bigram_vec.fit_transform(df["review_lemmatized"])

print("Bigram matrix shape:", X_bi.shape)

bigrams = np.array(bigram_vec.get_feature_names_out())
bi_counts = np.asarray(X_bi.sum(axis=0)).ravel()

top_idx = bi_counts.argsort()[-15:][::-1]
top_bigrams = bigrams[top_idx]
top_bi_counts = bi_counts[top_idx]

plt.figure(figsize=(10,4))
plt.bar(top_bigrams, top_bi_counts)
plt.title("Top 15 Most Frequent Bigrams (N-grams)")
plt.xlabel("Bigram")
plt.ylabel("Count")
plt.xticks(rotation=45, ha="right")
plt.show()
```

```
Bigram matrix shape: (11207, 5000)
```

Figure 14. Bigram feature extraction using an N-gram.

N-grams extend BoW by capturing sequences of words rather than single terms. This is especially important for sentiment analysis because many sentiment expressions depend on short phrases, not isolated words. For example, “not good” conveys a negative meaning even though “good” alone is positive. Bigrams also capture common review phrases such as “works great”, “highly recommend”, or “caused rash”, which are strongly related to sentiment in Health & Personal Care reviews. In this project, bigrams are used to better model sentiment-bearing phrases and improve detection of rating–text misalignment.

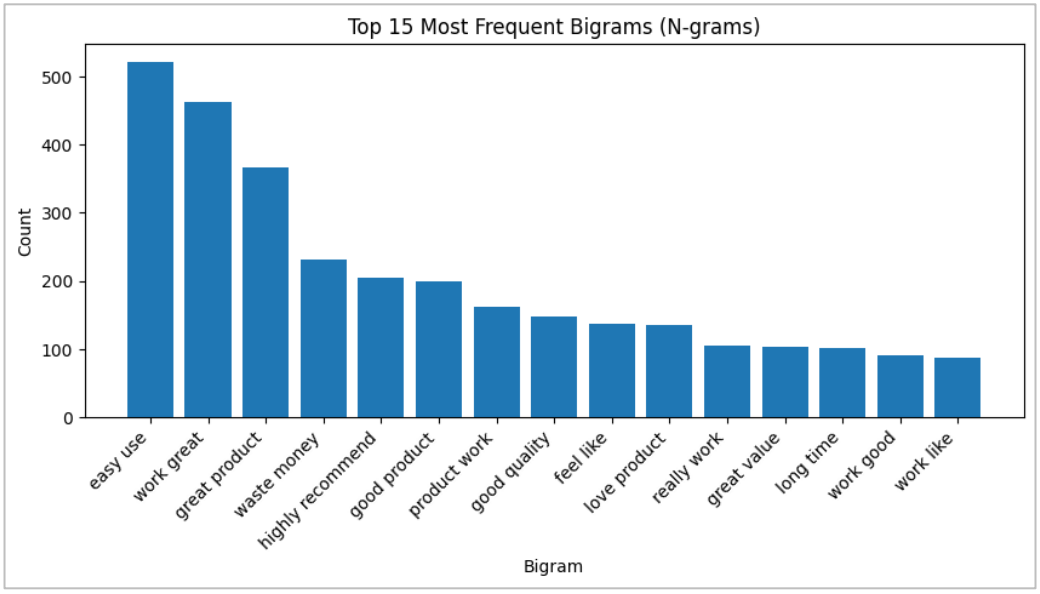

Figure 15. visualization of the top 15 most frequent bigrams

3. POS Tagging (Part-of-Speech Tagging)

Part-of-Speech tagging assigns grammatical labels to each token (e.g., noun, verb, adjective, adverb). This was used in this project to identify the grammatical structure of each review. This is valuable in sentiment analysis because sentiment is often carried by **adjectives and adverbs**, such as “amazing”, “terrible”, “very”, and “extremely”. POS features also help identify contrastive patterns such as *“works well but expensive”*, which often contribute to sentiment ambiguity.

3. POS Tagging + Sample + POS Distribution Plot

```
from nltk import pos_tag
from collections import Counter
import matplotlib.pyplot as plt

# Example POS tagging for one review
sample_tokens = df["tokens_lemma"].iloc[0]
print("Sample tokens:", sample_tokens[:25])
print("POS tags:", pos_tag(sample_tokens[:25]))

subset_tokens = df["tokens_lemma"].head(500).tolist()
all_tags = []
for toks in subset_tokens:
    all_tags.extend([tag for _, tag in pos_tag(toks)])

tag_counts = Counter(all_tags)
top_tags = tag_counts.most_common(12)

labels = [x[0] for x in top_tags]
values = [x[1] for x in top_tags]

plt.figure(figsize=(10,4))
plt.bar(labels, values)
plt.title("Top POS Tags (Sample of 500 Reviews)")
plt.xlabel("POS Tag")
plt.ylabel("Count")
plt.show()
```

```
Sample tokens: ['work', 'use', 'car', 'window', 'fog', 'work', 'glass', 'well']
POS tags: [('work', 'NN'), ('use', 'NN'), ('car', 'NN'), ('window', 'NN'), ('fog', 'NN'), ('work', 'NN'), ('glass', 'NN'), ('well', 'RB')]
```

Figure 16. Part-of-speech tagging construction and example.

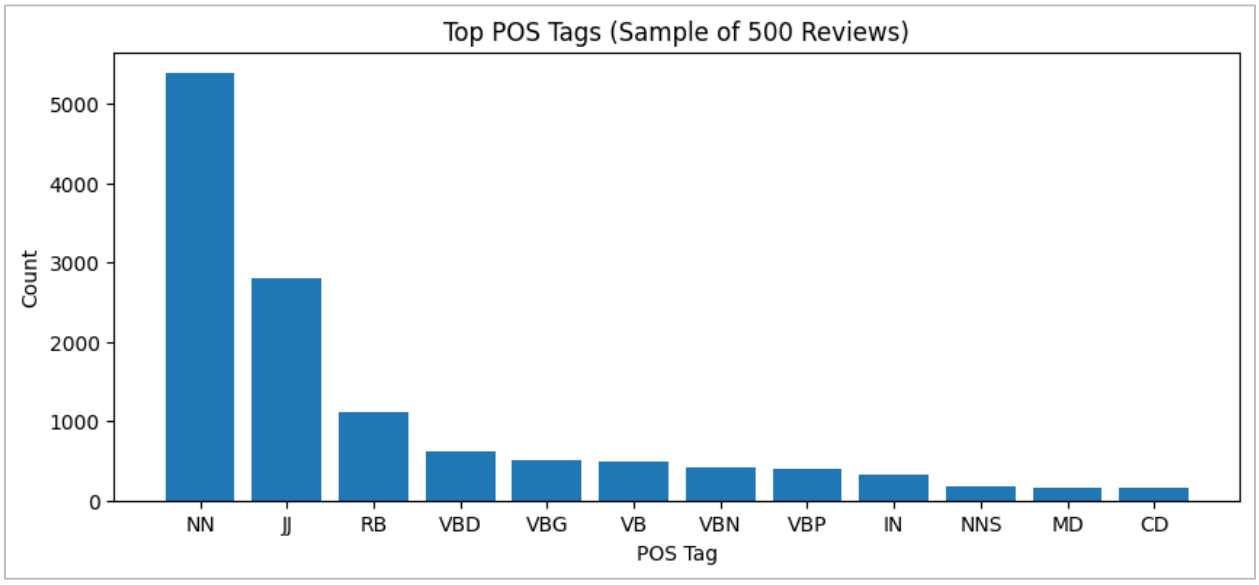

Figure 17. Distribution of the most frequent POS tags in the review corpus

Task 5 - Sentiment Modelling and Prediction

After preparing the dataset and generating machine-readable representations (Task 4), the next step was to build sentiment models from two different perspectives:

1. **Text-based sentiment model (BERT)** - independent transformer model was applied to the cleaned review text to generate a sentiment signal based purely on what customers wrote.
2. **Rating-based sentiment models** - supervised ML trained using polarity labels derived from star ratings to represent sentiment implied by the numeric ratings.

This design supports the project goal of comparing alignment and disagreement between star-rating sentiment and text-derived sentiment and later constructing a combined reliability signal.

1. Text-Based Sentiment using BERT

To generate an independent sentiment signal from the review text, a pre-trained **BERT-family transformer** sentiment model was applied directly to the cleaned reviews text to predict sentiment classes (**negative, neutral, positive**).
This text-based model predicts sentiment without using star ratings, allowing a fair comparison between what customers wrote and what they rated.
The BERT outputs were stored as:

- *bert_sentiment_3* (predicted sentiment class)
- *bert_score* (confidence score)

Figure 18. BERT-based sentiment inference on review text with sample predicted labels and confidence scores.

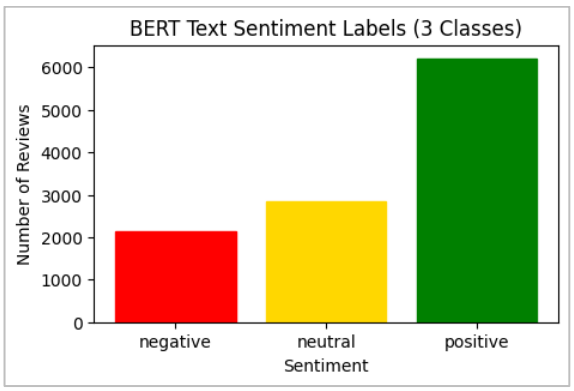

Figure 19. Distribution of BERT-derived sentiment labels across negative, neutral, and positive classes.

2. Rating-Based Sentiment using ML Classifiers

To model sentiment derived from star ratings, ratings were mapped into three polarity groups: **1–2 = Negative**, **3 = Neutral**, and **4–5 = Positive**. These rating-derived labels were used as target classes to train supervised machine learning models on textual features extracted from the cleaned reviews column.

Create Rating-based labels (3-class)

```
import numpy as np
RATING_COL = "rating" if "rating" in df_cleaned.columns else "star_rating"
def map_rating_to_polarity(r):
    if r in [1, 2]:
        return "negative"
    elif r == 3:
        return "neutral"
    elif r in [4, 5]:
        return "positive"
    return np.nan

df_cleaned["rating_polarity_3"] = df_cleaned[RATING_COL].apply(map_rating_to_polarity)
df_cleaned = df_cleaned.dropna(subset=["rating_polarity_3"]).copy()
df_cleaned["rating_polarity_3"].value_counts()
```

rating_polarity_3	count
positive	7649
negative	2686
neutral	872

dtype: int64

Figure 20. Mapping of star ratings into three sentiment polarity groups (1–2: Negative, 3: Neutral, 4–5: Positive)

The dataset was divided into training and testing subsets using an 80:20 stratified split to preserve class distribution. Textual features were generated using a TF-IDF vectorizer with unigram and bigram representations, capturing both individual sentiment terms and short polarity-bearing phrases such as negations. The resulting feature space consisted of 23,432 terms, reflecting the diversity of sentiment expressions in customer reviews.

Three complementary classifiers were trained:

2.1 Logistic Regression (LR)

Logistic Regression was employed as a strong linear baseline for multi-class text classification. It is well suited to high-dimensional sparse TF-IDF features and provides stable probabilistic predictions. Class weighting was applied to mitigate potential class imbalance.

2.2 Support Vector Machine (SVM)

A linear Support Vector Machine classifier was trained to learn a maximum-margin decision boundary between sentiment classes. SVM is particularly effective for text classification tasks involving large sparse feature spaces and was included to provide a robust discriminative model.

2.3 Multinomial Naïve Bayes (MNB)

Multinomial Naïve Bayes was used as a probabilistic benchmark model specifically designed for word frequency data. Despite its simplifying independence assumptions, MNB often performs competitively in sentiment classification and serves as an efficient baseline for comparison.

```python
from sklearn.model_selection import train_test_split
from sklearn.feature_extraction.text import TfidfVectorizer

X = df_cleaned["reviews"].astype(str)
y = df_cleaned["rating_polarity_3"].astype(str)

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

tfidf = TfidfVectorizer(
    lowercase=True,
    ngram_range=(1, 2),
    min_df=2,
    max_df=0.95
)

X_train_tfidf = tfidf.fit_transform(X_train)
X_test_tfidf  = tfidf.transform(X_test)

X_train_tfidf.shape, X_test_tfidf.shape
```

```
((8965, 23432), (2242, 23432))
```

Figure 21. Train–test split of review text

```python
from sklearn.linear_model import LogisticRegression
from sklearn.svm import LinearSVC
from sklearn.naive_bayes import MultinomialNB

lr = LogisticRegression(max_iter=2000, class_weight="balanced")
svm = LinearSVC(class_weight="balanced")
mnb = MultinomialNB()

lr.fit(X_train_tfidf, y_train)
svm.fit(X_train_tfidf, y_train)
mnb.fit(X_train_tfidf, y_train)

pred_lr  = lr.predict(X_test_tfidf)
pred_svm = svm.predict(X_test_tfidf)
pred_mnb = mnb.predict(X_test_tfidf)
```

Figure 22. Training and prediction using Logistic Regression, SVM, and Multinomial Naive Bayes classifiers.

These models represent rating-based sentiment because their training labels originate from star ratings rather than human sentiment annotation.

Models Evaluation and Outputs

The performance of the rating-based classifiers is evaluated using standard classification metrics including precision, recall, and F1-score (Figure 23). These outputs enable Task 6 (agreement and disagreement analysis) and the final combined reliability indicator.

This evaluation measures how well each model learns sentiment patterns derived from star ratings and highlights differences in their predictive behaviour.

Predictions from all models were stored in the dataset as new columns to support later analysis (Figure 24):

- *pred_lr, pred_svm, pred_mnb* (rating-based predictions)
- *bert_sentiment_3* (text-based prediction)

```python
from sklearn.metrics import accuracy_score, precision_recall_fscore_support
import pandas as pd

models = {
    "Logistic Regression": pred_lr,
    "Support Vector Machine": pred_svm,
    "Naive Bayes": pred_mnb
}

rows = []

for name, preds in models.items():
    acc = accuracy_score(y_test, preds)
    precision, recall, f1, _ = precision_recall_fscore_support(
        y_test, preds, average="weighted"
    )

    rows.append({
        "Model": name,
        "Accuracy": round(acc, 6),
        "Precision": round(precision, 6),
        "Recall": round(recall, 6),
        "F1 Score": round(f1, 6)
    })

eval_df = pd.DataFrame(rows).sort_values(by="Accuracy", ascending=False)
eval_df
```

Figure 23. Computation of accuracy, precision, recall, and F1-score for evaluating sentiment classification models

```python
X_all_tfidf = tfidf.transform(df_cleaned["reviews"].astype(str))

df_cleaned["pred_lr"]  = lr.predict(X_all_tfidf)
df_cleaned["pred_svm"] = svm.predict(X_all_tfidf)
df_cleaned["pred_mnb"] = mnb.predict(X_all_tfidf)

df_cleaned[["rating_polarity_3","bert_sentiment_3","pred_lr","pred_svm","pred_mnb"]].head()
```

	rating_polarity_3	bert_sentiment_3	pred_lr	pred_svm	pred_mnb
0	positive	neutral	positive	positive	positive
1	neutral	neutral	neutral	neutral	positive
2	neutral	neutral	neutral	neutral	positive
3	positive	neutral	positive	positive	positive
4	positive	positive	positive	positive	positive

Figure 24. New columns creation out of rating-derived labels, BERT sentiment predictions, and classical ML model outputs.

Task 6 – Evaluation, Inferences, Recommendations and Reflection

Evaluation of Sentiment Models

Three rating-based machine learning models—Logistic Regression, Support Vector Machine, and Multinomial Naïve Bayes—were evaluated using standard classification metrics including accuracy, precision, recall, and F1-score (Figure 25). The Support Vector Machine achieved the highest performance with an accuracy of **83.45%** and an F1-score of **0.83**, followed closely by Logistic Regression, while Multinomial Naïve Bayes produced comparatively lower results. This confirms that linear discriminative models are more effective than probabilistic baselines for high-dimensional TF-IDF sentiment features.

In addition, a transformer-based BERT model was applied to generate independent text-based sentiment predictions. Rather than treating star ratings as perfect ground truth, BERT outputs were compared with rating-derived labels and the predictions of the three machine learning models to assess alignment and disagreement between sentiment signals. Agreement analysis revealed that the rating-based models were highly consistent, with **73.26% (8,210 reviews)** showing complete agreement across all three classifiers. BERT predictions matched the rating-derived polarity labels in **67.29% (7,541 reviews)** of the dataset, and in **56.23% (6,302 reviews)** of cases BERT agreed simultaneously with all rating-based models, representing the strongest form of sentiment consensus (Figure 26).

	Model	Accuracy	Precision	Recall	F1 Score
1	Support Vector Machine	0.834523	0.821078	0.834523	0.826875
0	Logistic Regression	0.803747	0.828224	0.803747	0.814201
2	Naive Bayes	0.755129	0.735685	0.755129	0.691081

Figure 25. Evaluation metrics of the trained sentiment classification models.

	Metric	Count	Percent
0	ML models all agree (LR=SVM=MNB)	8210	73.26
1	BERT agrees with rating label	7541	67.29
2	BERT agrees with all ML models	6302	56.23

Figure 26. Agreement levels between BERT sentiment labels, rating-derived labels, and classical ML models.

Agreement Groups and Reliability Levels

Based on the comparison between text-based and rating-based sentiment, four agreement groups were defined Figure 27.
These groups represent different patterns of alignment and disagreement between BERT sentiment and rating-based machine learning models, enabling structured analysis of sentiment consistency

Using these groups, a sentiment reliability indicator was constructed. The resulting distribution is shown below:

agreement_group	count
All agree (BERT + ML)	6302
ML agree, BERT differs	1908
Mixed/low agreement	1735
ML disagree, BERT matches rating	1262

Figure 27. Agreement groups across BERT, classical ML models, and rating-derived labels.

reliability_level	proportion
Very High	0.562
High	0.170
Low	0.155
Medium	0.113

Figure 28. Proportion of review reliability levels based on sentiment agreement analysis.

These results indicate that more than half of the reviews contain highly reliable sentiment signals, while approximately one quarter of the dataset falls into medium or low reliability categories, where interpretation of customer sentiment is more uncertain.

Inferences

The agreement analysis demonstrates that star-rating–based sentiment models are highly consistent, with the three machine learning classifiers producing identical predictions in **73.26%** of the reviews (Figure 26). This indicates that numeric ratings provide a stable but limited representation of customer sentiment.

The text-based BERT model matched the rating-derived polarity labels in **67.29%** of cases, suggesting that approximately one-third of reviews contain sentiment cues that are not fully reflected by star ratings. Furthermore, only **56.23%** of all reviews (Figure 26) exhibited complete agreement between BERT and all rating-based classifiers, highlighting that strong sentiment consensus is present in just over half of the dataset.

The constructed reliability indicator further revealed that **56.2%** of reviews were classified as *Very High* reliability, while **17.0%** were *High*. However, a notable proportion of reviews fell into *Medium* (**11.3%**) and *Low* (**15.5%**) reliability categories (Figure 28). These cases typically involve mixed or implicit sentiment, contrastive language, or vague expressions that numeric ratings fail to represent accurately.

Overall, these findings confirm that although star ratings are useful for capturing general sentiment trends, they do not consistently reflect the nuanced opinions expressed in review text. Integrating text-based sentiment analysis with rating-based models therefore provides a more trustworthy interpretation of customer feedback.

Recommendations

Based on the findings, it is recommended that customer feedback analysis systems should not rely exclusively on star ratings or text sentiment in isolation. Instead, both signals should be integrated using a reliability-aware framework. Reviews classified as *Very High* and *High* reliability can be confidently used for automated decision-making, while *Medium* and *Low* reliability cases should be flagged for manual inspection or additional contextual analysis. Figure 29 shows a proposed Framework process flow (image by Gen-AI).

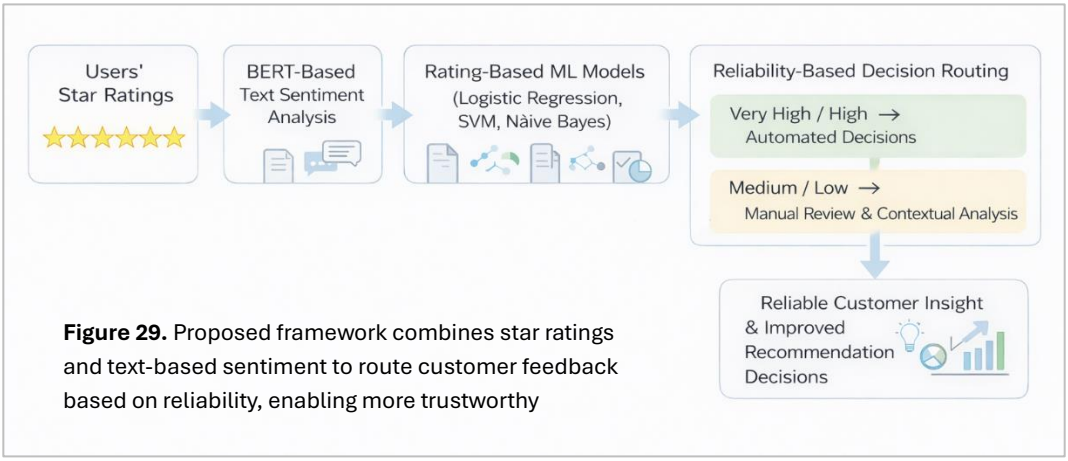

Figure 29. Proposed framework combines star ratings and text-based sentiment to route customer feedback based on reliability, enabling more trustworthy

This proposed reliability-aware recommendation framework integrates star ratings with BERT-based text sentiment and rating-trained machine learning models to route customer feedback into automated or manual decision pathways based on sentiment confidence, thereby improving the trustworthiness of recommendation outcomes.

Reflection

This project highlighted the value of comparing star ratings with text-based sentiment rather than relying on either signal in isolation. The analysis showed that although rating-based models are internally consistent, a substantial proportion of reviews express sentiment that is not fully reflected by numeric ratings. This confirmed that customer opinions are often nuanced and cannot be accurately represented by stars alone.

An important learning outcome was recognising how modelling choices, particularly label construction and preventing data leakage, directly affect evaluation credibility. Ensuring that text-based and rating-based models were treated as independent sentiment signals was critical to producing meaningful agreement and reliability results.

For future work, the framework could be extended by improving the BERT model using domain-specific training data and by refining the reliability indicator to support real-world decision-making in recommendation systems.

Extra Challenge - High-Confidence Polarity Gap Detection

As an additional challenge, the section identified *high-confidence polarity gap* cases, where BERT expressed strong sentiment confidence (≥ 0.90) that contradicted both star-rating polarity and all rating-based classifiers (Figure 30). These reviews represent instances where customers clearly articulated emotional opinions in text that were not reflected by their numeric ratings. Such cases highlight the risk of relying on star ratings alone and further justify the need for reliability-aware sentiment interpretation.

```
gap_cases = df_cleaned[
    (df_cleaned["bert_score"] >= 0.90) &
    (df_cleaned["bert_sentiment_3"] != df_cleaned["rating_polarity_3"]) &
    (df_cleaned["ml_all_agree"])
]

len(gap_cases), round(len(gap_cases)/len(df_cleaned)*100, 2)
```

```
(36, 0.32)
```

Figure 30. Identification of High-Confidence Polarity Gap Reviews Based on BERT Confidence and Rating-Based Agreement commendation decisions

```
gap_cases[[
    "reviews",
    "rating_polarity_3",
    "bert_sentiment_3",
    "bert_score",
    "pred_lr","pred_svm","pred_mnb"
]].head(len(gap_cases))
```

	reviews	rating_polarity_3	bert_sentiment_3	bert_score	pred_lr	pred_svm	pred_mnb
531	kids like backup grands occasionally run like	positive	neutral	0.908972	positive	positive	positive
800	smells great feel awake lasts duration shower ...	neutral	positive	0.976781	positive	positive	positive
1519	good muffins good	negative	positive	0.929519	positive	positive	positive
1659	hard tell actually works great flavor kids acc...	neutral	positive	0.941276	positive	positive	positive
1841	absolutely buy cheap cheap cheap cheap thankfu...	negative	positive	0.908237	negative	negative	negative
2407	ingredients blood sugar levels	positive	neutral	0.904631	positive	positive	positive
2521	great item great product high high price	neutral	positive	0.971751	positive	positive	positive
2766	rose scent super excited find bundle rose incl...	neutral	positive	0.983914	positive	positive	positive
2979	happy product product well secure well knee bi...	negative	positive	0.951825	positive	positive	positive
3043	got big butt stool great wheeling around kitch...	neutral	positive	0.901133	positive	positive	positive
3443	bad best cotton used bad gets job done	positive	negative	0.921774	positive	positive	positive
3835	misleading literally paying empty bottle cloth...	neutral	negative	0.914068	negative	negative	negative
4098	exactly described used backdrop stand side opp...	positive	neutral	0.905822	positive	positive	positive
4819	love dropper always defective love product muc...	positive	negative	0.927360	positive	positive	positive
4974	cute smell good gift sister absolutely adorabl...	neutral	positive	0.965779	positive	positive	positive
5121	delivers far product works well feet happy fal...	negative	positive	0.905120	positive	positive	positive

Figure 31. Sample High-Confidence Polarity Gap Reviews Showing Disagreement Between Text-Based and Rating-Based Sentiment Models

Figure 31 shows that there are 36 reviews (0.32%) were classified as high-confidence polarity gap cases, where BERT expressed strong sentiment that directly contradicted both star-rating polarity and all rating-based classifiers, indicating rare but highly informative inconsistencies in customer feedback.

References

[1] A. Almansour, R. Alotaibi, and H. Alharbi, “Text-rating review discrepancy (TRRD): an integrative review and implications for research,” *Future Business Journal*, vol. 8, 2022, Art. no. 14, doi: 10.1186/s43093-022-00114-y. https://doi.org/10.1186/s43093-022-00114-y

[2] M. Fazzolari, V. Cozza, M. Petrocchi, and A. Spognardi, “A study on text-score disagreement in online reviews,” arXiv:1707.06932, 2017. https://doi.org/10.48550/arXiv.1707.06932

[3] R. Aralikatte, G. Sridhara, N. Gantayat, and S. M. Mani, “Fault in your stars: An analysis of Android app reviews,” in *Proc. CoDS-COMAD*, 2018, pp. 36–44, doi: 10.1145/3152494.3152500. https://doi.org/10.1145/3152494.3152500

[4] Amazon UK, “Lifetime Essential Fold-in-Half Table, Granite Gray,” product page. [Online]. Available: https://www.amazon.co.uk/Lifetime-Essential-Fold-Half-Granite/dp/B016PR4KD4

[5] S. Sadiq, M. Umer, S. Ullah, S. Mirjalili, V. Rupapara, and M. Nappi, “Discrepancy detection between actual user reviews and numeric ratings of Google App store using deep learning,” *Expert Systems with Applications*, vol. 181, p. 115111, 2021, doi: 10.1016/j.eswa.2021.115111. https://doi.org/10.1016/j.eswa.2021.115111

[6] U. o. C. S. Diego, "Amazon Review Data 23," 2023. [Online]. Available: https://amazon-reviews-2023.github.io/#for-user-reviews