Intel Unnati Industrial Training Program 2024

INTEL PRODUCTS SENTIMENT ANALYSIS FROM ONLINE REVIEWS

TEAM
BYTE BLITZ

JADA VISWA CHAITANYA SAI
JYOTISKA BOSE

KALINGA INDUSTRIAL INSTITUTE OF TECHNOLOGY

Acknowledgement

We would like to extend our sincerest of gratitude to the individuals and organizations whose support and collaboration have been instrumental in the completion of this project.

Special thanks to professor Dr. Soumya Ranjan Mishra, our dedicated advisor for his unwavering guidance, insightful feedback, and continuous encouragement throughout the process.

We would like to express our appreciation to the dedicated team at Intel for their collaborative efforts and holding multiple sessions to help us throughout the process which equipped us with the necessary knowledge and expertise to complete this project. This endeavor would not have been possible without the generous support from Intel and thorough guidance of Mr. Debdyut Hazra.

Lastly, we would like to thank KIIT without whom we would not have had the opportunity of undertaking this journey.

-

Table of Contents

Abstract

Due to the evolution of Web 1.0 to Web 3.0, it has never been easier for users to generate and share their thoughts, views and approaches on the Internet. This leads to an explosive growth of subjective information (opinion data) on the Web. Therefore, goal in collecting, studying and exploiting this information is increasing from which several concepts have appeared such as Sentiment Analysis (SA) which focuses on opinion mining (identification and classification) from textual data.

In this project, we conducted sentiment analysis on customer reviews of Intel processors sourced from prominent e-commerce platforms like Amazon, BestBuy, and Flipkart. Leveraging web scraping techniques for real-time data collection we created an extensive dataset with 7k+ reviews worldwide. We used advanced machine learning models such as RoBERTa, VADER and others to accurately classify sentiments (positive, negative, neutral) expressed in these reviews. The predictions from these models uncovered nuanced insights into customer perceptions, preferences, and areas for improvement, thereby offering valuable guidance for Intel's product strategy and marketing initiatives by means of thorough EDA and recommendations based on negative customer reviews.

Sentiment Analysis like this underscores the pivotal role of sentiment analysis in deriving actionable intelligence from large-scale consumer feedback to drive product innovation and customer satisfaction in the tech industry.

-

Introduction

Background:

Sentiment analysis, often known as opinion mining, looks at how individuals feel about particular things. Sentiment analysis is a subfield of computational linguistics and NLP that deals with techniques for extracting, categorizing, comprehending, and assessing the opinions expressed in online publications. It is also known as opinion mining of texts. The advancement of sentiment analysis has been made easier by the abundance of online text data, especially when it comes to speculating on people's attitudes, opinions, and beliefs. Sentiment analysis has been widely utilized to forecast public sentiment and trends in a variety of scenarios. Academics studying political communication use sentiment analysis on social media posts to gather public opinion on presidential candidates and to precisely forecast election outcome.

Online commerce is one of the economic areas in the contemporary world that is expanding the quickest. Nowadays, a lot of goods are purchased through online merchants. Reviews frequently affect online product purchases. Therefore, the importance of finding fake reviews is increasing, and the success of a system for identifying bogus reviews relies heavily on sentiment analysis

Objective:

This project aims to conduct a comprehensive sentiment analysis on customer reviews of Intel processors to classify sentiments accurately. By analyzing and categorizing sentiments expressed across various platforms, including e-commerce giants and specialized technical review sites, our aim is to derive actionable insights that can inform Intel's product enhancement strategies. This includes identifying strengths, weaknesses, and areas for improvement based on customer feedback gathered from multiple sources over the past 3-4 years.

Methods:

To achieve this objective, we utilized a multi-faceted approach encompassing web scraping, data preprocessing, and advanced machine learning techniques:

- Web Scraping: Reviews were collected in real-time from e-commerce platforms such as Amazon, BestBuy, and Flipkart using custom scrapers built with BeautifulSoup and Selenium.
- Data Preprocessing: The collected data underwent thorough preprocessing, including lemmatization, stop-word removal, punctuation cleaning, and normalization to ensure data quality and consistency.
- Machine Learning Models: Several state-of-the-art machine learning models were employed, including RoBERTa, VADER, TextBlob, XGBoost, Random Forest, and Sentiment Vectorization (Sector Vector). These models were chosen for their ability to accurately classify sentiments and extract meaningful insights from large volumes of textual data.

Findings:

Our analysis yielded insightful findings:

- Sentiment Classification: We classified sentiments (positive, negative, neutral) expressed in customer reviews with high accuracy using RoBERTa and VADER models.
- Affinity Clustering: Through exploratory data analysis, we identified clusters of reviews with similar sentiments, providing a deeper understanding of customer preferences and concerns.
- Sentiment Trends: Analysis over time revealed evolving trends in customer sentiment towards Intel processors, highlighting shifts in consumer perceptions and expectations.

Scope:

This study encompasses a broad scope of data sources and an analysis timeframe spanning from **2020 to 2024**, i.e **3-4 years**.

- Data Sources: Reviews have been collected from prominent e-commerce platforms such as Amazon, BestBuy, and Flipkart, where customers provide detailed feedback on Intel processors. Additionally, reviews from technical review sites such as Tom's Hardware and PCMag have been included to capture expert opinions and in-depth technical assessments on a per product basis.

- Our Dataset Scraped from the sites consists of **7k+** Product Reviews.

- Analysis Timeframe: The analysis will cover reviews accumulated over the past **3-4 years** i.e ranging from **2020 to 2024** allowing us to capture the latest consumer sentiments and trends over time.

-

Literature Review

Significance of User Reviews:

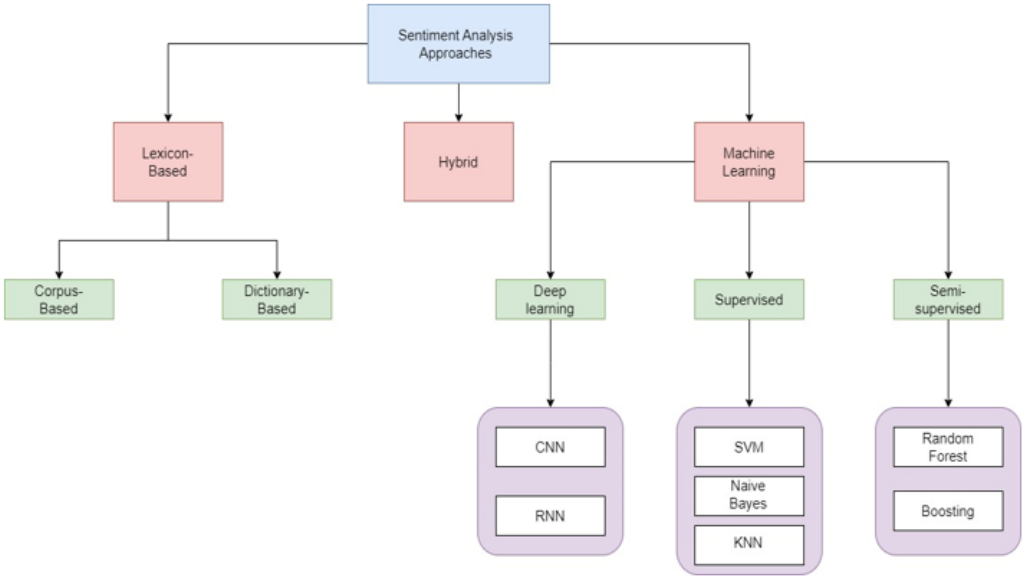
Customer feedback plays a pivotal role in shaping purchasing decisions and guiding product development strategies in the tech industry. Reviews provided by consumers on platforms such as Amazon, BestBuy, Flipkart, and specialized technical review sites offer direct insights into product performance, usability, and customer satisfaction. These reviews not only influence potential buyers but also serve as valuable feedback loops for companies like Intel to iterate and improve their product offerings.

Previous Studies and Approaches:

Previous research in sentiment analysis of tech product reviews has utilized various methodologies to extract meaningful insights from textual data. Studies have employed both rule-based and machine learning approaches to classify sentiments (positive, negative, neutral) expressed in reviews. Techniques such as Natural Language Processing (NLP), sentiment lexicons (e.g., VADER), machine learning models (e.g., SVM, LSTM), and deep learning architectures (e.g., BERT) have been extensively applied to analyze and categorize sentiment across diverse datasets. Researchers have focused on enhancing accuracy, scalability, and interpretability of sentiment analysis models to meet the evolving demands of consumer feedback analysis.

Sentiment Analysis Techniques and Visualization:

Sentiment analysis utilizes various techniques and tools to effectively extract, quantify, and visualize sentiment from textual data. These techniques can be broadly categorized into Lexicon-based Models, Machine Learning Models and Hybrid Models, and various visualization methods:



1. Lexicon-based Approaches:

Lexicon-based methods utilize sentiment lexicons or dictionaries that contain predefined sentiment scores for words and phrases based on their emotional context. In our project, we have made use of the VADER (Valence Aware Dictionary and sEntiment Reasoner) lexicon, which assigns polarity scores (positive, negative, neutral) to words and computes an overall sentiment score for sentences or documents and TextBlob.

I. **VADER**: This model uses a lexicon (a predefined dictionary of sentiment words) and rule-based heuristics to assess the sentiment of text.

II. **TextBlob**: While TextBlob incorporates machine learning for some tasks, its sentiment analysis module primarily relies on a lexicon-based approach with pattern matching.

2. Machine Learning Models:

Machine learning (ML) models in sentiment analysis involve training supervised classifiers to automatically categorize text into sentiment categories (e.g., positive, negative, neutral). The algorithms used in this project include Support Vector Machines (SVM), Random Forest, and XGBoost Models.

I. **Random Forest**: Random Forest is an ensemble learning method that constructs multiple decision trees during training and outputs the class that is the mode of the classes (classification) or mean prediction (regression) of the individual trees.

II. **Support Vector Machine (SVM)**: SVM is a supervised machine learning model used for classification and regression analysis. It finds an optimal hyperplane that separates data points into different classes.

III. **XGBoost**: XGBoost is a machine learning algorithm, specifically an implementation of gradient boosted decision trees, used for classification and regression tasks.

3. Hybrid Learning Models:

A hybrid model in the context of natural language processing combines elements of both rule-based or heuristic approaches (like lexicon-based methods) and data-driven techniques (such as machine learning or deep learning). In this project, the hybrid model used is RoBERTa.

RoBERTa: It combines unsupervised pre-training on large text corpora using self-supervised learning methods like masked language modeling. After pre-training, it fine-tunes its parameters on specific downstream tasks such as text classification or question answering, adapting its representations for optimal performance on these tasks.

-

Environments

1. **Jupyter Notebook**: Used primarily for interactive data analysis, exploration, and prototyping of machine learning models.

2. **Visual Studio Code** (VSCode): Served as the main integrated development environment (IDE) for writing, debugging, and managing Python scripts.

3. **Google Colab**: Leveraged for its cloud-based computing resources and collaborative features, it facilitated the execution of resource-intensive tasks such as training machine learning models on GPUs, without the need for local hardware upgrades.

4. Python: Chosen as the primary programming language due to its rich ecosystem of libraries for natural language processing (NLP) and machine learning. Some of the used Python libraries are as follows,

    - Transformers: Leveraged through the Hugging Face library for state-of-the-art natural language processing tasks, including sentiment analysis with models like Roberta.

    - TextBlob: Utilized for its simple yet effective sentiment analysis capabilities and text processing utilities.

    - RoBERTa: Specifically used from the transformers library for advanced sentiment analysis tasks, benefiting from its deep learning architecture and pre-trained model weights.

    - VADER: Employed for its rule-based sentiment analysis approach, particularly useful for quick sentiment polarity scoring of text.

    - XGBoost: Applied for gradient boosting-based machine learning tasks, including sentiment analysis, to handle non-linear relationships in data effectively.

    - NLTK (Natural Language Toolkit): Utilized for various NLP tasks such as text preprocessing (e.g., stop-word removal, tokenization) and language detection.

    - WordCloud: Employed to visualize word frequency and patterns in textual data, providing intuitive insights into sentiment-related keywords.

    - LangDetect: Integrated for language detection, crucial for identifying and handling multilingual data appropriately within the sentiment analysis pipeline.

    - Topic Modeling: Implemented to uncover underlying themes or topics within textual data, enhancing the understanding of sentiment trends across different categories.

Github Link

The Project is hosted at the following link:

https://github.com/viswachaitanyasai/Intel-Products-Sentiment-Analysis

Project Workflow

1. Web Scraping:
   - Scrape Intel processor reviews and related data from various e-commerce sites and store in CSV format.
   - Perform initial data preprocessing including steps like remove stop-words, punctuation, and convert text to lowercase.

2. Data Preprocessing & Cleaning:
   - **Translate non-English reviews** using Python libraries.
   - Perform Data Preprocessing such as Lowercase conversion, removal of punctuation, removal of numerical expressions, **removal of stop-words**, **tokenization** and **lemmatization**.
   - We perform further translation in **Google Sheets** using formulae (=GoogleTranslate(A5, "auto", “en") so that any reviews that might have been missed before are translated, ensuring data integrity and consistency.

3.Initial EDA:
   - We perform an Initial EDA on the dataset scraped from the sites to get a better understanding of our dataset and analyze the data present in it using various graphs and figures.

4. Scrape Technical Reviews:
   - Gather technical reviews from specific sites and store them in a CSV format.

5. Sentiment Analysis:
   - VADER (pre-trained): Analyze sentiment based on our cleaned dataset (having 7k+ Reviews).
     - TextBlob (pre-trained): Perform sentiment analysis as a comparative measure.
     - RoBERTa (fine-tuned): Trained on a dataset sourced from Kaggle (having 32k+ Reviews) to increase model accuracy
     - Train RoBERTa on VADER's output based on ratings.
     - Train RoBERTa on sentiment-based output derived from VADER.
     - Determine that RoBERTa trained on sentiments yields higher accuracy.
   - Use the RoBERTa sentiment-trained data to train other models:
     - XGBoost, Random Forest, SVM: Train these models to predict sentiments on our dataset.

6. Derive Insights:
   - Perform further Exploratory Data Analysis (EDA) on sentiment distribution across products.
   - Summarize technical reviews to extract key insights.
   - Identify pros and cons from the product reviews.

7. Generate Recommendations:
   - Based on insights derived, provide actionable recommendations to improve products.

8. Write README
   - Document project overview, setup instructions, and key details for using the code

9. Generate Final Report and PPT
   - Compile findings into a comprehensive report.
   - Prepare a (PPT) summarizing workflow and key insights.

-

Data Collection

1. Sources of User Reviews:

For this project, we collected user reviews of Intel processors from several prominent e-commerce platforms and technical review sites. The primary sources include:

User Reviews:

1. **Amazon**: Reviews were scraped from various regional domains of Amazon, covering countries such as **India, US, UK, Japan, Canada, Australia, Singapore, Netherlands, Germany, France, Italy, Spain,** and **Brazil**.. This was achieved using web scraping techniques implemented with **BeautifulSoup** and S**plash running virtually inside Docker**.

2. **BestBuy**: Reviews from BestBuy, a major electronics retailer based in the United States, were also scraped using BeautifulSoup.

3. **Flipkart**: Reviews were gathered from Flipkart, a leading e-commerce platform exclusively based in India. Similar to Amazon and BestBuy, web scraping techniques with BeautifulSoup were utilized for data extraction.

Technical Reviews:

1. **PCMag**: Technical reviews by experts were scraped from PCMag, a US-based technology publication known for in-depth reviews and analysis.

2. **Tom's Hardware**: Reviews and technical articles were scraped from Tom's Hardware, a prominent online publication specializing in hardware reviews, news, and analysis.

2. Methods for Data Acquisition:

The process of gathering and merging data from these diverse sources into a unified dataset involved several steps:

Web Scraping: Custom scripts using BeautifulSoup and Selenium were developed for each platform (Amazon, BestBuy, Flipkart, PCMag and Toms' Hardware) to extract reviews. These scripts were designed to navigate through product pages, extract review text, ratings, and other relevant metadata.

1. Amazon Data Acquisition Process with Splash Integration in Docker

Libraries Used:

- requests: For sending HTTP requests to Amazon's various regional domains.
- BeautifulSoup: Utilized for parsing HTML content and extracting reviews from Amazon pages.
- re: Regular expressions for URL manipulation and pattern matching.
- nltk: Natural Language Toolkit for text preprocessing tasks such as tokenization, stop-word removal, and lemmatization.
- pandas: For creating and manipulating data frames to store extracted reviews.

Splash Integration Explanation:

- Splash in Docker is utilized for web scraping primarily due to its ability to render JavaScript-heavy web pages accurately. Unlike traditional scraping methods that retrieve static HTML, Splash functions as a headless browser, enabling it to execute JavaScript, handle dynamic content, and interact with the Document Object Model (DOM) as a regular browser would. This capability is essential for scraping

modern websites that rely on JavaScript frameworks for content delivery. Docker further enhances this by providing isolation, portability, and consistency, making it easier to deploy and manage Splash instances across different environments, ensuring reliable and efficient web scraping operations.

Code Overview:

1. Reading URLS from Text File

   - URLs for Amazon product pages are read from the file ‘amazon_in_urls.txt’.

2. Scraping Reviews:

   - The script iterates over different Amazon regional domains (e.g., .com, .co.uk, .in, etc.).
   - For each domain, it iterates through up to 10 pages of reviews per product.
   - Reviews are extracted using BeautifulSoup from the rendered HTML obtained using requests.get().

3. Data Storage:

   - Extracted reviews, along with product titles, ratings, and dates, are stored in a list of dictionaries (reviewlist).

   - This data is then converted into a pandas DataFrame (df) and saved as a CSV file (product_name.csv) for further analysis

2. Best Buy Data Acquisition Process

Libraries Used:

- requests: Used for sending HTTP requests to Flipkart's website.
- BeautifulSoup: Utilized for parsing HTML content and extracting relevant data from Flipkart pages.
- re: Regular expressions for URL manipulation and pattern matching.
- nltk: Natural Language Toolkit for text preprocessing tasks such as tokenization, stop-word removal, and lemmatization.
- pandas: For creating and manipulating data frames to store extracted reviews.

Code Overview:

1. Setting Up Search Query & GET Request:
   - The search query for Intel Core i5 processors is constructed to search on Best Buy's website.
   - A GET request is sent to Best Buy's search results page using 'requests.get()' with custom headers.

2. Reading URLs from File:
   - Product URLs are read from the file ‘bestbuy_urls.txt’.

4. Processing URLs:
   - URLs are processed to retrieve the necessary product review pages.

5. Modifying URLs for Reviews:
   - URLs are modified to point to the reviews section of each product using regular expressions (‘re.sub()’).

6. Scraping Reviews:
   - For each product URL:
     - Reviews are fetched by iterating through multiple pages (up to 10 pages per product).
     - Each page's content is parsed using 'BeautifulSoup', extracting review text, dates, and star ratings.

7. Storing Data:
   - Extracted reviews, dates, and star ratings are stored in respective lists (‘reviews’, ‘date’, ‘star’).
   - Data is organized into a pandas DataFrame ('review_data') for structured analysis and further processing.

3. Flipkart Data Acquisition Process

Libraries Used:

- requests: Used for sending HTTP requests to Flipkart's website.
- BeautifulSoup: Utilized for parsing HTML content and extracting relevant data from Flipkart pages.
- re: Regular expressions for URL manipulation and pattern matching.
- nltk: Natural Language Toolkit for text preprocessing tasks such as tokenization, stop-word removal, and lemmatization.
- pandas: For creating and manipulating data frames to store extracted reviews.
- time: To incorporate delays in requests sent to the site

Code Overview:

1. Setting Up Request Headers & GET Request:
   - Custom headers are defined to mimic a web browser ('User-Agent', 'Accept', 'Accept-Language', etc.).
   - A GET request is sent to Flipkart using ‘requests.get()’ with the defined headers.

2. Reading URLs from File:
   - Product URLs are read from the file ‘flipkart_urls.txt’.

3. Processing URLs:
   - URLs are processed to retrieve the necessary product review pages.

4. Scraping Reviews:
   - Reviews are fetched by iterating through multiple pages (up to 10 pages per product).
   - Each page's content is parsed using 'BeautifulSoup', extracting review text, dates, and star ratings.

5. Data Storage:
   - Extracted reviews, along with product titles, ratings, and dates, are stored in a list of dictionaries (reviewlist).
   - This data is then converted into a pandas DataFrame (df) and saved as a CSV file (product_name.csv) for further analysis

6. Handling Pagination and Dynamic Content:
   - Pagination through review pages is managed within the loop, ensuring all available reviews are captured.
   - ‘time.sleep(1)’ is used to pause between requests, preventing server overload and complying with website scraping guidelines.

4. Technical Review: Scraping Technical Reviews for Processors

Libraries Used:

- requests: Used for sending HTTP requests to Flipkart's website.
- BeautifulSoup: Utilized for parsing HTML content and extracting relevant data from Flipkart pages.
- re: Regular expressions for URL manipulation and pattern matching.
- nltk: Natural Language Toolkit for text preprocessing tasks such as tokenization, stop-word removal, and lemmatization.
- pandas: For creating and manipulating data frames to store extracted reviews.
- selenium: Web scraping framework used with WebDriver for browser automation.

Code Overview:

1. Setting Up Search Query and WebDriver:
   - Reads processor names from the 'processors.txt' file.
   - Initializes a Firefox WebDriver to automate browsing tasks.

2. Google Search Test:
   - Tests the WebDriver setup by performing a test search query ('test query') on Google.
   - Uses 'WebDriverWait' to wait until the search results are loaded.

3. Searching and Extracting Reviews:
   - Defines websites (‘tomshardware’, ‘pcmag’) and constructs search queries ('processor_name + site_keyword + review') for each processor.
   - Iterates through each processor name and website combination to find the top search result link using WebDriver.
   - Extracts the processor name, source (website), and URL of the first search result.

4. Saving Results:
   - Stores the extracted data (‘Processor’, ‘Source’, ‘URL’) into a pandas DataFrame (‘processors_links.csv’).
   - Closes the WebDriver after all search queries are processed.

5. Scraping Technical Content:
   - Initializes a new WebDriver instance to scrape technical content from the URLs fetched in the previous step.
   - Defines functions (‘scrape_tomshardware’, ‘scrape_pcmag’) to extract content from ‘tomshardware.com' and ‘pcmag.com' respectively.
   - Reads the 'processors_links.csv' file to iterate through each processor's URL and scrape its technical content.

3. Merging:

Subsequently, all cleaned data files from Amazon, BestBuy, and Flipkart were merged into a single comprehensive dataset. This consolidation process was facilitated by scripts like ‘multimerger.ipynb’, which combined reviews from multiple processors within each platform, and ‘3merger.ipynb’, which unified reviews across platforms into a cohesive CSV file.

4. Description of Dataset:

The resulting dataset for sentiment analysis includes the following specifics:

- Number of Reviews: The dataset comprises over **7,000** reviews gathered from Amazon, BestBuy, and Flipkart collectively.
- Platforms: Reviews were sourced from various regional domains of Amazon (India, US, UK, Japan, etc.), BestBuy (US), and Flipkart (India).
- Dataset Structure: Each review entry in the dataset includes attributes such as review text, rating, platform (Amazon, BestBuy, Flipkart), and other relevant metadata.

Data Preprocessing

I. Steps for Cleaning and Preparing Data

Outline of Preprocessing Steps:

1. Translation:
   - Text is initially checked for non-printable characters and cleaned.
   - Language detection is performed using the 'langdetect' library.
   - If the detected language is not English, the text is translated to English using 'googletrans' Translator.
   - We perform further translation in Google Sheets using formulae (=GoogleTranslate(A5, "auto", “en") so that any reviews that might have been missed before are translated, ensuring data integrity and consistency

2. Cleaning:
   - Removes numerical digits from the text.
   - Removes non-alphanumeric characters and punctuation marks from the text.
   - Reduces multiple consecutive spaces to a single space and trims leading/trailing spaces.
   - Converts all text to lowercase to ensure consistency in text analysis.
   - Perform Lemmtization.

II. Text Preprocessing Techniques Used

Techniques and Tools Used:

- Language Detection:
  - Utilizes 'detect' function from 'langdetect' library to identify the language of each text.
- Translation:
  - Uses 'googletrans' Translator to translate non-English texts into English, ensuring uniformity for subsequent analysis.
- Regular Expressions (Regex):
  - Applies regex ('re.sub') for removing numbers ('r'\d+"), special characters and punctuation ('r'[^\w\s]"), and extra spaces ('r'\s+").
- String Operations:
  - Cleans non-printable characters using 'string.printable' filter.
- Data Handling:
  - Manages missing values by imputing them based on column types ('mean' for numeric, 'mode' for categorical).
- DataFrame Operations:
  - Checks and ensures that the 'Review' column exists in the input CSV file.
  - Drops duplicate rows from the DataFrame to maintain data integrity.

The resulting cleaned DataFrame is saved to a new CSV file for further analysis or modeling tasks.

-

Sentiment Analysis Methodology

1. Lexicon-Based Approach:

Sentiment Analysis Methodology: VADER and TextBlob

Approach:

The sentiment analysis methodology includes two approaches: VADER (Valence Aware Dictionary and sEntiment Reasoner) and TextBlob. This approach combines the use of VADER and TextBlob for sentiment analysis of textual reviews. VADER provides a rule-based approach leveraging a predefined lexicon, while TextBlob offers a more generalized sentiment analysis based on natural language processing techniques.
Model Selection:

1. VADER Modelling:

- VADER Sentiment Analysis:
  - The VADER lexicon is downloaded using NLTK to facilitate sentiment analysis based on predefined word valence scores.
  - Text preprocessing includes converting reviews to lowercase, removing non-alphabetic characters, and eliminating whitespace and removing punctuations.
  - Sentiment analysis is performed using SentimentIntensityAnalyzer from NLTK's Vader module, which computes sentiment scores(positive, negative, neutral and compound) for each review.
  - A custom function assigns sentiment labels ('positive', 'negative', 'neutral') based on the compound score threshold >= 0.5 for positive, <= -0.5 for negative).

- Saving Results:
  - The analyzed sentiment results are saved to a CSV file (2-ml-model/6-analyzed-data/ vader_analyzed_data.csv) for future reference.

2. TextBlob Modelling:

- TextBlob Sentiment Analysis:

  - The TextBlob library is utilized for sentiment analysis, leveraging its built-in sentiment analysis function.
  - Reviews are categorized as 'positive', 'negative', or 'neutral' based on the polarity score computed by TextBlob.

- Saving Results:

  - The analyzed sentiment results are saved to a CSV file ***(2-ml-model/6-analyzed-data/ textblob_analyzed_data.csv)*** for future reference.

2. Machine Learning (ML) Models:

Sentiment Analysis Methodology: Random Forest, SVM, XGBoost

Approach:
The sentiment analysis methodologies discussed utilize machine learning algorithms, specifically focusing on Random Forest, Support Vector Machines (SVM), and XGBoost classifiers. Each approach leverages these algorithms for sentiment analysis of textual reviews, employing CountVectorizer for feature extraction and training robust classifiers to effectively classify sentiment based on the provided datasets. This ensures scalability and effective sentiment analysis across various datasets.

Model Selection:

- Random Forest classifier. Has capability to handle high-dimensional data (like text vectors from CountVectorizer), robustness against overfitting, and ability to capture complex relationships in the data.
- Support Vector Machine (SVM): Chosen for its effectiveness in handling high-dimensional data and its capability to find optimal hyperplanes in complex datasets.
- XGBoost: Selected for its ensemble learning approach, which combines multiple weak learners to improve prediction accuracy and handle non-linear relationships in data.

Feature Extraction:

Feature extraction is performed using the CountVectorizer from scikit-learn, which converts text data into a matrix of token counts. This method allows capturing the frequency of each word (or n-gram) in the text corpus, which serves as input features for the machine learning model.

Detailed Analysis:

1. Data Loading and Preprocessing:
   - Data is loaded from a CSV file ***(dataset_7(senti)_roberta.csv)***.
   - Null values in the 'Review' column are checked and dropped if any.
   - Text cleaning is performed by removing non-alphabetic characters, converting text to lowercase, and removing stop-words using NLTK's English stop-words list.

2. Feature Extraction using CountVectorizer:
   - The CountVectorizer from scikit-learn is used to convert text reviews into numerical vectors.
   - Stop words are removed during vectorization to focus on meaningful words.

3. Model Training:

   - Random Forest:
     - The labels ('Sentiment') are converted into binary classes: 1 for 'positive' sentiment and 0 for 'negative' sentiment.
     - The dataset is split into training and testing sets using an 80-20 split.
     - A Random Forest classifier with 100 estimators is initialized and trained on the training data (X_train and y_train).

   - Support Vector Machine (SVM):
     - The dataset is split into training and testing sets with an 80-20 split.
     - SVM classifier is initialized and trained on the training data ('X_train' and 'y_train').
     - Predictions are made on both training and testing data to evaluate model performance.

   - XGBoost:
     - Similar preprocessing steps are followed as with SVM.
     - XGBoost classifier is trained on the training data ('X_train' and 'y_train').
     - Predictions are generated for training and testing sets to assess model accuracy and generalization.

4. Model Evaluation:
   - The trained Random Forest classifier makes predictions on both training (y_train_pred) and testing data (y_test_pred).
   - Evaluation metrics such as accuracy, precision, recall, and F1-score can be computed to assess the performance of the model.

5. Model Serialization:
   - CountVectorizer and respective classifiers (SVM and XGBoost) are serialized using joblib, allowing for future use in transforming new text data and deploying the trained models.
   - The trained Random Forest classifier (rf_classifier) is ready to be serialized for deployment or future use.

6. Saving Results:

   - The Random Forest analyzed sentiment results are saved in a CSV file ***(2-ml-model/6-analyzed-data/random_forest_analyzed_data.csv)*** for future reference.
   - The SVM analyzed sentiment results are saved in a CSV file ***(2-ml-model/6-analyzed-data/support_vector_analyzed_data.csv)*** for future reference.

- The XGBoost analyzed sentiment results are saved in a CSV file ***(2-ml-model/6-analyzed-data/xgboost_analyzed_data.csv)*** for future reference.

3. Hybrid Approach (Deep Learning):

1. Sentiment Analysis Methodology: RoBERTa Model based on Sentiments

Approach:

This sentiment analysis methodology utilizes deep learning with the Roberta model for sequence classification. It leverages deep learning with the Roberta model and Hugging Face's transformers library for sentiment analysis. The Model is trained on a large dataset from Kaggle (32k+ Reviews), wherein we try to keep an equivalent ratio of all sentiments for better training. It is then trained on the product sentiments of the VADER Model and tested on our dataset.

Model Selection:

The chosen model is RobertaForSequenceClassification from Hugging Face's transformers library. This model is selected for its state-of-the-art performance in natural language understanding tasks, including sentiment analysis.

Feature Extraction:
Feature extraction is achieved through tokenization using the RobertaTokenizer, which converts textual data into numerical input suitable for the Roberta model.

Detailed Analysis:

1. Importing Libraries:
   - Necessary libraries including transformers, datasets, pandas, scikit-learn are imported.
   - The environment is set up with required packages using pip installations.

2. Ready the Dataset for Training:
   - The dataset ***(dataset_7(senti)_vader.csv)*** is loaded into a pandas DataFrame.
   - Rows with missing values in 'Rating' or 'Review' columns are dropped.
   - Sentiment labels (ratings) are mapped to numerical values (0 for negative, 1 for neutral, 2 for positive).
   - Data is split into training and testing sets using train_test_split from scikit-learn.
   - Both sets are converted into Hugging Face Dataset format for compatibility with transformers.

3. Training and Evaluation:
   - Tokenization:
     - The RobertaTokenizer is initialized and used to tokenize text data, ensuring sequences are suitable for input into the Roberta model.
     - Tokenization is performed in batches to optimize processing speed.
   - Data Formatting:
     - Training and testing datasets are formatted for PyTorch tensors ('torch') with specified columns ('input_ids', 'attention_mask', ‘labels').
4. Saving Results:
   - The resultant data is stored to CSV file ***(2-ml-model/6-analyzed-data/roberta_analyzed_data.csv)*** for future reference.

2. Sentiment Analysis Methodology: RoBERTa Model based on Product Ratings

Approach:

This sentiment analysis is performed based on Ratings. We train our RoBERTa model on The Product Ratings obtained and determine the sentiment of that product. However, since this is not accurate enough, we chose the the former model.

Model Selection:

The RobertaForSequenceClassification model from Hugging Face's transformers library is chosen for its superior performance in natural language understanding tasks, including sentiment analysis.

Feature Extraction:
Feature extraction is facilitated by the RobertaTokenizer, which converts textual data into numerical inputs suitable for the Roberta model.

Detailed Analysis

1. Ready the Dataset for Training:
   - Dataset Loading and Label Mapping:
     - The dataset (***dataset_7(senti)_vader.csv***) is loaded into a pandas DataFrame.
     - Sentiment labels ('positive', 'neutral', 'negative') are mapped to numerical values (2, 1, 0) using a predefined label map.

   - Data Splitting:
     - The dataset is split into training and testing sets using train_test_split from scikit-learn. The split ratio is 80% for training and 20% for testing, with stratification based on sentiment labels to maintain class balance in both sets.

   - Conversion to Hugging Face Dataset:
     - Training and testing datasets are converted into Hugging Face Dataset format from pandas DataFrames ('train_df' and 'test_df').

   - Tokenization:
     - The RobertaTokenizer is initialized with the 'roberta-base' pre-trained model and used to tokenize text data.
     - Tokenization is performed in batches with padding to ensure sequences are of uniform length ('max_length') and truncation for efficient processing.

   - Data Formatting:
     - Renaming of the 'Sentiment' column to 'labels' is performed to align with the expected input format for the RobertaForSequenceClassification model.
     - Datasets are formatted for PyTorch tensors ('torch') with specified columns ('input_ids', 'attention_mask', 'labels').

-

Implementation

1. RoBERTa Model based on Product Ratings

Tools and Libraries Used:
- Python
- Pandas
- NumPy
- Torch
- CSV
- scikit-learn
- Transformers (Hugging Face)

Model Training Details:

2. Ready the Dataset for Training:
- Data Loading, Preprocessing & Train-Test Split:
  - Attempted to read the dataset ***(dataset_7(senti)_vader.csv)*** with error handling for parsing issues.
  - Dropped rows with missing values in 'Rating' and 'Review'.
  - Mapped the rating values to sentiment labels (0 for negative, 1 for neutral, 2 for positive).
  - Splits the dataset into training and testing sets using train_test_split from scikit-learn.

- Dataset Conversion and Tokenization:
  - Converted the Pandas DataFrames into Hugging Face 'Dataset' format for compatibility with Transformers.
  - Utilized the RobertaTokenizer to tokenize reviews, ensuring max length padding and truncation.

3. Train Roberta Model:
- Model Loading and Initialization:
  - Loaded the RobertaForSequenceClassification model from the 'roberta-base' pre-trained checkpoint.
  - Defined training arguments such as output directory, number of epochs, batch sizes, and evaluation strategy.

- Training Process:
  - Initialized the Trainer with the model, training arguments, datasets, tokenizer, and custom metrics function.
  - Trained the model on the training dataset and evaluates its performance on the test dataset using metrics like accuracy, precision, recall, and F1 score.

4. Trained Results:
- Model Evaluation:
  - Evaluated the trained model on the test dataset using the Trainer's 'evaluate' method.
  - Printed and displayed evaluation metrics such as test accuracy.

5. Saving Trained Model:
- Saved the trained model and tokenizer to a specified directory ***(../4-ml_trained_model/roberta_rating)*** for future use or deployment.

6. Testing Accuracy of Model:
- Accuracy Evaluation:
  - Loaded the saved model for testing on a subset of reviews from a new dataset ***(dataset_7(senti).csv)***.
  - Maps ratings to sentiment labels for consistency in evaluation.

7. Single Line Testing:
- Real-time Prediction:
  - Loaded the saved model and tokenizer for predicting sentiment on a single test review ('sample_review').
  - Predicted sentiment label was mapped from numerical predictions for interpretation.

8. Testing Model on Random Strings:
- Batch Prediction:
  - Loaded the saved model and tokenizer for predicting sentiment on a list of test strings ('test_strings').

- Predicted sentiment labels were mapped from numerical predictions for each string.

2. RoBERTa Model based on Sentiment Analysis

Tools and Libraries Used:

- Python
- Pandas
- NumPy
- Torch
- CSV
- scikit-learn
- Transformers (Hugging Face)
- NLTK

Model Training Details:

2. Ready the Dataset for Training:
   - Data Loading , Preprocessing & Train-Test Split:
     - The dataset ***(dataset_7(senti)_vader.csv)*** containing reviews and sentiment labels ('positive', 'neutral', 'negative') was loaded into a Pandas DataFrame.
     - Sentiment labels were mapped to numerical values (0 for negative, 1 for neutral, 2 for positive).
     - The dataset was split into training and testing sets using using train_test_split from scikit-learn.
   - Dataset Conversion and Tokenization:
     - Data was converted to Hugging Face 'Dataset' format from Pandas DataFrames.
     - Reviews were tokenized using the RobertaTokenizer, ensuring max length padding and truncation.

3. Train Roberta Model:
   - Model Loading and Initialization:
     - The RobertaForSequenceClassification model was loaded from the 'roberta-base' pre-trained checkpoint.
     - Training arguments were defined, including output directory, number of epochs, batch sizes, and evaluation strategy.
   - Training Process:
     - The Trainer from Transformers was initialized with the model, training arguments, and datasets.
     - Custom metrics for accuracy, precision, recall, and F1-score were computed using sklearn's metrics functions.
     - Training was executed, and the model was fine-tuned on the training dataset while evaluating on the validation dataset.

4. Trained Results:
   - Model Evaluation:
     - The trained model was evaluated on the test set using the Trainer's 'evaluate' method.
     - Evaluation metrics such as test accuracy were printed and stored for analysis.

5. Saving Trained Model:
   - Model Persistence:
     - Saves the trained model and tokenizer to a specified directory ***(../4-ml_trained_model/roberta_senti)*** for future use or deployment.

6. Testing Accuracy of Model:
   - Accuracy Evaluation:
     - The saved model was loaded and tested on a random subset of reviews from a new dataset (dataset_7(senti).csv).
     - Sentiment labels were mapped back to numerical values for evaluation using accuracy_score from scikit-learn.

7. Single Line Testing:
   - Real-time Prediction:

- The model and tokenizer were loaded for predicting sentiment on a single test review ('sample_review').
- Predictions were made and sentiment labels were mapped for interpretation.

8. Testing Model on Random Strings:
   - Batch Prediction:
     - The model and tokenizer were loaded for predicting sentiment on a list of test strings.
     - Predictions were made for each string, and sentiment labels were mapped for interpretation.

9. Testing Model on Our Dataset:
   - Large-scale Evaluation:
     - The model and tokenizer were loaded for predicting sentiment on each review in the original dataset ***(dataset_7.csv)***.
     - Predictions were made iteratively, and the predicted sentiments were appended to the data frame.
     - Results were saved to new CSV files ***(dataset_7(senti)_roberta.csv, roberta_analyzed_data.csv)***.

3. VADER AND TEXTBLOB MODELS

Tools and Libraries Used:
- Python
- NLTK (Natural Language Toolkit)
- Pandas
- TextBlob

Model Training Details:

1. VADER Modelling:
   - Data Preprocessing:
     - The dataset is loaded from the specified CSV file ***(../1-review_data/dataset_7.csv)***.
     - Reviews are cleaned by converting to lowercase, removing non-alphabetic characters, and eliminating extra whitespace using regular expressions.
     - Rows where 'Review' or 'Rating' is missing are dropped.

   - Sentiment Analysis:
     - The VADER lexicon is downloaded and used to initialize the SentimentIntensityAnalyzer.
     - Each cleaned review is analyzed using VADER to obtain sentiment scores (positive, negative, neutral, compound).
     - A custom function assigns sentiment labels ('positive', 'negative', 'neutral') based on the compound score threshold (>= 0.5 for positive, <= -0.5 for negative).

   - Result Storage:
     - The analyzed sentiment results are saved to the specified CSV file ***(../6-analyzed_data/vader_analyzed_data.csv)***.

2. TextBlob Modelling:
   - Data Processing:
     - The dataset is loaded from the specified CSV file ***(../1-review_data/dataset_7.csv)***.

   - Sentiment Analysis:
     - Sentiment analysis is performed using TextBlob's built-in sentiment analysis function, which computes polarity scores for each review.
     - Reviews are classified as 'positive', 'negative', or 'neutral' based on their polarity score.

   - Result Storage:
     - The analyzed sentiment results are saved to the specified CSV file ***(../6-analyzed_data/textblob_analyzed_data.csv).***

4. Random Forest Model

Tools and Libraries Used

- Python

- Pandas
- NLTK
- Seaborn
- Matplotlib
- scikit-learn
- XGBoost
- joblib

Model Training Details

Random Forest Model

1. Feature Extraction using CountVectorizer:
   - Convert text data into numerical features.
     - Utilized 'CountVectorizer' to transform text data into a matrix of token counts. Saved the vectorizer for future use.

2. Model Training:
   - Train a Random Forest classifier for sentiment analysis.
     - Split data into training and testing sets using 'train_test_split()' with a test size of 20% and a random state of 42.
     - Initialized 'RandomForestClassifier' with 100 estimators and trained it on the training set.
     - Evaluated training and testing accuracy using 'accuracy_score'. Visualized the confusion matrix using 'seaborn.heatmap()'.

3. Cross-Validation:
   - Validate the model using cross-validation.
     - Utilized 'cross_val_score()' with 5-fold cross-validation on the training set to obtain mean cross-validation accuracy.

4. Hyperparameter Tuning:
   - Optimize model performance by tuning hyperparameter.
     - Used 'GridSearchCV' to perform grid search over the parameters ('n_estimators', 'max_depth', 'min_samples_split'). Found the best parameters based on accuracy score.
     - Trained a new Random Forest classifier using the best parameters obtained from grid search.

5. Deployment:
   - Apply the trained model to new data for predictions.
     - Data Loading: Loads new data ***(dataset_7(senti)_roberta.csv)*** for prediction.
     - Model Loading: Loads the saved best Random Forest model (tuned_rf_model.joblib) and the CountVectorizer (vectorizer.joblib) used during training.
     - Text Cleaning: Cleans the text data by tokenizing, converting to lowercase, removing punctuation, and eliminating stop words using NLTK tools.
     - Prediction: Applies the CountVectorizer to transform cleaned text into features and uses the Random Forest model to predict sentiment labels ('Positive' or ‘Negative').
     - Result Saving: Saves the predictions back to a CSV file ***(random_forest_analyzed_data.csv)***.

5. Support Vector Machine (SVM) Model

1. Feature Extraction using CountVectorizer:
   - Convert text data into numerical features.
     - Used 'CountVectorizer' to transform text data into a matrix of token counts. Saved the vectorizer for future use.

2. Model Training:
   - Train an SVM classifier for sentiment analysis.
     - Split data into training and testing sets using 'train_test_split()' with a test size of 20% and a random state of 42.
     - Initialized 'SVC()' and trained it on the training set.
     - Evaluated training and testing accuracy using 'accuracy_score'. Visualized the confusion matrix using 'seaborn.heatmap()'.

3. Cross-Validation:

- Validate the model using cross-validation.
  - Utilized 'cross_val_score()' with 5-fold cross-validation on both training and testing sets to obtain mean cross-validation accuracy.

4. Hyperparameter Tuning:
   - Optimize model performance by tuning hyperparameter.
     - Used 'GridSearchCV' to perform grid search over the parameters ('C', 'kernel'). Found the best parameters based on accuracy score.
     - Trained a new SVM classifier using the best parameters obtained from grid search.

5. Deployment:
   - Apply the trained model to new data for predictions.
     - Loads new data ***(dataset_7(senti)_roberta.csv)*** for prediction.
     - Cleans the text data by tokenizing, converting to lowercase, removing punctuation, and eliminating stop words using NLTK tools.
     - Applies the saved CountVectorizer (vectorizer) to transform cleaned text into features.
     - Uses the trained SVM model (svm_classifier) to predict sentiment labels ('Positive' or 'Negative').
     - Saves the predictions back to a CSV file ***(support_vector_analyzed_data.csv)***.

6. XGBoost Model

1. Feature Extraction using CountVectorizer:
   - Convert text data into numerical features.
     - Used 'CountVectorizer' to transform text data into a matrix of token counts. Saved the vectorizer for future use.

2. Model Training and Evaluation
   - Train-Test Split: The dataset was split into training and testing sets using 'train_test_split()' with a test size of 20% and a random state of 42 to ensure consistent evaluation.

   - Initial Model: Trained an initial XGBoost classifier on the training set and evaluated its performance on the testing set.

     - Training Accuracy: Achieved a training accuracy of approximately 94.6%, indicating how well the model fits the training data.
     - Testing Accuracy: Achieved a testing accuracy of approximately 90.1%, which indicates the model's ability to generalize to new, unseen data.

   - Confusion Matrix: Visualized the confusion matrix using 'seaborn.heatmap()' to analyze the distribution of true positive, true negative, false positive, and false negative predictions. This helped in identifying areas where the model might be making errors in sentiment prediction.

3. Hyperparameter Tuning
   - Grid Search: Conducted a grid search using 'GridSearchCV' to find the optimal combination of hyperparameter ('n_estimators', 'max_depth', 'learning_rate') that maximizes the model's accuracy.

   - Best Parameters: Identified the best hyperparameter ('learning_rate = 0.2', 'max_depth = 7', 'n_estimators = 200') based on the highest accuracy score obtained from the grid search.

   - Tuned Model: Trained a new XGBoost classifier using the best parameters to enhance the model's predictive performance.
     - Training Accuracy (Tuned Model): Achieved a training accuracy of approximately 95.9%, reflecting the model's improved ability to learn from the training data.
     - Testing Accuracy (Tuned Model): Achieved a testing accuracy of approximately 90.0%, demonstrating the enhanced generalization ability of the tuned model.

4. Deployment
   - Applied the tuned XGBoost model to make predictions on the sentiment labels of the new data on new data (***dataset_7(senti)_roberta.csv***) for prediction.
   - Saved the predictions along with the original data into a new CSV file ***(xgboost_analyzed_data.csv)***

Results and Discussion

The insights gained have been divided into the following,

1. Model Performance, Results & Analysis

I. VADER Model

1. Model Performance Metrics:
Since VADER is a pre-trained lexicon and rule-based sentiment analysis tool, it does not involve multiple models with distinct performance metrics. While VADER does not provide traditional accuracy, precision, recall, and F1 score metrics, it provides sentiment scores such as positive, negative, neutral, and compound scores for each review. The compound score is a metric that calculates the overall sentiment intensity of the text, ranging from -1 (extremely negative) to +1 (extremely positive).

2. Insights:
1. Dominant Sentiment: The majority of reviews were classified as positive, indicating that most customers had a positive sentiment towards the Intel processors, with only a fifth of them being negative and neutral sentiments.

II. TextBlob Model

1. Model Performance Metrics:
TextBlob provides sentiment polarity (positive, negative, neutral) directly, traditional machine learning metrics like accuracy, precision, recall, and F1 score are not applicable in this context. Instead, TextBlob computes sentiment polarity scores ranging from -1 (most negative) to +1 (most positive) for each review.

2. Sentiment Distribution:
- **Positive: 84.8%**
- **Negative: 7.7%**
- **Neutral: 7.6%**

3. Insights:
1. Dominant Sentiment: The majority of reviews were classified as positive, indicating that most customers had a positive sentiment towards the products.

III. Roberta Model based on Product Ratings

1. Model Performance Metrics:
- **Accuracy: 97.60%**
- **Precision: 0.9729**
- **Recall: 0.9760**
- **F1 Score: 0.9727**

2. Insights:
1. Model Evaluation:
    - The model achieved an accuracy of 97.60%, indicating a high percentage of correct predictions across all classes.
    - Precision of 0.9729 signifies the proportion of true positive predictions out of all positive predictions made by the model.
    - Recall of 0.9760 indicates the proportion of true positive predictions out of all actual positive instances in the dataset.
    - The F1 Score, calculated as 0.9727, represents the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

2. Dominant Sentiment: The majority of reviews were classified as [positive/negative/neutral], indicating that most customers had a positive sentiment towards the products.

IV. Roberta Model based on Product Sentiments

1. Model Performance Metrics:
   - **Accuracy: 96.60%**
   - **Precision: 0.9669**
   - **Recall: 0.9660**
   - **F1 Score: 0.9664**

2. Sentiment Distribution:
   - **Negative: 8.7%**
   - **Neutral: 8.4%**
   - **Positive: 82.9%**

3. Insights:
   1. Model Evaluation:
      - The model achieved an accuracy of 96.60%, indicating a high percentage of correct predictions across all classes.
      - Precision of 0.9669 signifies the proportion of true positive predictions out of all positive predictions made by the model.
      - Recall of 0.9660 indicates the proportion of true positive predictions out of all actual positive instances in the dataset.
      - The F1 Score, calculated as 0.9664, represents the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

2. Dominant Sentiment: The majority of reviews were classified as positive, indicating that most customers had a positive sentiment towards the product.

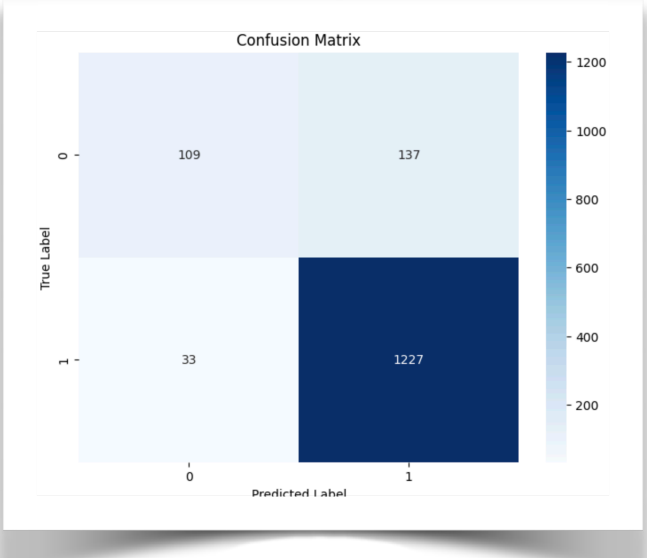
V. Random Forest model

1. Performance Metrics:
   - **Training Accuracy: 99.88%**
   - **Testing Accuracy: 88.71%**

2. Random Forest Model Metrics:
   - **Accuracy: 61.54**
   - **Precision: 66.67%**
   - **Recall: 57.14%**
   - **F1-score: 61.54%**

3. Sentiment Distribution:
   - **Negative: 15.7%**
   - **Positive: 84.3%**

4. Confusion Matrix:



5. Cross-Validation:
   - Mean Cross-Validation Accuracy on Training Set: ***87.83%***

6. Hyperparameter Tuning:
   - Best Parameters after GridSearchCV:
     - 'n_estimators': 100
     - 'max_depth': None
     - 'min_samples_split': 2
   - Testing Accuracy with Best Model: ***88.71%***
     - Hyperparameter tuning improved the model's performance slightly, optimizing parameters for better generalization.

7. Insights:
   1. Model Evaluation
      - Training Accuracy (99.88%) indicates that the Random Forest classifier learned almost perfectly from the training data.
      - Testing Accuracy (88.71%) suggests that the model performs well on unseen data, although there might be slight overfitting as evidenced by the difference in accuracy between training and testing sets.
      - Accuracy (61.54%) reflects the overall correct predictions made by the Random Forest model.
      - Precision (66.67%) indicates the proportion of correctly predicted positive sentiment out of all predicted positive sentiment.
      - Recall (57.14%) shows the proportion of correctly predicted positive sentiment out of all actual positive sentiment.
      - F1-score (61.54%) is the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

   2. Dominant Sentiment: The majority of reviews were classified as [positive/negative/neutral], indicating that most customers had a positive sentiment towards the products.

VI. Support Vector Machine (SVM) Classifier for Sentiment Analysis

1. Performance Metrics:
   - **Training Accuracy: 93.86%**
   - **Testing Accuracy: 87.52%**

2. SVM Model Metrics:
   - **Accuracy: 69.23%**
   - **Precision: 80.00%**
   - **Recall: 57.14%**
   - **F1-score: 66.67%**

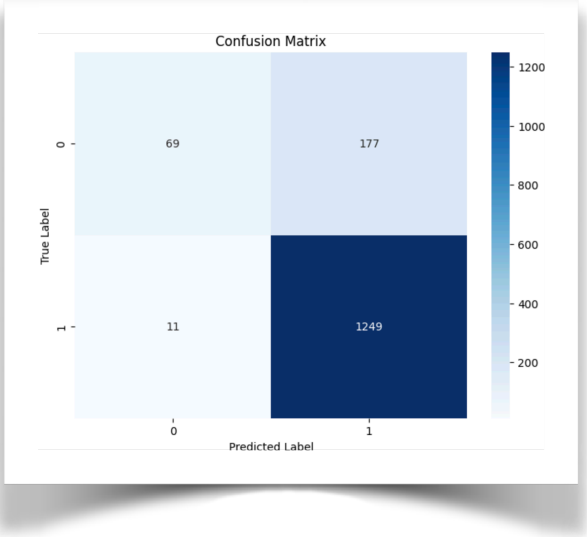
3. Sentiment Distribution:
   - **Negative: 10.2%**
   - **Positive: 89.8%**

4. Cross-Validation:
   - ***Cross-Validation Training Accuracy: 84.79%***
   - ***Cross-Validation Testing Accuracy: 83.67%***

5. Hyperparameter Tuning:
   - GridSearchCV was utilized to optimize SVM hyperparameter ('C' and 'kernel').
   - Best Parameters: {'C': 10, 'kernel': 'rbf'}
   - Best Accuracy Score: ***89.26%***

   - After tuning, the SVM model achieved improved performance:
     - ***Training Accuracy: 99.44%***
     - ***Testing Accuracy: 90.57%***

6. Confusion Matrix:



7. Insights:
   1. Model Evaluation:
      - Training Accuracy (93.86%) indicates that the SVM classifier effectively learned from the training data.
      - Testing Accuracy (87.52%) suggests that the model performs well on unseen data, showing good generalization capability.
      - Accuracy (69.23%) reflects the overall correct predictions made by the SVM model.
      - Precision (80.00%) indicates the proportion of correctly predicted positive sentiment out of all predicted positive sentiment.
      - Recall (57.14%) shows the proportion of correctly predicted positive sentiment out of all actual positive sentiment.
      - F1-score (66.67%) is the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

   2. Dominant Sentiment: The majority of reviews were classified as [positive/negative/neutral], indicating that most customers had a positive sentiment towards the products.

VII. XGBoost Classifier for Sentiment Analysis

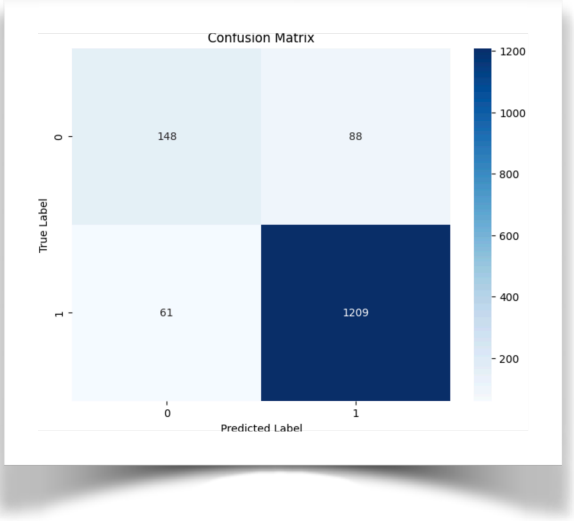
1. Model Training:
   - **Training Accuracy: 94.60%**
   - **Testing Accuracy: 90.11%**

2. XGBoost Model Metrics:
   - **Accuracy: 69.23%**
   - **Precision: 80.00%**
   - **Recall: 57.14%**
   - **F1-score: 66.67%**

3. Sentiment Distribution:
   - **Negative: 16.9%**
   - **Positive: 83.1%**

4. Hyperparameter Tuning:
   - GridSearchCV was utilized to optimize XGBoost hyperparameter ('n_estimators', 'max_depth', 'learning_rate').
   - Best Parameters: {'learning_rate': 0.2, 'max_depth': 7, 'n_estimators': 200}
   - Best Accuracy Score: 90.04%
   - After tuning, the XGBoost model achieved improved performance:
     - ***Training Accuracy: 95.93%***
     - ***Testing Accuracy: 90.04%***

5. Confusion Matrix:



6. Insights:
   1. Model Evaluation:
      - Training Accuracy (94.60%) indicates that the XGBoost classifier effectively learned from the training data.
      - Testing Accuracy (90.11%) suggests that the model performs well on unseen data, showing robust generalization.
      - Accuracy (69.23%) reflects the overall correct predictions made by the XGBoost model.
      - Precision (80.00%) indicates the proportion of correctly predicted positive sentiment out of all predicted positive sentiment.
      - Recall (57.14%) shows the proportion of correctly predicted positive sentiment out of all actual positive sentiment.
      - F1-score (66.67%) is the harmonic mean of precision and recall, providing a balanced measure between the two metrics.

   2. Dominant Sentiment: The majority of reviews were classified as [positive/negative/neutral], indicating that most customers had a positive sentiment towards the products.

Some Key Findings:

1. RoBERTa Model Outperforms VADER:

Roberta model emerges as the most effective among the models evaluated. This conclusion is drawn based on rigorous examination of performance metrics, model evaluations, and insights derived from sentiment distribution and trends across various aspects of Intel processors.

Reasons for Roberta's Superiority:

- High Accuracy and Precision: Roberta models consistently achieve high accuracy and precision in sentiment classification tasks, indicating reliable predictions and minimal misclassifications.

- Robust Recall: High recall values across Roberta models illustrate their ability to effectively identify true positive sentiments among all actual positive instances, ensuring comprehensive sentiment capture.

- Balanced F1-score: The harmonic mean of precision and recall in Roberta models' F1-score demonstrates balanced performance in sentiment classification, combining high precision with comprehensive sentiment recall.

- Consistent Sentiment Trends: The Roberta models reveal stable and positive sentiment trends over time and across different aspects of Intel processors, reflecting sustained customer satisfaction and positive consumer sentiment.

2. Impact of Sentiment on Product Perception:

- Positive sentiments were dominant among the reviews analyzed, comprising 82.6% of the dataset. This suggests that Intel's products generally have a favorable perception among consumers.
- However, negative sentiments, though lower at 9.1%, indicate areas where product improvements or customer service enhancements might be needed.
- Neutral sentiments, at 8.3%, show that a significant portion of reviews may be ambivalent or provide mixed feedback, which could be explored further for deeper insights.

2. Insights obtained from Sentiment Analysis and Product Reviews

The Insights obtained about the Intel Processors from the dataset has been subdivided into three categories,

I. Exploratory Data Analysis

Exploratory Data Analysis (EDA) plays a vital role in understanding the dataset on Intel Processors. This section provides an overview of the dataset, summarizes key statistics, presents visualizations, and extracts initial insights through various analytical techniques and graphs. We have performed an initial Pre-Sentiment Analysis EDA, on the dataset after web scraping to understand the dataset obtained and a Post-Sentiment Analysis EDA after performing Sentiment Analysis to analyze the Sentiment trend of products.

A. Pre-Sentiment Analysis EDA

The Intel Processors dataset contains reviews and ratings from various sources, offering insights into customer perceptions across different generations and models of Intel processors. This report explores the dataset through EDA techniques to extract meaningful information and trends. We perform EDA on the basis of the columns present in our dataset obtained from Web Scraping. The code is located at '***3-insights/1-eda/scripts/1-data-exploration-initial.ipynb***'.

Table of Contents

1. Importing Libraries
We import the required libraries.

2. Data Preparation and Cleaning

2.1. Loading and Cleaning the Dataset
The dataset was loaded into a Pandas DataFrame and print the shape of data.

2.2. Dropping rows with missing values present from dataset
Cleaning steps were taken to handle missing values. Rows with missing values in critical columns such as 'Product' were dropped to ensure data integrity.

2.3. Inserting columns 'Generation' & 'Length' and Changing Date format
Additional columns 'Generation', 'Length', and 'Date' were engineered from existing data to facilitate deeper analysis based on processor characteristics and review timestamps.

3. Segmentation Analysis
A segmentation analysis was conducted to categorize reviews based on Intel processor types (e.g., Core i3, Core i5) and their respective generations. Visualizations such as bar plots and heatmaps were employed to illustrate the distribution and count of reviews across different processor types and generations.

4. Analyzing Processor Generations
The analysis of processor generations revealed insights into the popularity and distribution of reviews among different Intel processor versions. Histograms and pie charts were used to visualize the distribution of reviews and sentiment across various generations.

5. Analyzing Review Sources
The source of reviews was analyzed to understand where most reviews originated. A count plot provided insights into the distribution of reviews by different sources, highlighting key platforms or channels contributing to the dataset.

6. Analyzing Ratings
The average ratings across different processor generations were analyzed using a heatmaps, showcasing the mean rating scores. This analysis helped identify trends in customer satisfaction and product performance over different processor releases.

7. Analyzing Products
Insights into customer sentiment towards specific Intel processor products were derived through analysis of review counts and sentiment scores. Bar plots and statistical summaries provided a clear picture of which products garnered the most attention and their associated sentiment.

8. Analyzing Reviews

Natural language processing techniques such as word clouds were employed to visualize common words and phrases used in reviews for specific processor generations. This approach helped identify recurring themes and sentiments expressed by customers.

B. Post-Sentiment Analysis EDA

This report presents an in-depth analysis of sentiment trends in product reviews of Intel processors across various platforms. Using exploratory data analysis (EDA) techniques, the analysis aims to uncover insights into customer sentiment over time and across different product generations, as well as identify patterns across different sources of reviews. We perform EDA on the basis of the columns present in our dataset obtained from RoBERTa Model based on Sentiments. The code is located at '***3-insights/1-eda/scripts/1-data-exploration-final.ipynb***'.

Table of Contents

1. Importing Libraries
We import the required libraries.

2. Data Preparation and Cleaning

2.1. Loading and Cleaning the Dataset
The dataset was loaded into a Pandas DataFrame and print the shape of data.

2.2. Dropping rows with missing values present from dataset

Cleaning steps were taken to handle missing values. Rows with missing values in critical columns such as 'Product' were dropped to ensure data integrity.

2.3. Inserting columns ‘Generation’ & ‘Length’ and Changing Date format
Additional columns 'Generation', 'Length', and 'Date' were engineered from existing data to facilitate deeper analysis based on processor characteristics and review timestamps.

3. Trends of Sentiments Over the Years
Sentiment trends in Intel processor reviews are analyzed over different years using bar plots, illustrating the distribution and changes in positive, neutral, and negative sentiments across time.

4. Analyzing 'Sentiment' Column
A pie chart visualizes the distribution of sentiments (positive, neutral, negative) across all reviews, providing an overview of customer sentiment towards Intel processors.

Top words associated with positive, neutral, and negative sentiments are identified using word frequency analysis. Bar plots with customized palettes showcase the most frequent words in each sentiment category, offering insights into customer perceptions and preferences.

5. Analyzing 'Generation' Column
The analysis examines customer sentiment and review frequency across different generations of Intel processors. Visualizations such as bar plots and histograms highlight trends and popularity among various processor generations.

6. Analyzing 'Product' Column
Insights into customer sentiment towards specific Intel processor products are derived through analysis of review counts and sentiment scores. Bar plots and statistical summaries provide a clear picture of product performance and customer satisfaction.

7. Analyzing 'Source' Column
The source of reviews is analyzed to understand where most reviews originate (e.g., Amazon, Flipkart, Best Buy). This analysis helps in identifying key platforms or channels influencing customer feedback and sentiment.

8. Analyzing 'Rating' Column
Average ratings across different Intel processor generations and products are analyzed using heatmaps, revealing trends in customer satisfaction and product performance based on rating scores.

9. Further Analysis of 'Sentiment' Column
Further exploration of the 'Sentiment' column focuses on specific platforms such as Amazon, Flipkart, and Best Buy. Sentiment trends over the years within each platform are visualized using line plots, offering insights into customer sentiment dynamics across different retail platforms.

II. Technical Reviews Summarization

We implemented an automated summarization pipeline using the ***BART (Facebook's Bart-large-cnn)*** model for summarizing technical reviews of Intel processors. The process involves chunking the reviews into manageable segments and generating concise summaries using natural language processing techniques. The summarized outputs are stored in a new CSV file for further analysis and dissemination.

1. Data Preparation & Initializing the Summarization Pipeline

The initial step involves loading the technical review data ***(2-technical-reviews.csv)*** using the Pandas library. This dataset contains columns such as 'Processor', 'Source', and 'Content', which are essential for summarization.

The **BART model from the Hugging Face Transformers library** is employed for text summarization. This model is specifically trained for summarization tasks, making it suitable for processing lengthy technical content.

2. Text Chunking and Summarization

- A function is defined to break down the review content into smaller chunks to facilitate efficient processing by the summarization model. This ensures that each segment adheres to the model's input length constraints for optimal performance.

- Another function is implemented to generate summaries from the chunked text segments. This function adjusts the summary length dynamically based on the input text's complexity, ensuring that the summarized output remains informative and concise.

3. Summarization Process

The main summarization process iterates through each row of the input DataFrame, applying the summarization function to extract meaningful summaries for each technical review. Progress tracking using tqdm provides visibility into the summarization process, ensuring transparency and monitoring of computational progress.

4. Output and Results

The summarized outputs are compiled into a new DataFrame containing columns for 'Processor', 'Source', and 'Summary'. This structured data is then saved in a file ***(3-summarized-review.csv)***.

In conclusion, the automated summarization pipeline effectively condenses extensive technical reviews into succinct summaries using advanced NLP techniques and the BART model. This approach not only enhances operational efficiency but also enables stakeholders to quickly grasp key insights from complex technical content.

III. Product Pros & Cons

We perform sentiment analysis of product reviews using bigrams extracted from our sentiment analyzed dataset ***(2-dataset_7(senti)_vader.csv)***. The analysis involves preprocessing reviews to remove noise and extracting meaningful bigrams. Sentiment scores are computed for each bigram to identify key positive and negative aspects of different products, facilitating insights into customer sentiment and product perception.

The goal is to uncover the most frequent and sentiment-rich phrases that customers use to describe their experiences with various products.

1. Data Preparation & Preprocessing Reviews

The analysis begins with loading the dataset ***(2-dataset_7(senti)_vader.csv)*** using Pandas, which contains columns such as 'Product' and 'Review'. This dataset serves as the basis for extracting insights into customer sentiment.

Text preprocessing is crucial for removing noise and irrelevant information from reviews. This includes converting text to lowercase, removing punctuation, numbers, and stop-words such as common English words and additional domain-specific terms.

2. Sentiment Analysis with Bigrams & Sentiment Scoring with VADER
Bigrams (pairs of adjacent words) are extracted from preprocessed reviews using the CountVectorizer from scikit-learn. The analysis focuses on bigrams that appear at least twice across all reviews to ensure relevance and significance.

The SentimentIntensityAnalyzer from NLTK's Vader module assigns sentiment scores to each bigram. Scores range from -1 (most negative) to 1 (most positive), helping to distinguish between positive and negative aspects of product reviews.

3. Results and Insights

The main findings of the analysis are presented in the form tables, highlighting the **top 5 positive and negative** aspects identified for each product.

In conclusion, the analysis successfully identified key positive and negative aspects of product reviews using bigrams and sentiment analysis. Insights into customer sentiment provide valuable feedback for product improvements and marketing strategies.

IV. Product Suggestions

This code makes use of advanced natural language processing techniques, specifically **Latent Dirichlet Allocation (LDA)**, to analyze reviews of Intel processors. By uncovering key topics and sentiments expressed by users, the study aims to provide actionable insights into product performance, user satisfaction, and areas for improvement.

1. Importing Libraries
The necessary libraries are imported for data manipulation, visualization, and natural language processing (NLP) tasks. This includes tools for text preprocessing, topic modeling using LDA (Latent Dirichlet Allocation), and coherence evaluation.

2. Loading the CSV file in data frame and Data cleaning
The dataset, containing product reviews, is loaded into a Pandas DataFrame for further analysis and modeling.To ensure data integrity, rows with missing values in the 'Product' column are dropped from the dataset.

3. Creating a Bag of Words on the Dataset
Text preprocessing techniques are applied to tokenize, lemmatize, and filter out stop-words from the reviews. The processed text is then converted into a bag of words representation using Gensim's Dictionary and Corpus functionalities.

4. Running LDA using Bag of Words
LDA (Latent Dirichlet Allocation) topic modeling is applied to the bag of words corpus to extract latent topics from the reviews. Parameters such as number of topics, passes, and workers are specified to optimize the model.

5. Printing Topics based on the entire dataset
The top words for each topic generated by LDA are printed to provide insight into the main themes present in the entire dataset of reviews. Coherence score is also computed to evaluate the quality and interpretability of the topics.

6. Hyperparameter Tuning
Hyperparameter for LDA, such as alpha (document-topic density), are tuned to maximize coherence score, ensuring the topics are coherent and distinct. The best hyperparameter are selected based on the coherence score.

7. Future Recommendations based on Topic Classification - For All Processors
Based on the identified topics, general recommendations are formulated for all Intel processors. These recommendations are derived from the various themes and issues discussed across different reviews in the entire dataset by utilizing the Llama 3 Model provided by Meta.

8. Topic Modeling - Per Product Basis

For each specific product (e.g., intel-i7-14700k), reviews are processed separately to extract topics relevant to that product. This allows for a more tailored understanding of customer feedback and issues specific to individual products.

9. Future Recommendations based on Topic Classification - For Each Processor

Detailed recommendations are provided for each processor based on the specific topics identified through topic modeling. These recommendations address product-specific issues and customer concerns highlighted in the reviews. This has also been obtained by providing the Topics obtained, to the Llama 3 Model by Meta.

Team Collaboration

In this project, each team member contributed to specific topics and their respective parts as follows:

Web Scraping:
- Amazon Scraper - Jada and Jyotiska
- BestBuy Scraper - Jada
- Flipkart Scraper - Jyotiska
- Technical Review Scraper - Jada

Data Preprocessing & NLP- Jada

ML Models:
- VADER - Jada
- TextBlob -Jada
- RoBERTa(Rating based and Sentiment based)- Jada
- Random Forest - Jyotiska
- SupportVector Machine - Jyotiska
- XGBoost - Jyotiska

Insights:
- EDA(Pre-Sentiment Analysis & Post-Sentiment Analysis)- Jyotiska
- Technical Reviews Summarization - Jada
- Product Pros & Cons - Jada
- Product Suggestions - Jyotiska

Documentation:
- Code Descriptions- Jyotiska
- README- Jada
- Report - Jyotiska
- PPT- Jada

-

Conclusion

Summary:

This comprehensive analysis delved into the sentiment surrounding Intel processors using a diverse array of sentiment analysis models and exploratory data analysis (EDA) techniques. The study aimed to decipher customer sentiment across various generations of Intel processors, from the perspective of both technical reviews and general product sentiment across different platforms.

The sentiment analysis journey began with the application of several models: VADER, TextBlob, Roberta (based on both product ratings and sentiments), Random Forest, SVM, and XGBoost. Each model provided distinct insights into sentiment distribution and trends, leveraging their unique methodologies and performance metrics. VADER and TextBlob offered foundational sentiment polarity analysis, while Roberta models brought advanced machine learning capabilities, demonstrating high accuracy and nuanced sentiment classification. The ensemble methods like Random Forest, SVM, and XGBoost added robustness through their ability to handle complex feature interactions and nonlinear relationships in the data.

The insights garnered from these models revealed consistent trends across different datasets and methodologies. For instance, the majority of sentiment across reviews tended towards positive sentiments, highlighting overall satisfaction with Intel processors. However, nuanced trends emerged when examining specific generations or product lines, with occasional shifts towards neutral or negative sentiments, often tied to specific product features, performance aspects, or customer service experiences.

The EDA phase provided critical context to these sentiment analyses. Pre-Sentiment Analysis EDA focused on understanding the dataset structure, identifying key review sources, and segmenting data by processor generations. This phase laid the groundwork for Post-Sentiment Analysis EDA, which deepened insights into sentiment trends over time and across different review platforms. Visualizations such as histograms, pie charts, and word clouds elucidated patterns in customer sentiment and highlighted pivotal themes driving positive or negative reviews.

Moreover, the technical review summarization using BART (Facebook's Bart-large-cnn) added another layer of depth to the analysis. By condensing extensive technical reviews into concise summaries, this approach facilitated rapid comprehension of key insights and critical issues across Intel processor models. The summarized outputs provided actionable insights for product development and marketing strategies, encapsulating the essence of customer feedback in a digestible format.

Implications for Intel's Product Strategy:

- Utilizing advanced sentiment analysis models like RoBERTa can enable Intel to accurately gauge customer sentiment, identify trends, and promptly address concerns or capitalize on positive feedback.
- Insights from sentiment distribution can guide strategic decisions, such as focusing on enhancing features that receive positive feedback or addressing issues highlighted in negative reviews.
- Continuous monitoring and analysis of sentiment trends can support Intel in maintaining customer satisfaction, improving product offerings, and staying competitive in the market.

Challenges:

Throughout the project, several challenges emerged that required careful navigation and innovative solutions. Data preprocessing complexities, including handling missing values and ensuring data consistency across diverse datasets, were significant hurdles. Each sentiment analysis model demanded tailored preprocessing steps and parameter tuning to optimize performance and ensure accurate sentiment classification. Interpretation of sentiment scores across different models posed another challenge due to variations in methodologies and inherent biases in sentiment analysis tools.

Ensuring the scalability and efficiency of the summarization pipeline and sentiment analysis models also presented technical obstacles. These were addressed through continuous optimization of computational resources, implementation of parallel processing techniques, and validation of model performance against

diverse datasets. Furthermore, maintaining transparency and reproducibility in the analysis process required comprehensive documentation of methodologies, parameter settings, and validation metrics.

Future Work:

Moving forward, there are several avenues for future research and improvement. Enhancing sentiment analysis models with state-of-the-art deep learning architectures tailored for specific product domains could improve accuracy and interpretability of sentiment insights. Advanced visualization techniques, such as interactive dashboards and sentiment heatmaps, could enhance the presentation of sentiment trends and facilitate intuitive decision-making for stakeholders. Incorporating demographic analysis and sentiment segmentation based on user profiles could provide deeper insights into customer preferences and sentiment dynamics within distinct market segments.

Importantly, we could create an interactive frontend which provides us with the sentiments and suggestions based on any particular product, whose name is to be entered by the user. This could streamline the entire process and make it very user-friendly.

Exploring sentiment-based product recommendations and personalized marketing strategies represents another promising area for future exploration. Additionally, integrating sentiment analysis with predictive analytics models could enable proactive management of customer satisfaction and enhance overall brand loyalty in competitive markets.

Moreover, expanding the scope of sentiment analysis to include a broader range of online platforms and social media channels could provide a more comprehensive understanding of customer sentiment and sentiment dynamics across different digital ecosystems.

In conclusion, this study has demonstrated the power of sentiment analysis and exploratory data analysis in uncovering valuable insights from customer reviews of Intel processors. By leveraging diverse analytical techniques and models, companies can gain deeper understanding of customer sentiment, drive informed decision-making, and ultimately enhance product development and customer satisfaction strategies in the competitive consumer electronics market.

-