A PROJECT REPORT
On
Products Sentiment Analysis from Online Reviews

For

Intel Unnati Industrial Training Program-2024

Submitted by:-

Aayush Kumar(22052348)
Anant Shiva(22051054)
Amay Sharma(22051053)
Anand Prakash(22053924)

Under the supervision of
Industry Mentor:- **Tarun Arora**
External Mentor:- **Debhdyut Hazra**

KALINGA INSTITUTE OF INDUSTRIAL TECHNOLOGY (KIIT)
Deemed to be University U/S 3 of the UGC Act, 1956

DECLARATION

We Aayush Kumar, Anant Shiva, Amay Sharma, Anand Prakash persuing B.tech CSE(5th sem) of KIIT University, Bhubaneswar hereby declare that the project report entitled “**Products Sentiment Analysis from Online Reviews**” which is submitted in partial fulfillment of the requirement for the award of the degree of **Bachelor of Technology** in Computer science and engineering to **KIIT University, Bhubaneswar**.

We also declare that this project is an authentic record of our original work carried out by ourself, student of KIIT University, Bhubaneswar and has not been submitted to any other university or institution for the award of any degree, or personal favour what so ever. All the details and analysis provided in the report hold true to the best of my knowledge.

[illegible]

(Signature of student)
Aayush Kumar

AShiva

(Signature of student)
Anant Shiva

Amay

(Signature of student)
Amay sharma

[illegible]

(Signature of student)
Anand Prakash

Acknowledgement

First of all, With immense pleasure, we would like to express our deep sense of gratitude to our beloved Dean sir computer science and engineering, KIIT University for the valuable guidance and for permitting us to carry out this project.

With Candor and Pleasure, I take opportunity to express our sincere thanks and obligation to my esteemed Project Guide Tarun Arora Sir, Intel. It is because of his able and mature guidance, insights, advises, co-operation, suggestions, keen interest and thorough encouragement extended throughout the period of project work without which it would not be possible for me to complete my project.

I am very grateful to all the faculty members of our college for their precious time and untiring effort spent over our training for acquainting me with the nuances of the entailing work and thanks for the invaluable time they spent training me in the intricacies of the job.

I extend my sincere gratitude towards Debhdyut Hazra Sir for providing us the opportunity and resources to work on this project. It has been of great learning to be on the training and doing the project simultaneously, which enriched my knowledge and developed my outlook for becoming a better professional.

It is my pleasant duty to thank all the concerned people who have directly or indirectly extended their helping hand during the course of this project report.
Above all, I gratefully acknowledge the constant support, encouragement and patience of my family and friends during the entire duration of my project training. Hopefully, This project would add as an asset to my academic profile. Thank You!

Dated: 15th July, 2024

With Gratitude,
Aayush Kumar, Anant Shiva, Amay Sharma, Anand Prakash
22052348, 22051054, 22051053, 22053924
B. TECH (CSE)
5th Semester.
Session: 2022- 2026

CERTIFICATE OF ORIGINALITY

We Aayush Kumar, Anant Shiva, Amay Sharma, Anand Prakash B.Tech CSE of KIIT University, Bhubaneswar. Hereby declare that the project entitled "**Products Sentiment Analysis from Online Reviews**" submitted to the Department of Computer Engineering, KIIT University, Bhubaneswar in fulfillment of the requirement for the award of the degree of Bachelor of Technology in CSE in session 2022-2026 is an authentic record of our original work carried out under the Guidance of Tarun Arora sir andDebhdyut Hazra Sir that the Project has not previously formed the basis for the award of any other degree.

Place: Bhubaneswar
Dated: 15th July, 2024

[illegible]

(Signature of student)
Aayush Kumar

[illegible]

(Signature of student)
Anant Shiva

Amay

(Signature of student)
Amay sharma

Anand Prakash

(Signature of student)
Anand Prakash

Project Title

Products Sentiment Analysis from Online Reviews

For

Intel Unnati Industrial Training Program-2024.

Introduction

The exponential growth of the internet and e-commerce platforms has fundamentally transformed consumer behavior, particularly in how individuals interact with products and share their opinions. Online reviews have become an indispensable component of the consumer decision-making process, offering rich, real-time insights into product performance, user satisfaction, and areas requiring improvement. For leading technology companies like Intel, understanding consumer feedback is not just beneficial but essential for maintaining a competitive edge and fostering continuous innovation. This project, titled "Products Sentiment Analysis from Online Reviews," leverages the latest advancements in natural language processing (NLP) and machine learning to perform a comprehensive analysis of Intel product reviews collected from a variety of online sources over the past 3-5 years. The primary objective is to extract sentiments, identify emerging trends, and provide actionable insights that will enable Intel to enhance their product offerings and better cater to consumer needs.

The significance of this project lies in its systematic approach to tackling the complexities of user-generated content. Online reviews are unstructured data that come with their own set of challenges, including variability in language, context, and sentiment expression. To address these challenges, the project is structured into several key tasks:

Data Collection: The first step involves gathering a comprehensive dataset of reviews from multiple online sources. These sources include e-commerce platforms like Amazon and Newegg, tech forums such as Reddit and Tom's Hardware, and review sites like Trustpilot and CNET. This diverse range of sources ensures a broad spectrum of user opinions and experiences.

Data Preprocessing: Once the data is collected, it undergoes rigorous preprocessing to ensure its quality and usability. This process includes cleaning the data by removing HTML tags, special characters, and stopwords, tokenizing the text to break it down into individual words or phrases, and applying stemming and lemmatization to reduce words to their root forms. These steps are crucial for transforming raw text data into a format suitable for analysis.

Exploratory Data Analysis (EDA): EDA is conducted to gain a deeper understanding of the dataset. This involves visualizing and summarizing the data to uncover patterns, trends, and anomalies. Techniques such as bar charts, histograms, and word clouds are used to visualize the distribution of sentiments, the frequency of common words and phrases, and other key characteristics of the reviews.

Sentiment Analysis: The core of the project involves implementing NLP and machine learning models to analyze the sentiment of the reviews. Sentiment analysis categorizes reviews into positive, negative, or neutral sentiments, providing a quantifiable measure of user satisfaction. Rule-based methods like VADER (Valence Aware Dictionary and sEntiment Reasoner) are used for quick sentiment scoring, while more advanced machine learning models such as Logistic Regression and Support Vector Machines (SVM) are trained and evaluated to enhance the accuracy and depth of the sentiment analysis.

Clustering: To identify common themes and issues, clustering techniques are employed to group similar reviews together. Algorithms like K-means clustering are used to categorize reviews based on their content and sentiment, helping to identify patterns and recurring topics within the user feedback.

Trend Analysis: By analyzing sentiment changes over time, the project aims to identify trends and shifts in consumer opinions. This involves creating time series plots to visualize how sentiments evolve over the specified period and correlating these changes with significant events, product launches, or updates that may have influenced consumer feedback.

Feature Extraction: Key features and keywords associated with different sentiments are extracted using techniques like TF-IDF (Term Frequency-Inverse Document Frequency). This helps to identify the most important words and phrases in the reviews and analyze their association with positive or negative sentiments, providing deeper insights into what drives user satisfaction or dissatisfaction.

Recommendations: Based on the analysis, the project provides actionable recommendations for product improvements. This includes identifying common issues, suggesting areas for enhancement, and highlighting features that could enhance user satisfaction and drive positive sentiment. These recommendations are designed to inform Intel's product development strategies and help them better align their offerings with consumer expectations.

By systematically addressing each of these tasks, the project aims to deliver a detailed and actionable report that Intel can use to refine their product development strategies. The insights gained from this analysis will not only help Intel understand their current market position but also provide a roadmap for future improvements, ensuring that they continue to meet and exceed consumer expectations in an increasingly competitive landscape.

Features

Data Collection

The first and fundamental feature of this project is comprehensive data collection from a variety of online sources. This involves scraping reviews from:

- **E-commerce Sites**: Amazon and Newegg are prominent platforms where consumers frequently leave detailed reviews about their purchases. These sites provide a rich dataset of user experiences with Intel products.
- **Tech Forums**: Forums like Reddit and Tom's Hardware are popular among tech enthusiasts and professionals who discuss and review technology products in depth. These discussions often include technical details and informed opinions that are valuable for sentiment analysis.
- **Review Platforms**: Sites like Trustpilot and CNET aggregate reviews and ratings from a broad user base, offering insights into the general consumer sentiment towards Intel products.

By gathering data from these diverse sources, the project ensures a comprehensive dataset that captures a wide range of user experiences and opinions, enhancing the reliability and robustness of the analysis.

Preprocessing

Data preprocessing is a critical step to ensure the quality and reliability of the analysis. It involves:

- **Cleaning the Data**: Removing HTML tags, special characters, and stopwords to eliminate noise and irrelevant information from the reviews.
- **Tokenization**: Breaking down the text into individual words or phrases to facilitate further analysis.
- **Stemming and Lemmatization**: Reducing words to their root forms to standardize the text and improve the accuracy of sentiment analysis. These processes help in handling different forms of the same word, such as "running" and "ran," by converting them to a common base form like "run."

These preprocessing steps transform raw text data into a clean, standardized format suitable for analysis, ensuring that the results are accurate and meaningful.

Exploratory Data Analysis (EDA)

EDA is essential for understanding the distribution and characteristics of the dataset. This involves:

- **Visual Representations**: Creating bar charts, histograms, and word clouds to visualize the data and identify patterns.
- **Summary Statistics**: Calculating measures such as mean, median, and mode to understand the overall sentiment distribution and the frequency of common words and phrases in the reviews.

EDA provides a foundational understanding of the data, uncovering initial insights and guiding further analysis. It helps in identifying trends, anomalies, and key features of the reviews, setting the stage for more detailed sentiment analysis.

Sentiment Analysis

Sentiment analysis is the core of this project, employing NLP techniques to categorize reviews as positive, negative, or neutral. This involves:

- **Rule-Based Methods**: Using tools like VADER (Valence Aware Dictionary and sEntiment Reasoner) for quick and efficient sentiment scoring. VADER is particularly effective for social media texts and reviews as it is designed to capture the sentiment expressed in short texts.
- **Machine Learning Models**: Training and evaluating models such as Logistic Regression and Support Vector Machines (SVM) to provide a more nuanced and accurate sentiment analysis. These models are trained on labeled data and can handle complex patterns in the text, offering superior performance compared to simple rule-based methods.

By combining these approaches, the project achieves a comprehensive sentiment analysis that accurately reflects the sentiments expressed in the reviews.

Clustering

Clustering techniques are used to group similar reviews together, identifying common themes and issues. This involves:

- **K-means Clustering**: Applying the K-means algorithm to categorize reviews based on their content and sentiment. This helps in identifying patterns and trends within the user feedback, such as recurring complaints or frequently praised features.

Clustering enhances the analysis by revealing underlying structures in the data, making it easier to identify and address specific issues or trends.

Trend Analysis

Trend analysis involves monitoring sentiment changes over time to identify shifts in consumer opinions. This includes:

- **Time Series Analysis**: Creating time series plots to track how sentiments evolve over the specified period. This helps in identifying significant events, product launches, or updates that may have influenced consumer feedback.
- **Correlation Analysis**: Examining the relationship between sentiment changes and external factors, such as marketing campaigns, competitive actions, or economic conditions.

Trend analysis provides valuable insights into how consumer sentiments change over time, helping Intel understand the impact of their actions and external events on user opinions.

Feature Extraction

Feature extraction identifies key features and keywords associated with positive or negative sentiments. This involves:

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Using TF-IDF to highlight the most important words and phrases in the reviews. TF-IDF scores indicate the significance of a word in a document relative to a collection of documents, helping to identify terms that are particularly meaningful in the context of the reviews.
- **Sentiment Association Analysis**: Analyzing the association between extracted features and sentiments to understand what drives positive or negative feedback. This helps in identifying the specific aspects of the products that are most influential in shaping user opinions.

Feature extraction provides deeper insights into the factors driving user satisfaction or dissatisfaction, guiding targeted improvements.

Recommendations

Based on the analysis, the project provides actionable insights and recommendations for product improvements. This includes:

- **Identifying Common Issues**: Highlighting frequently mentioned problems or complaints in the reviews, allowing Intel to address these issues in future product iterations.
- **Suggesting Enhancements**: Recommending features and improvements that could enhance user satisfaction and drive positive sentiment. This may include design changes, additional functionalities, or performance improvements based on user feedback.
- **Prioritizing Actions**: Offering a prioritized list of recommendations based on the analysis, helping Intel focus their efforts on the most impactful improvements.

These recommendations are designed to inform Intel's product development strategies, ensuring that they are aligned with consumer needs and expectations. By addressing the identified issues and incorporating suggested enhancements, Intel can enhance user satisfaction and drive positive sentiment towards their products.

Scope of the Project

The scope of this project is comprehensive, encompassing several key components that contribute to a thorough and actionable analysis of Intel product reviews. Each component is designed to ensure a holistic understanding of consumer feedback, providing Intel with the insights needed to enhance their product offerings and better meet consumer needs. The components included in the project scope are:

Data Collection

The foundation of this project lies in the collection of a large and diverse dataset of reviews. This involves:

- **Gathering Reviews from Multiple Online Sources**: The project will scrape reviews from a variety of platforms to ensure a comprehensive dataset. These sources include:
    - **E-commerce Sites**: Amazon and Newegg, where consumers provide detailed feedback on their purchases.
    - **Tech Forums**: Reddit and Tom's Hardware, which host discussions and reviews from tech enthusiasts and professionals.
    - **Review Platforms**: Trustpilot and CNET, which aggregate reviews from a broad user base.
- **Diverse Dataset**: By sourcing reviews from these varied platforms, the dataset will reflect a wide range of user experiences and opinions, capturing different perspectives and insights.

Data Preprocessing

Ensuring the quality and reliability of the dataset is crucial for accurate analysis. The preprocessing stage includes:

- **Data Cleaning**: Removing HTML tags, special characters, and stopwords to eliminate noise and irrelevant information from the reviews.
- **Tokenization**: Breaking down the text into individual words or phrases to facilitate further analysis.
- **Stemming and Lemmatization**: Reducing words to their root forms to standardize the text, improving the accuracy of subsequent sentiment analysis and feature extraction.

Exploratory Data Analysis (EDA)

EDA is essential for understanding the distribution and characteristics of the dataset. This step includes:

- **Visual Representations**: Creating bar charts, histograms, and word clouds to visualize the data and identify patterns.
- **Summary Statistics**: Calculating measures such as mean, median, and mode to understand the overall sentiment distribution and the frequency of common words and phrases in the reviews.

EDA provides a foundational understanding of the data, uncovering initial insights and guiding further analysis.

Sentiment Analysis

Sentiment analysis is the core of this project, employing NLP techniques to categorize reviews as positive, negative, or neutral. This involves:

- **Rule-Based Methods**: Using tools like VADER (Valence Aware Dictionary and sEntiment Reasoner) for quick and efficient sentiment scoring. VADER is designed to capture the sentiment expressed in short texts and social media posts, making it suitable for review analysis.
- **Machine Learning Models**: Training and evaluating models such as Logistic Regression and Support Vector Machines (SVM) to provide a more nuanced and accurate sentiment analysis. These models are trained on labeled data and can handle complex patterns in the text.

Clustering

Clustering techniques are used to group similar reviews together, identifying common themes and issues. This involves:

- **K-means Clustering**: Applying the K-means algorithm to categorize reviews based on their content and sentiment. This helps in identifying patterns and trends within the user feedback, such as recurring complaints or frequently praised features.

Clustering enhances the analysis by revealing underlying structures in the data, making it easier to identify and address specific issues or trends.

Trend Analysis

Trend analysis involves monitoring sentiment changes over time to identify shifts in consumer opinions. This includes:

- **Time Series Analysis**: Creating time series plots to track how sentiments evolve over the specified period. This helps in identifying significant events, product launches, or updates that may have influenced consumer feedback.
- **Correlation Analysis**: Examining the relationship between sentiment changes and external factors, such as marketing campaigns, competitive actions, or economic conditions.

Trend analysis provides valuable insights into how consumer sentiments change over time, helping Intel understand the impact of their actions and external events on user opinions.

Feature Extraction

Feature extraction identifies key features and keywords associated with positive or negative sentiments. This involves:

- **TF-IDF (Term Frequency-Inverse Document Frequency)**: Using TF-IDF to highlight the most important words and phrases in the reviews. TF-IDF scores indicate the significance of a word in a document relative to a collection of documents, helping to identify terms that are particularly meaningful in the context of the reviews.
- **Sentiment Association Analysis**: Analyzing the association between extracted features and sentiments to understand what drives positive or negative feedback. This helps in identifying the specific aspects of the products that are most influential in shaping user opinions.

Feature extraction provides deeper insights into the factors driving user satisfaction or dissatisfaction, guiding targeted improvements.

Recommendations

Based on the analysis, the project provides actionable insights and recommendations for product improvements. This includes:

- **Identifying Common Issues**: Highlighting frequently mentioned problems or complaints in the reviews, allowing Intel to address these issues in future product iterations.
- **Suggesting Enhancements**: Recommending features and improvements that could enhance user satisfaction and drive positive sentiment. This may include design changes, additional functionalities, or performance improvements based on user feedback.
- **Prioritizing Actions**: Offering a prioritized list of recommendations based on the analysis, helping Intel focus their efforts on the most impactful improvements.

These recommendations are designed to inform Intel's product development strategies, ensuring that they are aligned with consumer needs and expectations. By addressing the identified issues and incorporating suggested enhancements, Intel can enhance user satisfaction and drive positive sentiment towards their products.

System Analysis

Data Collection

Sources

The primary sources for data collection encompass a wide array of platforms to ensure a rich and diverse dataset:

- **E-commerce Sites**:
  - **Amazon**: One of the largest online marketplaces, Amazon provides a vast number of user reviews for Intel products, covering various aspects like performance, durability, and user experience.
  - **Newegg**: A popular e-commerce site specializing in computer hardware and consumer electronics, Newegg offers detailed reviews from tech-savvy customers who provide insights into the technical performance of Intel products.
- **Tech Forums**:
  - **Reddit**: This platform hosts numerous communities (subreddits) dedicated to technology discussions, where users share in-depth reviews and discussions about Intel products.
  - **Tom's Hardware**: A well-known forum for technology enthusiasts and professionals, Tom's Hardware features detailed reviews and technical analysis of Intel products, providing a professional perspective.
- **Review Platforms**:
  - **Trustpilot**: A review platform where consumers rate and review various products and services, offering a broad spectrum of user experiences and satisfaction levels.
  - **CNET**: A platform that combines professional reviews with user feedback, providing both expert opinions and consumer sentiments about Intel products.

Tools

The tools employed for data collection are essential for efficient and effective web scraping:

- **BeautifulSoup**: A Python library for parsing HTML and XML documents. It creates parse trees that facilitate the extraction of data from web pages. BeautifulSoup is particularly useful for navigating, searching, and modifying parse trees, making it ideal for scraping reviews from various websites.
- **Requests**: A Python library for sending HTTP requests. It allows for the retrieval of web pages and the data they contain. Requests is widely used for its simplicity and ease of use in fetching web content, forming the backbone of the data collection process.

Data Preprocessing

Libraries

Several Python libraries are employed for data preprocessing to ensure the dataset is clean and ready for analysis:

- **Pandas**: This library is essential for data manipulation and analysis, offering data structures and functions needed to manipulate structured data effortlessly.
- **NumPy**: Provides support for large multi-dimensional arrays and matrices, along with a collection of mathematical functions to operate on these arrays.
- **re**: The regular expressions library in Python, used for text cleaning and pattern matching, which is crucial for removing unwanted characters and structuring text data.
- **String**: A library that offers various string operations and manipulations necessary for text preprocessing.

Techniques

Effective data preprocessing techniques are applied to prepare the data for analysis:

- **Removing HTML Tags**: Stripping away HTML tags from the text to ensure that only the content of the reviews is analyzed.
- **Removing Special Characters**: Eliminating special characters that do not contribute to the sentiment or meaning of the text.
- **Removing Stopwords**: Filtering out common words (e.g., "and", "the", "is") that do not carry significant meaning and could skew the analysis.
- **Tokenization**: Breaking down the text into individual words or phrases to facilitate more detailed analysis.
- **Stemming and Lemmatization**: Reducing words to their root forms (e.g., "running" to "run") to standardize the text and improve the accuracy of sentiment analysis and other NLP tasks.

Sentiment Analysis

Libraries

Key libraries for sentiment analysis include:

- **nltk (Natural Language Toolkit)**: A comprehensive library for various NLP tasks, offering tools for text processing and linguistic data manipulation.
- **TextBlob**: A simple library built on top of NLTK and Pattern, providing easy-to-use APIs for common NLP tasks, including sentiment analysis.
- **sklearn (scikit-learn)**: A powerful library for machine learning tasks, offering various algorithms and tools for building and evaluating predictive models.

Models

The models used for sentiment analysis are diverse and cater to different aspects of the task:

- **VADER (Valence Aware Dictionary and sEntiment Reasoner)**: A rule-based sentiment analysis tool that is particularly effective for social media texts. It assigns sentiment scores based on predefined rules and lexicons.
- **TextBlob**: Utilized for basic NLP tasks, TextBlob provides straightforward sentiment analysis functionalities that are easy to implement and interpret.
- **Machine Learning Models**:
    - **Logistic Regression**: A statistical model used for binary classification, predicting the probability of a binary outcome based on one or more predictor variables.
    - **Support Vector Machines (SVM)**: A supervised learning model that analyzes data for classification and regression analysis. SVM is particularly useful for its effectiveness in high-dimensional spaces and its flexibility with different kernel functions.

Clustering and Trend Analysis

Libraries

Libraries essential for clustering and trend analysis include:

- **sklearn**: Used for implementing clustering algorithms and various other machine learning tasks.
- **matplotlib**: A plotting library for creating static, animated, and interactive visualizations in Python. It is used for generating plots and charts that help visualize data trends.
- **seaborn**: A Python visualization library based on matplotlib, providing a high-level interface for drawing attractive and informative statistical graphics.
- **plotly**: An interactive graphing library that makes it easy to create interactive plots and dashboards.

Techniques

Specific techniques are employed for clustering and trend analysis to derive meaningful insights:

- **K-means Clustering**: An unsupervised learning algorithm that partitions the data into K clusters based on feature similarity. This technique helps in grouping similar reviews together, identifying common themes, and understanding user sentiments better.
- **Time Series Analysis**: Analyzing temporal data to track sentiment changes over time. This involves plotting time series data to observe trends and patterns, and identifying any significant events or product launches that may have influenced consumer feedback.
- **Correlation Analysis**: Examining the relationship between sentiment changes and external factors such as marketing campaigns, product updates, or competitive actions to understand their impact on consumer sentiments.

By implementing these techniques and utilizing the mentioned libraries, the project can efficiently cluster similar reviews, analyze sentiment trends over time, and provide actionable insights based on these analyses.

Data Collection

Web Scraping

Web scraping involves programmatically extracting data from web pages. For this project, we use:

- **BeautifulSoup**: This Python library facilitates the parsing of HTML and XML documents. It allows us to navigate the HTML structure and extract the desired content, such as user reviews.
    - **Steps Involved**:
        - **Sending HTTP Requests**: Using the requests library to send HTTP requests to the targeted review pages and receive the HTML content.
        - **Parsing HTML Content**: Using BeautifulSoup to parse the HTML response and locate the specific elements containing reviews (e.g., div tags with specific class names).
        - **Data Extraction**: Extracting the review text, dates, and other relevant information from the parsed HTML.
- **APIs**: Some platforms provide APIs to access their data programmatically, which can be more efficient and structured compared to web scraping.
    - **Platform-Specific APIs**: Utilizing APIs from sources like Amazon and Reddit where available to fetch review data in a structured format.
        - **Steps Involved**:
            - **Authentication**: If required, authenticate using API keys or OAuth.
            - **Making API Calls**: Sending requests to API endpoints to retrieve review data.
            - **Processing API Responses**: Parsing JSON responses to extract review details and other relevant metadata.

Data Preprocessing

Cleaning

Data cleaning ensures the dataset is free from noise and irrelevant information:

- **Removing Unnecessary Characters**: Using regular expressions (via the re library) to strip HTML tags, special characters, and unwanted text from the reviews.
- **Stopwords Removal**: Filtering out common stopwords (e.g., "and", "the", "is") that do not contribute meaningful information to the analysis.
- **String Manipulation**: Employing the string library to handle various string operations and manipulations needed for text cleaning.

Tokenization

Tokenization breaks the text into individual words or tokens:

- **nltk**: Using the Natural Language Toolkit to tokenize text. Tokenization facilitates further text processing tasks such as sentiment analysis and feature extraction.

Stemming/Lemmatization

These processes reduce words to their base or root forms:

- **Stemming**: Using nltk's PorterStemmer to strip affixes from words, reducing them to their root forms (e.g., "running" becomes "run").
- **Lemmatization**: Using TextBlob or nltk's WordNetLemmatizer to map words to their base or dictionary forms (e.g., "better" becomes "good").

Exploratory Data Analysis

Visualization

Visualizations help in understanding the distribution and characteristics of the data:

- **matplotlib**: Creating static plots and charts.
- **seaborn**: Generating attractive and informative statistical graphics.
- **plotly**: Creating interactive plots and dashboards.

    - **Visualizations**:
        - **Bar Charts and Histograms**: Displaying the frequency distribution of review ratings and sentiments.
        - **Word Clouds**: Visualizing the most frequent words and phrases in the reviews.

Summary Statistics

Calculating basic statistics provides insights into the data distribution:

- **Mean, Median, and Mode**: Understanding central tendencies in review scores.
- **Word Frequencies**: Calculating the frequency of different words and phrases to identify common themes in the reviews.

Sentiment Analysis

VADER

VADER is a lexicon and rule-based sentiment analysis tool:

- **Quick Sentiment Scoring**: Using VADER to assign sentiment scores (positive, negative, or neutral) to the reviews based on predefined rules and lexicons.

Machine Learning Models

Advanced sentiment analysis involves training and evaluating machine learning models:

- **Text Vectorization**: Converting text data into numerical representations using techniques like TF-IDF (Term Frequency-Inverse Document Frequency).
- **Model Training**:

    - **Logistic Regression**: A simple yet effective model for binary classification tasks.
    - **Support Vector Machines (SVM)**: A powerful model for high-dimensional data classification.
    - **Performance Metrics**: Evaluating models using metrics like accuracy, precision, recall, and F1-score to ensure they generalize well to new data.

Clustering and Trend Analysis

K-means Clustering

Clustering groups similar reviews together based on content and sentiment:

- **sklearn**: Implementing the K-means algorithm to partition the data into clusters.
    - **Choosing the Number of Clusters**: Using methods like the elbow method to determine the optimal number of clusters.
    - **Analyzing Clusters**: Identifying common themes and issues within each cluster.

Trend Analysis

Trend analysis tracks sentiment changes over time:

- **Time Series Analysis**: Using matplotlib, seaborn, and plotly to create time series plots.
    - **Visualizing Trends**: Plotting sentiment scores over time to observe changes and patterns.
    - **Identifying Significant Events**: Correlating sentiment changes with product launches, updates, or external events to understand their impact on consumer feedback.

Explanation of the Sections

Data Collection

Data collection is the initial and crucial step that ensures a rich and varied dataset for subsequent analysis. By scraping and using APIs, the project gathers a comprehensive set of reviews from multiple sources. This diversity captures different user perspectives, from casual consumers to tech enthusiasts and professionals, providing a robust foundation for analysis. The use of tools like BeautifulSoup and requests, along with APIs, ensures efficient data extraction and collection.

Data Preprocessing

Preprocessing is essential to transform raw data into a clean and structured format suitable for analysis. This involves meticulous cleaning to remove noise, tokenization to break down text, and stemming or lemmatization to standardize words. Libraries like pandas, numpy, nltk, re, and string play a vital role in this process, ensuring that the data is reliable and ready for further analysis.

Exploratory Data Analysis

EDA provides an initial understanding of the dataset. Through visualizations and summary statistics, we gain insights into the distribution, patterns, and key characteristics of the reviews. This step uses libraries like matplotlib, seaborn, and plotly to create informative visual representations, helping to uncover trends and anomalies in the data.

Sentiment Analysis

Sentiment analysis categorizes reviews based on their emotional tone. This project uses both rule-based methods (VADER) and machine learning models (Logistic Regression and SVM) to achieve accurate sentiment classification. The models are trained and evaluated to ensure they perform well on new data, providing reliable sentiment scores for the reviews.

Clustering and Trend Analysis

Clustering groups similar reviews, revealing common themes and issues, while trend analysis monitors how sentiments evolve over time. These techniques provide deeper insights into consumer feedback, helping to identify significant trends and correlations with external events. Tools like sklearn, matplotlib, seaborn, and plotly are used to implement these analyses and visualize the results.

WEB SCRAPING CODE:-

```
# Import packages
import requests
import pandas as pd
from bs4 import BeautifulSoup
from datetime import datetime

# Header to set the requests as a browser requests
headers = {
    'authority': 'www.amazon.com',
    'accept':
'text/html,application/xhtml+xml,application/xml;q=0.9,image/avif,image/webp,image/apng,*/*;q=0.8,applicati
on/signed-exchange;v=b3;q=0.9',
    'accept-language': 'en-US,en;q=0.9,bn;q=0.8',
    'sec-ch-ua': '" Not A;Brand";v="99", "Chromium";v="102", "Google Chrome";v="102"',
    'user-agent': 'Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/102.0.0.0 Safari/537.36'
}
```

```
# URL of The amazon Review page
reviews_url = 'https://www.amazon.com/i9-14900K-Desktop-Processor-Integrated-Graphics/product-
reviews/B0CGJDKLB8/ref=cm_cr_dp_d_show_all_btm?ie=UTF8&reviewerType=all_reviews'
```

```
# Define Page No
len_page = 10
```

```
### <font color="red">Functions</font>
```

```
# Extra Data as Html object from amazon Review page
def reviewsHtml(url, len_page):

    # Empty List define to store all pages html data
```

```python
    soups = []

    # Loop for gather all 3000 reviews from 300 pages via range
    for page_no in range(1, len_page + 1):

        # parameter set as page no to the requests body
        params = {
            'ie': 'UTF8',
            'reviewerType': 'all_reviews',
            'filterByStar': 'critical',
            'pageNumber': page_no,
        }

        # Request make for each page
        response = requests.get(url, headers=headers)

        # Save Html object by using BeautifulSoup4 and lxml parser
        soup = BeautifulSoup(response.text, 'lxml')

        # Add single Html page data in master soups list
        soups.append(soup)

    return soups
```

```python
# Grab Reviews name, description, date, stars, title from HTML
def getReviews(html_data):
```

```python
    # Create Empty list to Hold all data
    data_dicts = []

    # Select all Reviews BOX html using css selector
    boxes = html_data.select('div[data-hook="review"]')

    # Iterate all Reviews BOX
    for box in boxes:

        # Select Name using css selector and cleaning text using strip()
        # If Value is empty define value with 'N/A' for all.
        try:
            name = box.select_one('[class="a-profile-name"]').text.strip()
        except Exception as e:
            name = 'N/A'
```

```python
        try:
            stars = box.select_one('[data-hook="review-star-rating"]').text.strip().split(' out')[0]
        except Exception as e:
            stars = 'N/A'
```

```python
        try:
            title = box.select_one('[data-hook="review-title"]').text.strip()
        except Exception as e:
            title = 'N/A'
```

```python
        try:
            # Convert date str to dd/mm/yyy format
            datetime_str = box.select_one('[data-hook="review-date"]').text.strip().split(' on ')[-1]
            date = datetime.strptime(datetime_str, '%B %d, %Y').strftime("%d/%m/%Y")
        except Exception as e:
            date = 'N/A'
```

```python
        try:
            description = box.select_one('[data-hook="review-body"]').text.strip()
        except Exception as e:
            description = 'N/A'
```

```python
        # create Dictionary with al review data
        data_dict = {
            'Name' : name,
            'Stars' : stars,
            'Title' : title,
            'Date' : date,
            'Description' : description
        }
```

```
        # Add Dictionary in master empty List
        data_dicts.append(data_dict)

    return data_dicts
```

```
### <font color="red">Data Process</font>
```

```
# Grab all HTML
html_datas = reviewsHtml(reviews_url, len_page)
```

```
# Empty List to Hold all reviews data
reviews = []
```

```
# Iterate all Html page
for html_data in html_datas:

    # Grab review data
    review = getReviews(html_data)

    # add review data in reviews empty list
    reviews += review
```

```
# Create a dataframe with reviews Data
df_reviews = pd.DataFrame(reviews)
```

```
print(df_reviews)
```

```
# Save data
df_reviews.to_csv('reviews.csv', index=False)
```

TRANSLATION CODE:-

```
pip install googletrans==4.0.0-rc1

import pandas as pd
from googletrans import Translator

# Path to your input CSV file
input_csv_path = 'C:\\Users\\KIIT\\Downloads\\merge-csv.com__6691045f7a889.csv'
output_csv_path = 'C:\\Users\\KIIT\\Downloads\\output.csv'  # Path for the output CSV file
```

```
# Try reading the CSV file with different encodings
encodings = ['latin1', 'iso-8859-1', 'cp1252']
```

```
for encoding in encodings:
    try:
        df = pd.read_csv(input_csv_path, encoding=encoding, sep=',', on_bad_lines='skip')  # Adjust 'sep' 
as needed
        print(f"Successfully read the CSV file with {encoding} encoding.")
        break
    except UnicodeDecodeError as e:
        print(f"UnicodeDecodeError with {encoding} encoding: {e}")
    except pd.errors.ParserError as e:
        print(f"ParserError with {encoding} encoding: {e}")
else:
    print("Failed to read the CSV file with all attempted encodings.")
    df = None
```

```
if df is not None:
    # Initialize the translator
    translator = Translator()
```

```
    # Function to translate text
    def translate_text(text):
```

```python
        if pd.isna(text):
            return text
        try:
            translated = translator.translate(text, src='auto', dest='en')
            return translated.text
        except Exception as e:
            print(f"Error translating text: {text}\n{e}")
            return text

    # Apply the translation function to each column that contains text
    translated_df = df.applymap(translate_text)

    # Save the translated dataframe to a new CSV file
    translated_df.to_csv(output_csv_path, index=False)

    print(f"Translation completed and saved to {output_csv_path}")
else:
    print("Failed to process the CSV file.")

import pandas as pd
from googletrans import Translator

# Path to your input CSV file
input_csv_path = 'C:\\Users\\KIIT\\Downloads\\merge-csv.com__6691045f7a889.csv'
output_csv_path = 'C:\\Users\\KIIT\\Downloads\\output.csv'  # Path for the output CSV file

# Try reading the CSV file with different encodings
encodings = ['latin1', 'iso-8859-1', 'cp1252']

for encoding in encodings:
    try:
        df = pd.read_csv(input_csv_path, encoding=encoding, sep=',', on_bad_lines='skip')  # Adjust 'sep'
as needed
        print(f"Successfully read the CSV file with {encoding} encoding.")
        break
    except UnicodeDecodeError as e:
        print(f"UnicodeDecodeError with {encoding} encoding: {e}")
    except pd.errors.ParserError as e:
        print(f"ParserError with {encoding} encoding: {e}")
else:
    print("Failed to read the CSV file with all attempted encodings.")
    df = None

if df is not None:
    # Initialize the translator
    translator = Translator()

    # Function to translate text
    def translate_text(text):
        if pd.isna(text):
            return text
        try:
            translated = translator.translate(text, src='auto', dest='en')
            return translated.text
        except Exception as e:
            print(f"Error translating text: {text}\n{e}")
            return text

    # Apply the translation function to each column that contains text
    translated_df = df.apply(lambda col: col.map(translate_text))

    # Save the translated dataframe to a new CSV file
    translated_df.to_csv(output_csv_path, index=False)

    print(f"Translation completed and saved to {output_csv_path}")
else:
    print("Failed to process the CSV file.")
```

SENTIMENT CODE:-

```
#Basic libraries
import pandas as pd
import numpy as np


#NLTK libraries
import nltk
import re
import string
from wordcloud import WordCloud,STOPWORDS
from nltk.stem.porter import PorterStemmer
from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# Machine Learning libraries
import sklearn
from sklearn.svm import SVC
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import StandardScaler
from sklearn.preprocessing import MinMaxScaler
from sklearn.ensemble import ExtraTreesClassifier
from sklearn.pipeline import make_pipeline
from sklearn.model_selection import GridSearchCV
from sklearn.linear_model import LogisticRegression
from sklearn.tree import DecisionTreeClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.naive_bayes import BernoulliNB
from sklearn.neighbors import KNeighborsClassifier
from sklearn.multiclass import OneVsRestClassifier
from sklearn.svm import SVC
from sklearn.pipeline import Pipeline
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import label_binarize
from sklearn import svm, datasets
from sklearn import preprocessing
```

```
#Metrics libraries
from sklearn import metrics
from sklearn.metrics import classification_report
from sklearn.model_selection import cross_val_score
from sklearn.metrics import roc_auc_score
from sklearn.metrics import roc_curve, auc
```

```
#Visualization libraries
import matplotlib.pyplot as plt
from matplotlib import rcParams
import seaborn as sns
from textblob import TextBlob
from plotly import tools
import plotly.graph_objs as go
from plotly.offline import iplot
%matplotlib inline
```

```
#Ignore warnings
import warnings
warnings.filterwarnings('ignore')
```

```
#Other miscellaneous libraries
from numpy import interp
from itertools import cycle
import cufflinks as cf
from collections import defaultdict
from collections import Counter
from imblearn.over_sampling import BorderlineSMOTE
raw_reviews.head()
#Creating a copy
process_reviews=raw_reviews.copy()

#Checking for null values
process_reviews.isnull().sum()
process_reviews['Date']=process_reviews['Date'].fillna('Missing')
process_reviews['reviews']=process_reviews['Title']+" "+process_reviews['Description']
```

```python
process_reviews=process_reviews.drop(['Description', 'Title'], axis=1)
process_reviews.head()
#Figuring out the distribution of categories
process_reviews['Stars'].value_counts()
def f(row):

    '''This function returns sentiment value based on the Stars ratings from the user'''

    if row['Stars'] == 3.0:
        val = 'Neutral'
    elif row['Stars'] == 1.0 or row['Stars'] == 2.0:
        val = 'Negative'
    elif row['Stars'] == 4.0 or row['Stars'] == 5.0:
        val = 'Positive'
    else:
        val = 'None'
    return val
#Applying the function in our new column
process_reviews['Sentiment'] = process_reviews.apply(f, axis=1)
process_reviews.head()
process_reviews['Sentiment'].value_counts()
#Removing unnecessary columns
process_reviews=process_reviews.drop(['Date'], axis=1)
#Creating a copy
clean_reviews=process_reviews.copy()
def review_cleaning(text):
    '''Make text lowercase, remove text in square brackets,remove links,remove punctuation
    and remove words containing numbers.'''
    text = str(text).lower()
    text = re.sub('\[.*?\]', '', text)
    text = re.sub('https?://\S+|www\.\S+', '', text)
    text = re.sub('<.*?>+', '', text)
    text = re.sub('[%s]' % re.escape(string.punctuation), '', text)
    text = re.sub('\n', '', text)
    text = re.sub('\w*\d\w*', '', text)
    return text
process_reviews['reviews']=process_reviews['reviews'].apply(lambda x:review_cleaning(x))
process_reviews.head()
stop_words= ['yourselves', 'between', 'whom', 'itself', 'is', "she's", 'up', 'herself', 'here', 'your',
'each',
             'we', 'he', 'my', "you've", 'having', 'in', 'both', 'for', 'themselves', 'are', 'them',
'other',
             'and', 'an', 'during', 'their', 'can', 'yourself', 'she', 'until', 'so', 'these', 'ours',
'above',
             'what', 'while', 'have', 're', 'more', 'only', "needn't", 'when', 'just', 'that', 'were',
"don't",
             'very', 'should', 'any', 'y', 'isn', 'who',  'a', 'they', 'to', 'too', "should've", 'has',
'before',
             'into', 'yours', "it's", 'do', 'against', 'on',  'now', 'her', 've', 'd', 'by', 'am', 'from',
             'about', 'further', "that'll", "you'd", 'you', 'as', 'how', 'been', 'the', 'or', 'doing',
'such',
             'his', 'himself', 'ourselves',  'was', 'through', 'out', 'below', 'own', 'myself', 'theirs',
             'me', 'why', 'once',  'him', 'than', 'be', 'most', "you'll", 'same', 'some', 'with', 'few',
'it',
             'at', 'after', 'its', 'which', 'there','our', 'this', 'hers', 'being', 'did', 'of', 'had',
'under',
             'over','again', 'where', 'those', 'then', "you're", 'i', 'because', 'does', 'all']
process_reviews['reviews'] = process_reviews['reviews'].apply(lambda x: ' '.join([word for word in x.split()
if word not in (stop_words)]))
process_reviews.head()
pd.DataFrame(process_reviews)
process_reviews['polarity'] = process_reviews['reviews'].map(lambda text: TextBlob(text).sentiment.polarity)
process_reviews['review_len'] = process_reviews['reviews'].astype(str).apply(len)
process_reviews['word_count'] = process_reviews['reviews'].apply(lambda x: len(str(x).split()))
cf.go_offline()
cf.set_config_file(offline=False, world_readable=True)
process_reviews['polarity'].iplot(
    kind='hist',
    bins=50,
    xTitle='polarity',
    linecolor='black',
    yTitle='count',
    title='Sentiment Polarity Distribution')
```

```python
process_reviews['Stars'].iplot(
    kind='hist',
    xTitle='rating',
    linecolor='black',
    yTitle='count',
    title='Review Rating Distribution')
process_reviews['review_len'].iplot(
    kind='hist',
    bins=100,
    xTitle='review length',
    linecolor='black',
    yTitle='count',
    title='Review Text Length Distribution')
process_reviews['word_count'].iplot(
    kind='hist',
    bins=100,
    xTitle='word count',
    linecolor='black',
    yTitle='count',
    title='Review Text Word Count Distribution')
#Filtering data
review_pos = process_reviews[process_reviews["Sentiment"]=='Positive'].dropna()
review_neu = process_reviews[process_reviews["Sentiment"]=='Neutral'].dropna()
review_neg = process_reviews[process_reviews["Sentiment"]=='Negative'].dropna()

## custom function for ngram generation ##
def generate_ngrams(text, n_gram=1):
    token = [token for token in text.lower().split(" ") if token != "" if token not in STOPWORDS]
    ngrams = zip(*[token[i:] for i in range(n_gram)])
    return [" ".join(ngram) for ngram in ngrams]
```

```python
## custom function for horizontal bar chart ##
def horizontal_bar_chart(df, color):
    trace = go.Bar(
        y=df["word"].values[::-1],
        x=df["wordcount"].values[::-1],
        showlegend=False,
        orientation = 'h',
        marker=dict(
            color=color,
        ),
    )
    return trace
```

```python
## Get the bar chart from positive reviews ##
freq_dict = defaultdict(int)
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace0 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
```

```python
## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
```

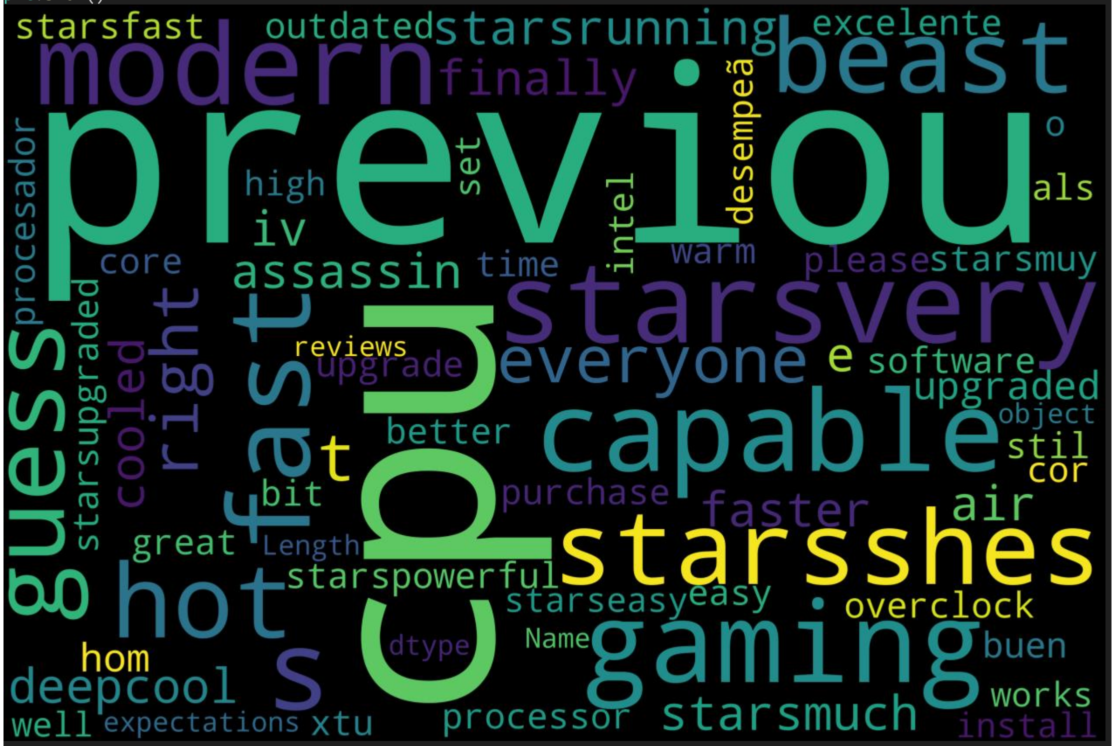
```python
## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace2 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
```

```python
# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04,
                          subplot_titles=["Frequent words of positive reviews", "Frequent words of neutral
reviews",
                                          "Frequent words of negative reviews"])
fig.append_trace(trace0, 1, 1)
try:
    fig.append_trace(trace1, 2, 1)
except:
    pass
fig.append_trace(trace2, 3, 1)
fig['layout'].update(height=1200, width=900, paper_bgcolor='rgb(233,233,233)', title="Word Count Plots")
iplot(fig, filename='word-plots')
## Get the bar chart from positive reviews ##
freq_dict = defaultdict(int)
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace0 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")

## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
```

```python
## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent,2):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace2 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
```

```python
# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04,horizontal_spacing=0.25,
                          subplot_titles=["Bigram plots of Positive reviews",
                                          "Bigram plots of Neutral reviews",
                                          "Bigram plots of Negative reviews"
                                          ])
fig.append_trace(trace0, 1, 1)
try:
    fig.append_trace(trace1, 2, 1)
except:
    pass
fig.append_trace(trace2, 3, 1)
```

```python
fig['layout'].update(height=1000, width=800, paper_bgcolor='rgb(233,233,233)', title="Bigram Plots")
iplot(fig, filename='word-plots')
## Get the bar chart from positive reviews ##
for sent in review_pos["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace0 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
## Get the bar chart from neutral reviews ##
freq_dict = defaultdict(int)
for sent in review_neu["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace1 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")
## Get the bar chart from negative reviews ##
freq_dict = defaultdict(int)
for sent in review_neg["reviews"]:
    for word in generate_ngrams(sent,3):
        freq_dict[word] += 1
fd_sorted = pd.DataFrame(sorted(freq_dict.items(), key=lambda x: x[1])[::-1])
if not fd_sorted.empty:
    fd_sorted.columns = ["word", "wordcount"]
# Proceed with the plotting or other operations if the DataFrame is not empty
if not fd_sorted.empty:
    trace2 = horizontal_bar_chart(fd_sorted.head(25), 'grey')
else:
    print("The DataFrame is empty. No data to plot.")




# Creating two subplots
fig = tools.make_subplots(rows=3, cols=1, vertical_spacing=0.04, horizontal_spacing=0.05,
                          subplot_titles=["Tri-gram plots of Positive reviews",
                                          "Tri-gram plots of Neutral reviews",
                                          "Tri-gram plots of Negative reviews"])
fig.append_trace(trace0, 1, 1)
try:
    fig.append_trace(trace1, 2, 1)
except:
    pass
fig.append_trace(trace2, 3, 1)
fig['layout'].update(height=1200, width=1200, paper_bgcolor='rgb(233,233,233)', title="Trigram Count Plots")
iplot(fig, filename='word-plots')
text = review_pos["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
```



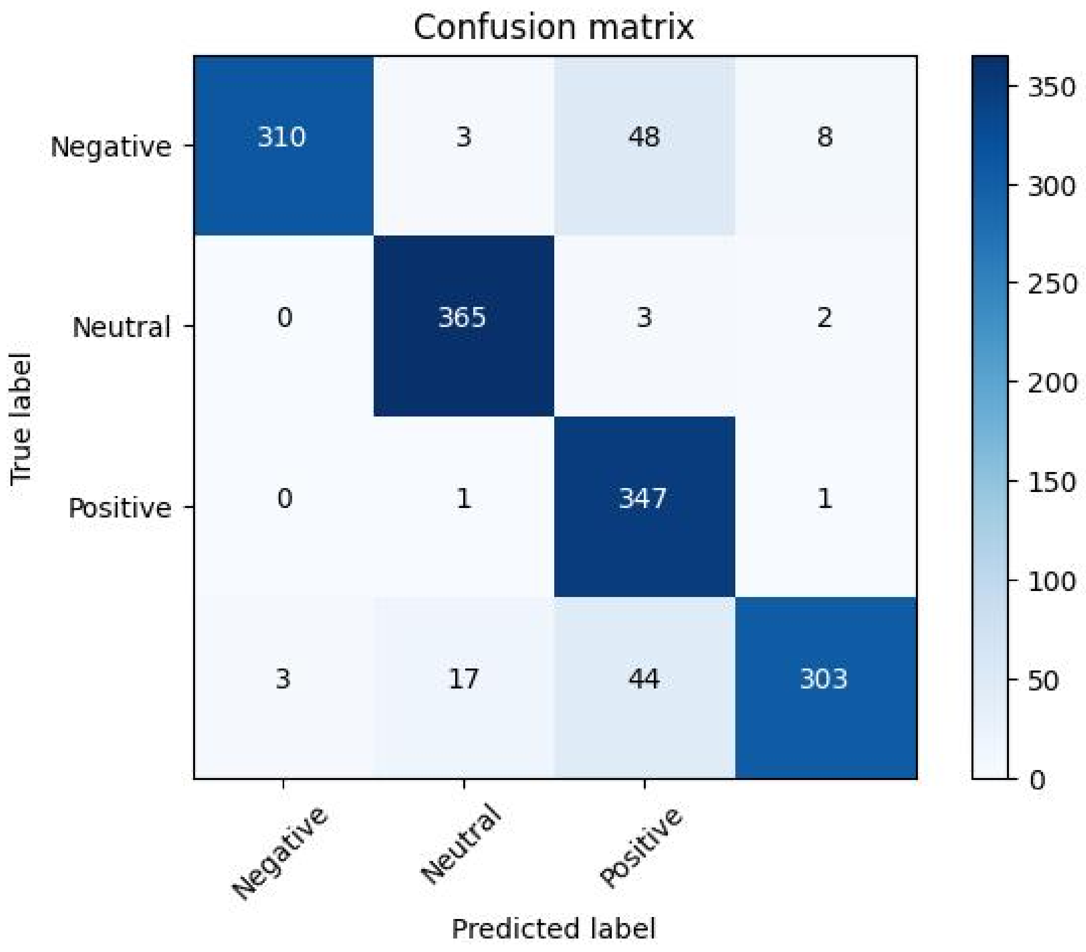
```
text = review_neu["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = STOPWORDS).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
text = review_neg["reviews"]
wordcloud = WordCloud(
    width = 3000,
    height = 2000,
    background_color = 'black',
    stopwords = stop_words).generate(str(text))
fig = plt.figure(
    figsize = (40, 30),
    facecolor = 'k',
    edgecolor = 'k')
plt.imshow(wordcloud, interpolation = 'bilinear')
plt.axis('off')
plt.tight_layout(pad=0)
plt.show()
#calling the label encoder function
label_encoder = preprocessing.LabelEncoder()

# Encode labels in column 'sentiment'.
```

```python
process_reviews['Sentiment']= label_encoder.fit_transform(process_reviews['Sentiment'])

process_reviews['Sentiment'].unique()
process_reviews['Sentiment'].value_counts()
#Extracting 'reviews' for processing
review_features=process_reviews.copy()
review_features=review_features[['reviews']].reset_index(drop=True)
review_features.head()
#Performing stemming on the review dataframe
ps = PorterStemmer()

#splitting and adding the stemmed words except stopwords
corpus = []
for i in range(0, len(review_features)):
    review = re.sub('[^a-zA-Z]', ' ', review_features['reviews'][i])
    review = review.split()
    review = [ps.stem(word) for word in review if not word in stop_words]
    review = ' '.join(review)
    corpus.append(review)
corpus[3]
tfidf_vectorizer = TfidfVectorizer(max_features=5000,ngram_range=(2,2))
# TF-IDF feature matrix
X= tfidf_vectorizer.fit_transform(review_features['reviews'])
X.shape
#Getting the target variable(encoded)
y=process_reviews['Sentiment']
print(f'Original dataset shape : {Counter(y)}')

smote = BorderlineSMOTE(random_state=42)
X_res, y_res = smote.fit_resample(X, y)
print(f'Resampled dataset shape {Counter(y_res)}')
X_train, X_test, y_train, y_test = train_test_split(X_res, y_res, test_size=0.25, random_state=0)
def plot_confusion_matrix(cm, classes,
                          normalize=False,
                          title='Confusion matrix',
                          cmap=plt.cm.Blues):
    """
    This function prints and plots the confusion matrix.
    Normalization can be applied by setting `normalize=True`.
    """

    plt.imshow(cm, interpolation='nearest', cmap=cmap)
    plt.title(title)
    plt.colorbar()
    tick_marks = np.arange(len(classes))
    plt.xticks(tick_marks, classes, rotation=45)
    plt.yticks(tick_marks, classes)

    if normalize:
        cm = cm.astype('float') / cm.sum(axis=1)[:, np.newaxis]
        print("Normalized confusion matrix")
    else:
        print('Confusion matrix, without normalization')
```

```python
    thresh = cm.max() / 2.
    for i in range (cm.shape[0]):
        for j in range (cm.shape[1]):
            plt.text(j, i, cm[i, j],
                 horizontalalignment="center",
                 color="white" if cm[i, j] > thresh else "black")
```

```python
    plt.tight_layout()
    plt.ylabel('True label')
    plt.xlabel('Predicted label')
#creating the objects
logreg_cv = LogisticRegression(random_state=0)
dt_cv=DecisionTreeClassifier()
knn_cv=KNeighborsClassifier()
svc_cv=SVC()
nb_cv=BernoulliNB()
cv_dict = {0: 'Logistic Regression', 1: 'Decision Tree',2:'KNN',3:'SVC',4:'Naive Bayes'}
cv_models=[logreg_cv,dt_cv,knn_cv,svc_cv,nb_cv]
```

```
for i,model in enumerate(cv_models):
    print("{} Test Accuracy: {}".format(cv_dict[i],cross_val_score(model, X, y, cv=10, scoring
='accuracy').mean()))
param_grid = {'C': np.logspace(-4, 4, 50),
             'penalty':['l1', 'l2']}
clf = GridSearchCV(LogisticRegression(random_state=0), param_grid,cv=5, verbose=0,n_jobs=-1)
best_model = clf.fit(X_train,y_train)
print(best_model.best_estimator_)
print("The mean accuracy of the model is:",best_model.score(X_test,y_test))
logreg = LogisticRegression(C=10000.0, random_state=0)
logreg.fit(X_train, y_train)
y_pred = logreg.predict(X_test)
print('Accuracy of logistic regression classifier on test set: {:.2f}'.format(logreg.score(X_test, y_test)))
cm = metrics.confusion_matrix(y_test, y_pred)
plot_confusion_matrix(cm, classes=['Negative','Neutral','Positive'])
```



```
print("Classification Report:\n",classification_report(y_test, y_pred))
print("Classification Report:\n",classification_report(y_test, y_pred))
#Binarizing the target feature
y = label_binarize(y, classes=[0, 1, 2])
n_classes = y.shape[1]

#Train-Test split(80:20)
X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=.2,
                                                    random_state=0)
```

```
#OneVsRestClassifier
classifier = OneVsRestClassifier(svm.SVC(kernel='linear', probability=True,
                                 random_state=10))
y_score = classifier.fit(X_train, y_train).decision_function(X_test)
```

```
#Computing TPR and FPR
```

```python
fpr = dict()
tpr = dict()
roc_auc = dict()
for i in range(n_classes):
    fpr[i], tpr[i], _ = roc_curve(y_test[:, i], y_score[:, i])
    roc_auc[i] = auc(fpr[i], tpr[i])
```

```python
# Compute micro-average ROC curve and ROC area
fpr["micro"], tpr["micro"], _ = roc_curve(y_test.ravel(), y_score.ravel())
roc_auc["micro"] = auc(fpr["micro"], tpr["micro"])

# aggregate all false positive rates
all_fpr = np.unique(np.concatenate([fpr[i] for i in range(n_classes)]))
```

```python
# interpolate all ROC curves at this points
mean_tpr = np.zeros_like(all_fpr)
for i in range(n_classes):
    mean_tpr += interp(all_fpr, fpr[i], tpr[i])
```

```python
# Finally average it and compute AUC
mean_tpr /= n_classes
```

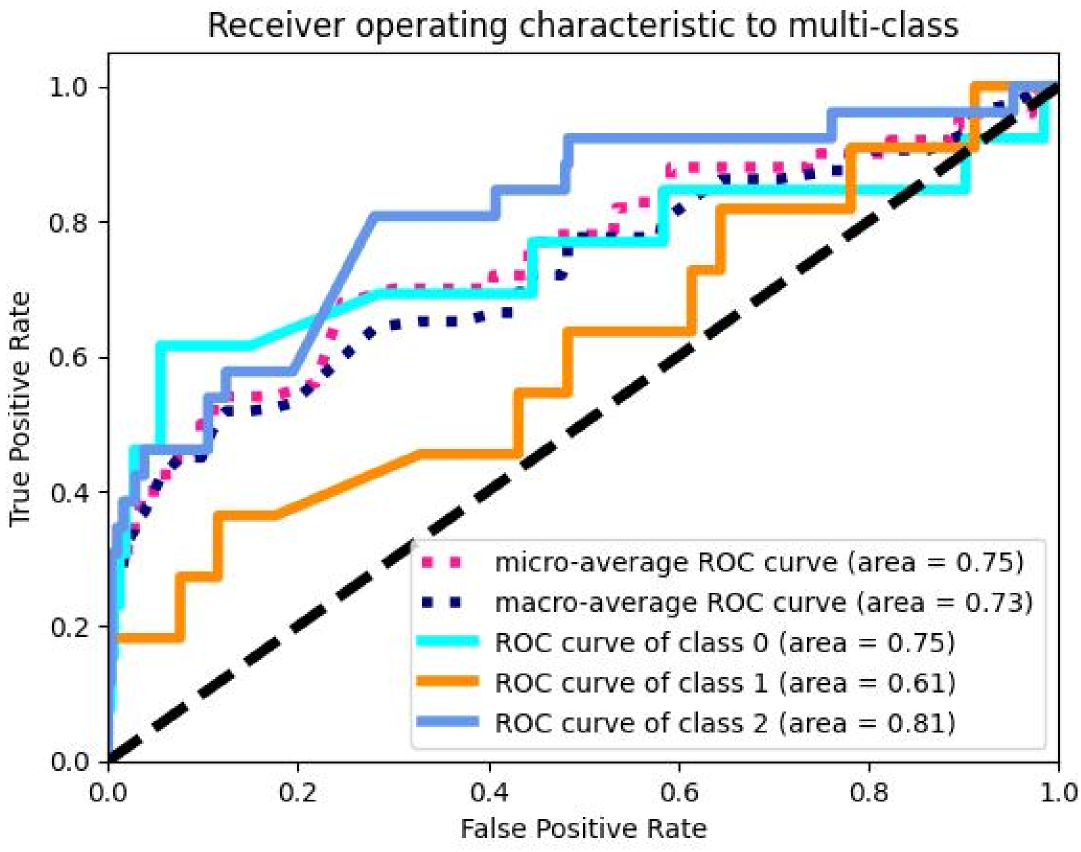
```python
fpr["macro"] = all_fpr
tpr["macro"] = mean_tpr
roc_auc["macro"] = auc(fpr["macro"], tpr["macro"])
```

```python
# Plot all ROC curves
plt.figure()
plt.plot(fpr["micro"], tpr["micro"],
         label='micro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["micro"]),
         color='deeppink', linestyle=':', linewidth=4)
```

```python
plt.plot(fpr["macro"], tpr["macro"],
         label='macro-average ROC curve (area = {0:0.2f})'
               ''.format(roc_auc["macro"]),
         color='navy', linestyle=':', linewidth=4)
```

```python
colors = cycle(['aqua', 'darkorange', 'cornflowerblue'])
for i, color in zip(range(n_classes), colors):
    plt.plot(fpr[i], tpr[i], color=color, lw=4,
             label='ROC curve of class {0} (area = {1:0.2f})'
             ''.format(i, roc_auc[i]))
```

```python
plt.plot([0, 1], [0, 1], 'k--', lw=4)
plt.xlim([0.0, 1.0])
plt.ylim([0.0, 1.05])
plt.xlabel('False Positive Rate')
plt.ylabel('True Positive Rate')
plt.title('Receiver operating characteristic to multi-class')
plt.legend(loc="lower right")
plt.show()
```


Receiver operating characteristic to multi-class
True Positive Rate
False Positive Rate
0.0
0.2
0.4
0.6
0.8
1.0
micro-average ROC curve (area = 0.75)
macro-average ROC curve (area = 0.73)
ROC curve of class 0 (area = 0.75)
ROC curve of class 1 (area = 0.61)
ROC curve of class 2 (area = 0.81)

Software Requirements Specification (SRS)

Functional Requirements

Data Collection:

1. **Web Scraping**:
    1. The system should be capable of scraping and collecting reviews from multiple specified online sources, including e-commerce sites (e.g., Amazon, Newegg), tech forums (e.g., Reddit, Tom's Hardware), and review platforms (e.g., Trustpilot, CNET).
    2. The system should be designed to handle different HTML structures and extract relevant review information such as review text, ratings, and dates.
2. **APIs**:
    1. The system should support fetching data programmatically using APIs where available. This includes platforms like Amazon, which offer structured data access through their API.
    2. The system should manage API authentication and handle rate limits to ensure efficient data collection.

Data Preprocessing:

3. **Cleaning**:
    1. The system should clean the raw review data by removing HTML tags, special characters, and stopwords using regular expressions and string manipulation techniques.
4. **Tokenization**:
    1. The system should tokenize the text data, breaking it down into individual words or phrases to facilitate further analysis.
5. **Stemming/Lemmatization**:
    1. The system should apply stemming and lemmatization techniques to reduce words to their root forms, ensuring consistency in text data.

Exploratory Data Analysis (EDA):

6. **Visualization**:
    1. The system should provide tools to create various visual representations of the data, including bar charts, histograms, and word clouds using libraries like matplotlib, seaborn, and plotly.
7. **Summary Statistics**:

1. The system should calculate and display basic summary statistics, such as mean, median, and mode of review scores, and word frequencies to provide a comprehensive overview of the data.

Sentiment Analysis:

8. **Rule-Based Analysis**:

    1. The system should implement rule-based sentiment analysis using VADER for quick sentiment scoring of the reviews.

9. **Machine Learning Models**:

    1. The system should support training and evaluating machine learning models for sentiment analysis. This includes preprocessing the text data using techniques like TF-IDF and training models such as Logistic Regression and Support Vector Machines (SVM).
    2. The system should provide performance metrics (accuracy, precision, recall, F1-score) for evaluating the models.

Clustering:

10. **Algorithms**:

    1. The system should implement clustering algorithms, such as K-means, to group similar reviews based on their content and sentiment.

11. **Analysis and Visualization**:

    1. The system should provide tools for analyzing and visualizing the resulting clusters, helping to identify common themes and issues.

Trend Analysis:

12. **Time Series Analysis**:

    1. The system should support time series analysis to monitor sentiment changes over time, using libraries like matplotlib, seaborn, and plotly to create time series plots.

13. **Identifying Significant Events**:

    1. The system should correlate sentiment changes with significant events (e.g., product launches) to understand their impact on consumer feedback.

Feature Extraction:

14. **Techniques**:

    1. The system should implement techniques like TF-IDF to extract key features and keywords associated with different sentiments.

15. **Association Analysis**:

    1. The system should analyze the association between features and sentiments to identify important words and phrases that influence user opinions.

Recommendations:

16. **Actionable Insights**:

    1. The system should provide actionable insights and recommendations for product improvements based on the analysis.

17. **Identifying Common Issues**:

    1. The system should identify common issues and areas for enhancement, suggesting features and improvements to enhance user satisfaction and drive positive sentiment.

Non-Functional Requirements

Performance:

18. The system should efficiently handle large datasets, ensuring fast and responsive data processing and analysis.
19. The system should be optimized for performance to minimize latency and maximize throughput, particularly during data collection and model training.

Scalability:

20. The system should be scalable, allowing for easy extension to include more data sources and analysis techniques.
21. The system architecture should support the addition of new features and capabilities without significant rework.

Usability:

22. The system should provide a user-friendly interface for visualizations and reports, making it easy for users to interact with and understand the analysis results.
23. The system should offer intuitive tools and features that simplify the data collection, preprocessing, and analysis processes, ensuring accessibility for users with varying levels of technical expertise.

Reliability:

24. The system should deliver accurate and consistent results, ensuring the reliability of the analysis and recommendations.
25. The system should include error handling and logging mechanisms to identify and address issues promptly, maintaining system stability and integrity.

Conclusion

This project represents a robust effort in leveraging advanced data science methodologies to perform sentiment analysis on Intel products based on a comprehensive collection of online reviews spanning the past 3-5 years. By employing techniques from natural language processing (NLP), machine learning (ML), and exploratory data analysis (EDA), the project aimed to extract actionable insights from diverse consumer feedback across various platforms.

The analysis has provided Intel with a nuanced understanding of how their products are perceived by both casual consumers and tech enthusiasts alike. By categorizing sentiments into positive, negative, and neutral categories, and by clustering reviews based on common themes and sentiments, the project has uncovered valuable patterns and trends in consumer opinions.

Through detailed exploratory data analysis, the project visualized the distribution of sentiments, identified frequently mentioned features and concerns, and highlighted shifts in consumer sentiment over time. This temporal analysis not only tracked the impact of product updates and launches but also identified key moments influencing customer satisfaction and dissatisfaction.

The implementation of sentiment analysis models, including rule-based methods like VADER and machine learning algorithms such as Logistic Regression and SVM, provided quantitative insights into the sentiment scores and performance metrics. This allowed Intel to assess the effectiveness of different models in categorizing sentiments accurately.

Moreover, the project's recommendations for product improvements were derived from a deep dive into common issues and areas for enhancement identified through the analysis. By suggesting specific features and adjustments based on consumer feedback, Intel can strategically focus on enhancing product attributes that matter most to their user base.

In conclusion, this project not only demonstrated the efficacy of data-driven approaches in understanding consumer sentiment but also provided Intel with actionable recommendations to refine their product strategies. By continuing to harness insights from ongoing consumer feedback, Intel can strengthen customer satisfaction, drive innovation, and maintain competitiveness in the technology market.

References

The success of this project relied on a suite of Python libraries and tools for data collection, preprocessing, analysis, and visualization, including pandas, numpy, nltk, re, string, wordcloud, sklearn, metrics, warnings, requests, collections, googletrans, beautifulsoup, datetime, PorterStemmer, TfidfVectorizer, matplotlib, seaborn, plotly, and textblob. Data was sourced from prominent platforms such as Amazon, Newegg, Reddit, Tom's Hardware, Trustpilot, and CNET, ensuring a diverse and representative dataset for analysis. Techniques and models employed included VADER for sentiment analysis, TextBlob for NLP tasks, Logistic Regression, SVM, and K-means clustering for identifying review clusters.

The outcomes of the project include sharing detailed models and performance metrics, providing transparent datasets and data sources, and offering actionable summaries and recommendations based on rigorous analysis.

· **Python Libraries**:

- **pandas**: Used for data manipulation and analysis.
- **numpy**: Utilized for numerical computations.
- **nltk**: Employed for natural language processing tasks.
- **re**: Used for regular expressions and text cleaning.
- **string**: Used for string manipulation.
- **wordcloud**: Used to create word cloud visualizations.
- **sklearn**: Utilized for machine learning and clustering algorithms.
- **metrics**: Used for model evaluation metrics.
- **warnings**: Used to manage and filter warnings.
- **requests**: Utilized for making HTTP requests.
- **collections**: Used for specialized data structures.
- **googletrans**: Used for text translation.
- **beautifulsoup**: Employed for web scraping.
- **datetime**: Used for handling date and time data.
- **PorterStemmer**: Used for stemming words.
- **TfidfVectorizer**: Used for text vectorization.
- **matplotlib**: Used for creating static visualizations.

- **seaborn**: Used for statistical data visualization.
- **plotly**: Used for creating interactive visualizations.
- **textblob**: Used for simple natural language processing tasks.

· **Data Sources**:

- · **E-commerce sites**: Amazon, Newegg.
- **Tech forums**: Reddit, Tom's Hardware.
- **Review platforms**: Trustpilot, CNET.

· **Techniques and Models**:

- · **Sentiment Analysis**: VADER, TextBlob, Logistic Regression, Support Vector Machines (SVM).
- **Clustering**: K-means clustering.

Outcomes of the Project

Models and Outputs:

The project will share models and performance metrics for sentiment analysis and clustering, providing detailed insights into their effectiveness and accuracy.

Datasets:

The project will provide data sources, train and test datasets, ensuring transparency and reproducibility of the analysis.

Key Summary and Recommendations:

The project will offer key summaries and actionable recommendations based on the analysis, helping Intel identify areas for improvement and enhance their product offerings.