BIG DATA ANALYTICS PROJECT
Master in Data Science and Advanced Analytics

NOVA Information Management School
Universidade Nova de Lisboa

Modelling Amazon Tech Reviews for Consumer Insight

Analysis of Amazon Electronics and Computers reviews (2022–2023) using Sentiment and Topic Modelling, Clustering, and Graph Analysis to uncover Consumer Purchasing Behaviour.

***AI** generated image* (Adobe Express, 2025)

Group 37

André Silvestre, 20240502

Filipa Pereira, 20240509

João Henriques, 20240499

Umeima Mahomed, 20240543

Fall/Spring Semester 2024-2025

TABLE OF CONTENTS

ABSTRACT

This project investigates consumer behaviour by analysing 2022-23 *Amazon Electronics – Computers* reviews using *PySpark*. Transformer models (*XLM-RoBERTa*, *mDeBERTa*) performed sentiment analysis, revealing textual nuances sometimes differing from star ratings. Topic classification identified key product themes. *K-Means* clustering segmented reviews for high-value products, while *GraphFrames* with *PageRank* and *Label Propagation* uncovered influential products and user communities. These findings support product optimization, vendor improvements, and more informed e-commerce strategies for business decision-making.

KEYWORDS: Big Data Platforms, Spark, Amazon, Sentiment Analysis, Topic Modelling, Clustering, Graph Analysis

1. INTRODUCTION

The exponential growth of social media and e-commerce activity has led to unprecedented levels of data generation. As reported by *DOMO's Data Never Sleeps 12th* report [1], 5.52 billion people (67.5% of the global population) are now online. This scale of interaction highlights the increasing importance of Big Data platforms that can efficiently acquire, process, and analyse massive and varied data streams to support decision-making.

In this project[1], we explored consumer purchasing behaviour by analysing user reviews from Amazon's Electronics - Computers category between Jan22 and Sep23. We applied sentiment and topic analysis, clustering and graph techniques using *PySpark* as our core API to access *Spark* components (*Spark SQL*, *MLlib*, *Streaming*) and *GraphFrames* package enabling distributed processing of structured, semi and unstructured data.

Although the dataset already contained user-provided ratings (from 1 to 5), we applied two Transformer-based models to independently analyse the reviews' written content and classify their sentiment as positive, neutral, or negative. By comparing the outputs of both models in a classification matrix, we created a combined sentiment score, which we then mapped back to a 5-level sentiment scale. This allowed us to verify the consistency between users' written opinions and the ratings they assigned. This task was supported by existing literature (**Table A1**), which validates the robustness of transformer models for sentiment and topic analysis tasks. Additionally, we conducted clustering analysis to identify distinct user and product segments based on review characteristics. Furthermore, user-product interaction structures were investigated via graph analysis using *PageRank* and *Label Propagation* algorithms to detect influential nodes and communities. Recent studies confirm the value of these techniques in e-commerce review networks [9]. Finally, a streaming simulation was developed to demonstrate the system's capability to process and analyse review data in near real-time.

The remainder of this project is organized following the CRISP-DM methodology, detailed in **Annex A**. A visual representation of our project workflow is provided in **Figure B1**, and the task distribution over time is illustrated in **Figure B2**. **Section 2** covers the *Data Collection, Exploration, and Preprocessing* stages. **Section 3** details the *Modelling* process, which includes *Sentiment*, *Topic*, *Clustering* and *Graph Analysis*, and a *Streaming Simulation*. Finally, **Section 4** presents our conclusions and suggests opportunities for future improvements.

2. DATA COLLECTION, EXPLORATION AND PREPROCESSING

2.1. Data Collection

The initial dataset, sourced from the *Amazon Reviews 2023* collection [12], contains over $571M$ reviews gathered between May96 and Sep23, across 33 product categories. It reflects Big Data characteristics as defined by Gartner's framework [13]: it is high-volume (hundreds of millions of reviews), high-velocity (collected continuously), and high-variety (text, images, videos, etc.). To ensure data relevance and address computational

[1] This work was developed in *Python* using *PySpark* in *Databricks Community Edition*, except for the *Transformer's* classification, due to reasons outlined **Section 3.1.**. For code visualization, we recommend consulting the notebooks in HTML format, as they provide an accessible and well-formatted view of the code, results, and visualizations, facilitating easier interpretation and sharing.

constraints, we applied a filtering process based on two criteria. First, the timeline was restricted to 2022–2023 to ensure recency and better alignment with current consumer behaviour. Second, we selected the *Electronics - Computers* category for its business relevance and product diversity. After filtering, we obtained two JSON files (one for reviews and another for product metadata), resulting in $\approx 100k$ unique products and over $1.3M$ user reviews. The files were initially in document-oriented format, containing nested structures. Using *Spark*, we converted the JSON files into tabular *Spark DataFrames*. This transformation simplified our pipeline by enabling structured access to key fields such as ratings, titles, categories, and prices, while still preserving complex fields like images and videos. The resulting variables' meaning can be found in **Table C1**. The two *DataFrames* were joined using the *parent_asin* field, which serves as a shared identifier for grouped product variants, ensuring accurate linkage between reviews and product attributes.

Due to the limitations of *Databricks Community Edition* (DCE), which only allows temporary storage under the */driver/* path, data would be lost after each session. As a workaround, we used *Google Colab* in parallel to save outputs to *Google Drive*. Each notebook was rerun in *Colab*, and its resulting outputs were saved, zipped, and reused downstream, combining Databricks for distributed processing with *Colab* for reliable modular storage.

2.2. Initial Data Preprocessing

Following data collection and filtering, deduplication was applied at the product level by retaining a single entry per product id, while keeping all review entries. No row-level duplicates were found in either *DataFrame*. In the reviews dataset, we identified only two fields contained missing values. In contrast, the product metadata had six fields with significant missing value count (**Figure C1**). Notably, variables *meta_bought_together* and *images* showed over 90% missing values and were therefore dropped. To standardize missing value handling, we used *Spark SQL* functions to convert empty strings and arrays to *NULL*. Additionally, timestamps were converted from *LongType* to *DateTime* to enable date filtering and feature extraction, while ratings were converted from *DoubleType* to *IntegerType*, aligning with their natural format (1–5) and optimizing memory usage.

2.3. Exploratory Data Analysis

This section presents the individual analysis applied to features for both products and reviews. Our goal was to assess the distribution and business relevance of each variable, indicating the required preprocessing actions (**Table C2**). Throughout this process, we used *Spark SQL* for aggregations and filtering. From an initial set of 14 product attributes and 10 review fields, we retained only 6 and 12 respectively. Variables were dropped either due to high number of missing values (e.g. *meta_images*) or lack of relevance for our analysis (e.g. *videos*).

Among products, *meta_title* underwent preprocessing to remove empty, invalid, and short titles. *meta_average_rating* showed a left-skewed distribution with the mean average rating at 4.15 and 4.5 as the most frequent average rating, highlighting that most products maintain a positive reputation among consumers (**Figure C2**). As for *meta_price*, around 40% of products lacked valid price data and were excluded from the analysis. For the remaining products, price showed a right-skewed distribution, with an average around \$164, suggesting a healthy mix of affordable peripherals and higher-end computing equipment (**Figure C2**). On the reviews side, rating showed a skewed distribution with 62.1% of reviews being 5 stars, suggesting user tendency toward positive feedback (**Figure C3**). To resolve inconsistencies where the review text was empty, but the title wasn't (and vice versa), we created the *review_text* variable, merging both. The *verified_purchase* field was used to exclude nearly 90k unverified reviews, improving feedback authenticity.

An analysis of review frequency over time (**Figure C4**) showed a peak in both review count and average rating in Jan23, following *Christmas*. Given the increase in purchases and potential strain on vendors during the holiday season, a drop in average rating could have been expected. Additionally, a steep decline after Jan23 in the number of reviews due to Amazon's enhanced review moderation [14] and slower E-Commerce growth [15]. Finally, we narrowed down our dataset to $\approx 59k$ products and $\approx 1M$ reviews after all filtering stages, including: dismissing unverified reviews, removing products without prices, and ensuring all text fields were valid.

2.4. Natural Language Processing (NLP)

As our goal was to perform topic classification based on each product's title and extract sentiment from reviews, textual analysis became essential to identify relevant vocabulary and inform further preprocessing. Textual data from *meta_title* and *review_text* underwent an NLP preprocessing pipeline (**Annex B**) using *PySpark MLlib* and *SQL*. This involved lowercasing, HTML tag and punctuation removal, tokenization with *Tokenizer*, and filtering via *StopWordsRemover*. N-gram analysis was performed to identify common word sequences.

Post-processing, *meta_title* displayed a left-skewed distribution (**Figure C5**), with most titles containing 15 to 30 words. Unigrams such as *'gb'*, *'laptop'*, *'usb'*, and *'case'* highlighted the dominance of laptops and related peripherals, suggesting that vendors emphasize memory capacity and connectivity as key selling points. The frequent appearance of terms like *'case'* in product titles also indicates that protective accessories are often marketed as essential complements (**Figure C6**). In the reviews, *review_text* showed a right-skewed distribution, with 50% of entries containing fewer than 28 words (**Figure C7**). Frequent tokens such as *'great'*, *'good'*, and *'works'*, highlighting overall customer satisfaction and suggesting strong product-market alignment for top-reviewed items. (**Figure C8**). Bigram analysis was attempted, but inconsistent spacing led to isolated words. We addressed this by refining the previous preprocessing steps to include additional spacing combinations removal.

3. MODELLING & ANALYSIS

3.1. Sentiment & Topic Analysis

Although the dataset already contained user-provided ratings, a deeper understanding of customer perception and the nuances behind these scores necessitated analysing the sentiment expressed directly in the review texts. This approach aimed to verify consistency between users' written opinions and the ratings they assigned, providing a more objective measure of satisfaction. For this purpose, we employed two pre-trained Transformer models [18][19] from the Hugging Face library [20] – *XLM-RoBERTa* [21] and *mDeBERTa* [22] – (**Annex C**) for a 3-class sentiment classification (Negative, Neutral, Positive) on review texts. Model inference was conducted in *Google Colab* due to the absence of GPU support in the DCE, which lacks the computational resources required for efficient processing of large-scale. Both models were used without additional fine-tuning and are capable of handling multilingual and nuanced language, such as sarcasm.

Both models demonstrated strong individual performance, as evidenced by their respective confidence score distributions (**Figure D1** and **Figure D2**). However, despite this individual proficiency, their predictions were not entirely congruent, exhibiting discrepancies. Notably, the *TXRBSF* model showed a greater tendency to classify reviews as Neutral (34.94%) compared to the *mDeBERTa* model (0.59%) (**Figure D3** and **Figure D4**).

To refine this and verify consistency between users' written opinions and ratings, we combined their outputs into a 5-class sentiment scheme: Negative (**N**), Negative Tendency (**NT**), Neutral (**Neu**), Positive Tendency (**PT**), and Positive (**P**) (**Figure D5**). If users are consistent (meaning their written sentiment matches their ratings) higher values will appear along the diagonal of the classification matrix, indicating strong alignment. For instance, while comparing the prediction and the actual rating, 1-star ratings mostly reflect **N** sentiment (61.9%) and 5-star ratings **P** sentiment (69.9%) (**Table D1**). Notably, 3-star ratings show a more diverse sentiment split, with significant portions labelled as **NT** (31.8%), **PT** (27.1%), and **N** (27.1%). This granular distinction is highly beneficial for the business, as it uncovers specific customer aspects hidden within ambiguous mid-tier ratings, providing clearer direction for product improvements.

Overall, the sentiment captured from text tends to lean more negative than the star ratings suggest. This is corroborated by the temporal analysis (**Figure D6**), where the average predicted sentiment consistently sits below the average true rating, regardless of the time. While cross analysing sentiment with price, we found it not discriminatory of the reviews' sentiment (**Table D2**). This could suggest that customer satisfaction is influenced more by product quality, features, and user experience rather than just its price point, indicating that both budget and premium products need to deliver on expectations to garner **P**/**PT** sentiment.

Topic classification was conducted, over the *product_title,* using a zero-shot *mDeBERTa* [22] model with 12 predefined categories adapted from *Amazon's* taxonomy. Some categories were merged to avoid overlap and ensure clearer classification which allowed us to identify recurring themes within reviews. The most frequent categories were *"Drives and Storage"* (21.84%) and *"Laptops"* (20.04%) (**Figure D7**). Topic analysis revealed that *'Laptops'* and *'Software'* tended to attract a slightly higher proportion of **N**/**NT** sentiment compared to other categories, whereas *'Desktops'* were generally viewed as **P**/**PT** (**Table D3**). Based on this, *Amazon* should consider reducing promotion of products with poor reviews and limiting their recommendations to customers to improve overall satisfaction. Among the top 10 stores, *SCREENARAMA* stood out with 63.2% **P** sentiment, while *Generic* had 20.4% **N** feedback, highlighting user preference for more recognizable brands (**Table D4**).

3.2. Clustering Analysis

Our clustering analysis focused on reviews for products priced above $\$250$ (**Table D2**), representing a key business segment ($100{,}476$ reviews across $9{,}313$ products) manageable within DCE' processing capabilities. These higher-priced items are more likely to involve detailed customer feedback and represent valuable opportunities for insight into premium product perception.

For the clustering process, which requires numerical inputs, the categorical column related to combined sentiment was transformed using *PySpark's StringIndexer*. Subsequently, relevant features (*rating*, *helpful_vote*, *combined_sentiment*, *n_reviews_text*, *meta_price*, *n_product_title*, *meta_rating_number*) were consolidated into a feature vector using *VectorAssembler*. This step is essential because *PySpark MLlib* algorithms expect input features to be presented as a single vector column. These features were then standardized with *StandardScaler* to ensure equal weighting in distance calculations. These steps were encapsulated in a *PySpark Pipeline*, ensuring a reproducible and efficient workflow suitable for large-scale data processing. Although *Principal Component Analysis* (**PCA**) was initially considered (**Annex D**), with rules of thumb suggesting 4 PC as optimal (**Figure E1**), the small dimensionality reduction (from 7 to 4 features) offered little benefit and risked losing interpretability, especially since high-dimensionality was not an issue, leading us to abandon PCA and retain the original features.

We selected the **K-Means** algorithm, implemented in *PySpark MLlib*, for its simplicity, scalability, and suitability for numerical data (**Annex E**). To determine the optimal number of clusters (k), we performed Silhouette analysis for k ranging from 2 to 10 (**Figure E2**). While $k = 6$ showed the highest average silhouette score (0.53), $k = 3$ (0.34) showed more consistent silhouette coefficients across its clusters, indicating better-defined and more cohesive groups with less noise. The inertia plot (**Figure E3**), using the elbow method, suggested an optimal k at 5. However, examining the cluster distribution for $k = 5$ (**Figure E4**) revealed highly imbalanced clusters (e.g. **Cluster 1** with only 1.13% of reviews), suggesting it might capture outliers rather than distinct segments. In contrast, $k = 3$ provided a more balanced and interpretable distribution (**Figure E4**), which combined with its ability to minimize the overall costs for the business (including computational expenses and strategy development), made it a more practical and insightful choice for our analysis.

The resulting three clusters were profiled based on their characteristics (**Table E2** and **Figure E6**). **Cluster 0** primarily contains reviews with high positive sentiment for products with relatively lower prices and a high number of ratings. **Cluster 1**, the largest segment, also shows **P** sentiment but for products that are generally higher priced and have fewer, but still substantial, ratings. **Cluster 2** is marked by **N** reviews, longer feedback, and slightly higher influence. For detailed insights and recommended marketing approaches, refer to **Table E2**.

3.3. Graph Analysis

Graph analysis offers *Amazon* a strategic advantage by mapping user-product interactions, revealing insights into popularity, influence, and preferences, while identifying customer communities. We constructed a bipartite graph using *GraphFrames*, an extension of *GraphX* leveraging *DataFrames* over *RDDs* for enhanced compatibility with DCE *Spark* ecosystem (details in **Annex F**). Vertices represent users and products priced above $250 (**Table D2**). Edges map reviews from *user* (source) to *products* (destination), meaning users have zero in-degree and products zero out-degree, reflecting the network's directional flow.

Degree distribution analysis, visualized through log-log histograms (**Figure F1**), revealed long-tailed patterns. Products' in-degrees average 10.8 (max 1,644), while users' out-degrees average 1.05 (max 24). This indicates that a few products receive many reviews, while most receive few. Notably, high in-degree products correlate with higher average classified sentiment (**Figure F2**), signaling strong popularity that can guide inventory and promotional focus. High out-degree users are prolific reviewers, ideal candidates for engagement initiatives.

PageRank identified influential products by their high in-degree. The top products, including a monitor and a gaming PC, had higher scores indicating a higher number of reviews. This reflects substantial customer engagement and visibility (**Figure F3**). Amazon can leverage this by prioritizing these items in marketing and stock. Community detection via **Label Propagation** identified 17,804 communities. The structure is fragmented, with 66.1% of communities being single-node, though the largest communities contained over 1,600 nodes (**Table F1** and **Figure F4**). Analysis of the top 10 communities showed consistently high average sentiment, generally above 4 (**Figure F5**), indicating positive experiences. Visualizing these top communities and their interconnections with products (**Figure F6**) revealed distinct user clusters. Larger communities suggest shared preferences, offering opportunities for product bundling, while smaller ones highlight niche markets.

3.4. Streaming Simulation

To demonstrate that the system can do near real-time sentiment analysis, two streaming tests were created. The first one used a simple loop to process fixed chunks of review data one by one. This helped check if the sentiment analysis worked correctly with small batches in DCE. The second, more advanced test used *Spark Streaming* component [29] to simulate a live data stream. It continuously monitored a folder in DBFS for new CSV files with reviews. When a file arrived, *Spark* read it as a micro-batch and ran the sentiment analysis using the *TXRBSF* classifier. The results were saved in a live in-memory *Spark SQL* table. This table updates over time (**Figure G1**), like in real-world systems that feed dashboards. Users can test the system by adding new CSV files (ensuring compliance with defined structure) to the folder and then running a *SQL* query to see updated results. There is a short delay due to the time *Spark* needs to detect files, schedule jobs, and run the model.

4. CONCLUSION

This project successfully analysed *Amazon* reviews from the *Electronics - Computers* category, covering Jan22 to Sep23, to derive actionable business insights. By applying sentiment and topic analysis using Transformer-based models, we uncovered nuanced customer perceptions, revealing discrepancies between user ratings and textual sentiment, with reviews often leaning more negative than ratings suggest. Topic classification highlighted key themes, guiding targeted product improvements. Clustering analysis identified three distinct user segments for products priced above $250, each with unique characteristics: positive sentiment for lower-priced items in *Mainstream Mentions* Cluster, positive but higher-priced products in *Premium Perspectives* Cluster, and negative, longer feedback in *Critical Reviews* Cluster, enabling tailored marketing strategies. Graph analysis, mapped user-product interactions, identifying influential products and cohesive communities with high sentiment, offering opportunities for inventory prioritization and product bundling. The streaming simulation demonstrated near real-time sentiment analysis capabilities using micro batch processing, proving the system's scalability for live data processing. Collectively, these findings support product optimization, vendor-specific enhancements, and data-driven decision-making.

Computational constraints, such as lack of GPU availability and lack of memory persistency, slowed processing of large-scale data. Additionally, clustering omitted some variables due to time and complexity limits which restricted deeper insights from additional features. Future work could include the fine-tuning of Transformer models and the inclusion of additional variables for the clustering profiling.

Overall, this project corroborates the value of platforms like *Databricks* for Big Data, efficiently handling complex analyses and delivering real-time insights to drive business decisions.

BIBLIOGRAPHICAL REFERENCES

[1] DOMO. (2024). *Data Never Sleeps 12.0 | Data Never Sleeps* [Infographic]. https://web-assets.domo.com/blog/wp-content/uploads/2024/12/Data-never-sleep-12.0.png

[2] Pramod Singh. (2019). *Machine Learning with PySpark : With Natural Language Processing and Recommender Systems*. Springerlink (Online Service) - Apress.

[3] Shah, M., Hazarika, Akaash Vishal, Malhotra, M., Patil, S. C., & Mohanty, J. (2025). *Bridging Emotions and Architecture: Sentiment Analysis in Modern Distributed Systems*. ArXiv.org. https://arxiv.org/abs/2503.18260

[4] Nkhata, G., Anjum, U., & Zhan, J. (2025). *Sentiment Analysis of Movie Reviews Using BERT*. ArXiv.org. https://arxiv.org/abs/2502.18841

[5] Bhargava, N., Radaideh, M. I., Hwang, K. O., Verma, A., & Radaideh, Majdi I. (2025). *On the Impact of Language Nuances on Sentiment Analysis with Large Language Models: Paraphrasing, Sarcasm, and Emojis*. ArXiv.org. https://arxiv.org/abs/2504.05603

[6] Ali, H., Hashmi, E., Yayilgan Yildirim, S., & Shaikh, S. (2024). Analyzing Amazon Products Sentiment: A Comparative Study of Machine and Deep Learning, and Transformer-Based Techniques. *Electronics*, *13*(7), 1305. https://doi.org/10.3390/electronics13071305

[7] Reddy Basani, M. (2024). *Analyzing Consumer Sentiment Using Big Data from E-Commerce Sites*. https://www.irjet.net/archives/V11/i10/IRJET-V11I10107.pdf

[8] Soni, J. (2024). Sentiment Analysis of Amazon Reviews using NLTK Vader and Robert. *International Journal of Scientific Research in Engineering and Management*, *08*(06), 1–5. https://doi.org/10.55041/ijsrem35700

[9] McGarry, K. (2023). Analyzing Social Media Data Using Sentiment Mining and Bigram Analysis for the Recommendation of YouTube Videos. *Information*, *14*(7), 408. https://doi.org/10.3390/info14070408

[10] Provost, F., & Fawcett, T. (2013). *Data Science for Business: What You Need to Know About Data Mining and Data-Analytic Thinking*. O'Reilly.

[11] Martinez-Plumed, F., Contreras-Ochando, L., Ferri, C., Hernandez Orallo, J., Kull, M., Lachiche, N., Ramirez Quintana, M. J., & Flach, P. A. (2021). CRISP-DM Twenty Years Later: From Data Mining Processes to Data Science Trajectories. IEEE Transactions on Knowledge and Data Engineering, 33(8).

[12] Hou, Yupeng, et al. "Bridging Language and Items for Retrieval and Recommendation." *ArXiv.org*, 2024, arxiv.org/abs/2403.03952?

[13] Gartner. (n.d.). *Big Data*. Gartner. https://www.gartner.com/en/information-technology/glossary/big-data

[14] PYMNTS. (2024, December 11). *Amazon Questions Influencers in Crackdown on Paid Reviews*. PYMNTS.com. https://www.pymnts.com/amazon-commerce/2024/amazon-questions-influencers-crackdown-paid-reviews/

[15] Pulse, M. (2023, December 13). *Year in Review 2023*. Marketplace Pulse. https://www.marketplacepulse.com/year-in-review-2023

[16] Jurafsky, D., & Martin, J. (2025). *Speech and Language Processing An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition with Language Models Third Edition draft*. https://web.stanford.edu/~jurafsky/slp3/ed3book_Jan25.pdf

[17] Alammar, J., & Maarten Grootendorst. (2024). *Hands-On Large Language Models*. O'Reilly Media.

[18] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017, June 12). *Attention Is All You Need*. ArXiv. https://arxiv.org/abs/1706.03762

[19] Merritt, R. (2022, March 25). *What Is a Transformer Model?* NVIDIA Blog. https://blogs.nvidia.com/blog/what-is-a-transformer-model/

[20] Hugging Face. (2024). *Hugging Face – On a mission to solve NLP, one commit at a time.* Huggingface.co. https://huggingface.co/

[21] *citizenlab/twitter-xlm-roberta-base-sentiment-finetunned · Hugging Face*. (2024, January 18). Huggingface.co. https://huggingface.co/citizenlab/twitter-xlm-roberta-base-sentiment-finetunned

[22] *MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7 · Hugging Face*. (2024, January 5). Huggingface.co. https://huggingface.co/MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7

[23] Maćkiewicz, A., & Ratajczak, W. (1993). Principal Components Analysis (PCA). *Computers & Geosciences*, 19(3), 303–342. https://doi.org/10.1016/0098-3004(93)90090-r

[24] Lloyd, S. (1982). Least squares quantization in PCM. *IEEE Transactions on Information Theory*, 28(2), 129–137. https://doi.org/10.1109/tit.1982.1056489

[25] Jain, A. K., Murty, M. N., & Flynn, P. J. (1999). Data clustering: a review. *ACM Computing Surveys*, 31(3), 264–323. https://doi.org/10.1145/331499.331504

[26] Berry, M. J. A., & Linoff, G. (2004). *Data Mining Techniques: For Marketing, Sales, and Customer Relationship Management.* Wiley Pub.

[27] Page, L., Brin, S., Motwani, R., & Winograd, T. (1999). *The PageRank Citation Ranking: Bringing Order to the Web.* Stanford InfoLab.

[28] Raghavan, U. N., Albert, R., & Kumara, S. (2007). *Near linear time algorithm to detect community structures in large-scale networks*. Physical Review E, 76(3), 036106.

[29] Apache Spark. (2014). *Spark Streaming - Spark 2.4.4 Documentation*. Apache.org. https://spark.apache.org/docs/latest/streaming-programming-guide.html

APPENDIX A. LITERATURE REVIEW

Table A1– Literature Review.
(Chronologically ordered from the most recent article to the oldest)

Paper Title	Abstract Summary	Methodology	Main Findings	Uses Spark?	Algorithms Used	Reference
Bridging Emotions and Architecture: Sentiment Analysis in Modern Distributed Systems	Analyses how sentiment analysis and distributed systems can work together for big data. Compares running models on one PC vs. a cluster for speed and accuracy.	Compared sentiment model training on single-node and 4-node distributed systems using *PySpark*. Applied *NLTK* preprocessing and BERT embeddings. Distributed pipeline parallelized preprocessing and embedding generation on 1.6M tweets.	The distributed setup (*Spark + BERT + LR*) outperformed single-node processing with $\approx 88\%$ accuracy. It significantly improved speed and scalability for sentiment analysis workflows.	Yes	*Apache Spark* (*MLlib*), *NLTK* (for preprocessing), *BERT* (for embeddings), Logistic Regression	**[3]**
Sentiment Analysis of Movie Reviews Using BERT	Proposes fine-tuning *BERT* with *BiLSTM* (*BERT+BiLSTM-SA*) for movie review sentiment analysis. Also presents a rule-based method (heuristic) to calculate the overall positive/negative sentiment for a movie from its reviews.	Fine-tuned *BERT* (*bert-base-uncased*) with a *BiLSTM* layer for binary sentiment classification. Initial BERT layers were frozen. Standard text preprocessing (lowercasing, tokenization) was used. Adam optimizer and sparse categorical cross-entropy were applied. A rule-based heuristic calculated overall movie sentiment.	The *BERT+BiLSTM-SA* model achieved state-of-the-art accuracy (up to 98.76%) across multiple datasets. Ablation studies confirmed improved generalization from combining BERT with BiLSTM.	No	*BERT* (*bert-base-uncased*), *BiLSTM*, Heuristic Algorithm.	**[4]**
On the Impact of Language Nuances on Sentiment Analysis with Large Language Models: Paraphrasing, Sarcasm, and Emojis	Explores how textual nuances (emojis, sarcasm) and data quality (fragmented language) affect sentiment analysis by LLMs, focusing on improvement via paraphrasing and sarcasm removal.	Fine-tuned LLMs on both domain-specific and general tweet datasets. Investigated the effects of sarcasm, emojis, and paraphrasing on model performance. Used manually labelled sarcastic tweet datasets and tested adversarial text augmentation. Evaluation with Accuracy, Precision, Recall, and F1-score.	Fine-tuned LLMs performed poorly on sarcasm (30%), but accuracy rose to 85% with adversarial augmentation. Sarcasm removal and general datasets also improved results. Paraphrasing helped fragmented inputs, while emojis had minimal impact.	Not Specified	LLMs (*albert-base-v2*, *DeBERTa*, *BERT*, *Falcon*, *Llama2/3*, *Mistral*, *GPT3.5*), Adversarial Text Augmentation, Sarcasm Removal, Emoji-to-Text Conversion, Sentiment Labelling Ensemble.	**[5]**

Paper Title	Abstract Summary	Methodology	Main Findings	Uses Spark?	Algorithms Used	Reference
Analysing Amazon Products Sentiment: A Comparative Study of Machine and Deep Learning, and Transformer-Based Techniques	The paper compares machine learning, deep learning, and transformer-based techniques for sentiment analysis of Amazon product reviews, finding that BERT outperforms other methods.	Used a dataset of 400K Amazon reviews. Applied preprocessing (removing stopwords, punctuation, tokenization). Converted text using BoW and TF-IDF. Trained ML (Naive Bayes, Logistic Regression, Random Forest), DL (CNN, *BiLSTM*), and transformers (*BERT*, *XLNet*). Evaluated with accuracy, precision, recall, and F1-score.	Among multiple NLP models, *BERT* emerged as the most accurate (89%) for Amazon reviews. Study highlighted strengths of transformer-based methods over traditional ML/DL.	No	Bag-of-Words (BOW), TF-IDF, Multinomial Naïve, Bayes (MNB), Random Forest (RF), Logistic Regression (LR), CNN, *BiLSTM* *BERT*, *XLNet*	**[6]**
Analysing Consumer Sentiment Using Big Data from E-Commerce Sites	Evaluates BERT, CNN, and LSTM on a large multilingual dataset (2.5M reviews) using big data technologies for processing.	Analysed 2.5M reviews in Turkish, Arabic, and English using Apache Spark and Hadoop. Trained *BERT*, CNN, and LSTM models using *TensorFlow* and *PyTorch*. Focused on scalable processing and multilingual sentiment classification.	*BERT* outperformed CNN and LSTM across languages with 95% accuracy. Demonstrated effectiveness of *Spark*/*Hadoop* in supporting large-scale multilingual sentiment analysis.	Yes	*BERT*, CNN, LSTM	**[7]**
Sentiment Analysis of Amazon Reviews using NLTK Vader and Robert	This paper compares the performance of *VADER* and *RoBERTa* for sentiment analysis of Amazon reviews.	Pre-processed Amazon reviews using Python libraries and NLTK for POS tagging. Extracted features and applied ML models for polarity classification. Compared rule-based *VADER* with transformer-based *RoBERTa*, examining semantic performance differences.	*RoBERTa* provided higher accuracy and better generalization than VADER. Findings emphasized *RoBERTa's* strength in deeper semantic understanding, though VADER was more efficient computationally.	Not Specified	NLTK, VADER, *RoBERTa*, POS tagging	**[8]**
Combining Sentiment Analysis and Graph Theory for YouTube Recommendations (Conceptual Example)	Proposes combining sentiment from social media posts with graph theory to provide YouTube video recommendations on controversial topics like climate change.	Combined sentiment analysis from user interactions (posts, likes/dislikes) with user-content relationships modelled via GNNs (using iGraph). Applied Non-negative Matrix Factorization (NMF) to generate YouTube video recommendations for polarizing topics like climate change.	Successfully combined sentiment analysis and graph data for YouTube recommendations but faced challenges in cross-forum user linkage. Showed promise in enhancing content suggestions for controversial topics.	Not Specified	Sentiment Analysis, Graph Theory	**[9]**

APPENDIX B. PROJECT SCHEMA

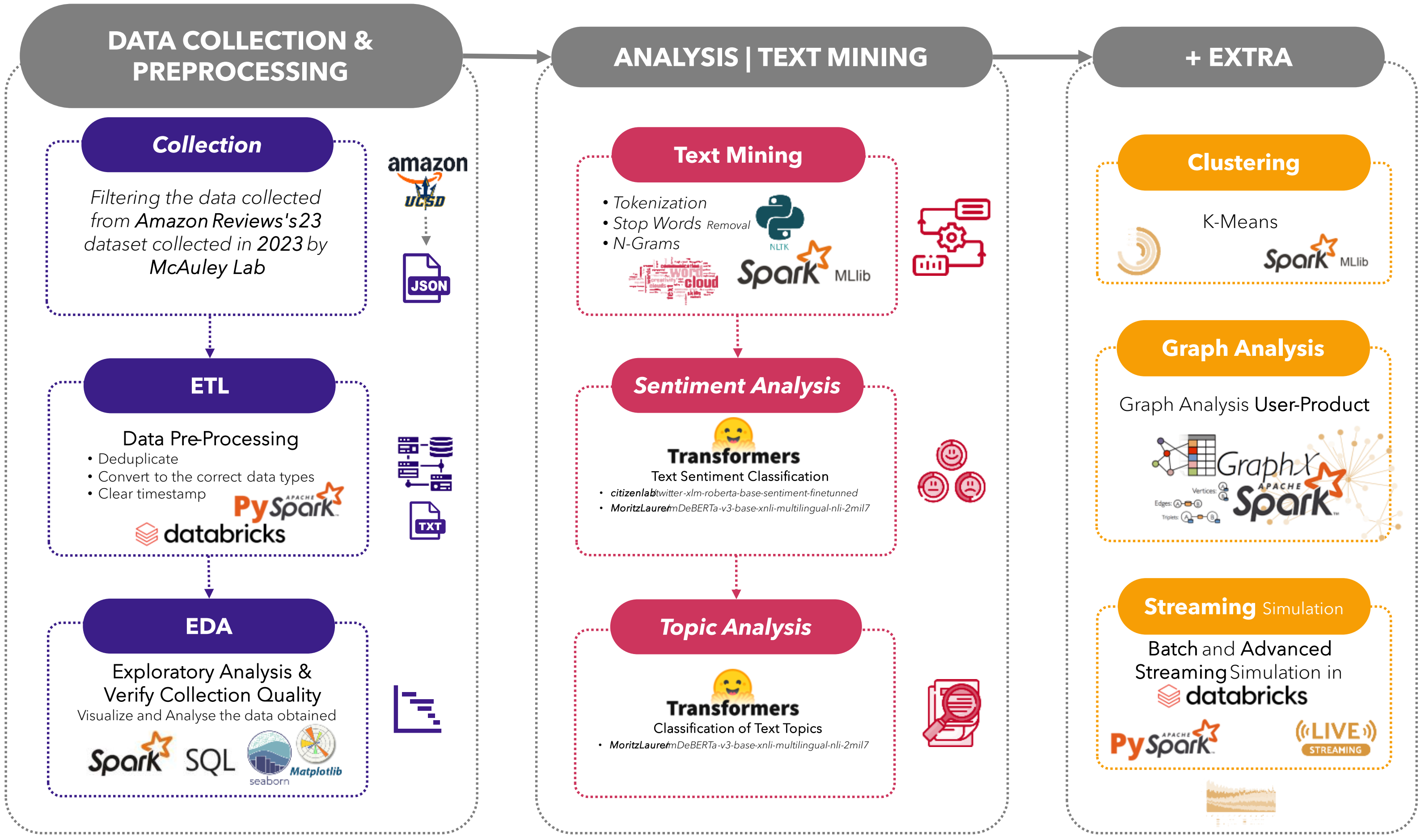

DATA COLLECTION & PREPROCESSING
Collection
Filtering the data collected from Amazon Reviews's23 dataset collected in 2023 by McAuley Lab
amazon
UCSD
JSON
ETL
Data Pre-Processing
• Deduplicate
• Convert to the correct data types
• Clear timestamp
PySpark
databricks
TXT
EDA
Exploratory Analysis & Verify Collection Quality
Visualize and Analyse the data obtained
Spark SQL
seaborn
Matplotlib
ANALYSIS | TEXT MINING
Text Mining
• Tokenization
• Stop Words Removal
• N-Grams
NLTK
word cloud
Spark MLlib
Sentiment Analysis
Transformers
Text Sentiment Classification
• citizenlab/twitter-xlm-roberta-base-sentiment-finetunned
• MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7
Topic Analysis
Transformers
Classification of Text Topics
• MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7
+ EXTRA
Clustering
K-Means
Spark MLlib
Graph Analysis
Graph Analysis User-Product
GraphX
Apache Spark
Streaming Simulation
Batch and Advanced Streaming Simulation in databricks
PySpark
LIVE STREAMING

Figure B1 – Project Schema.

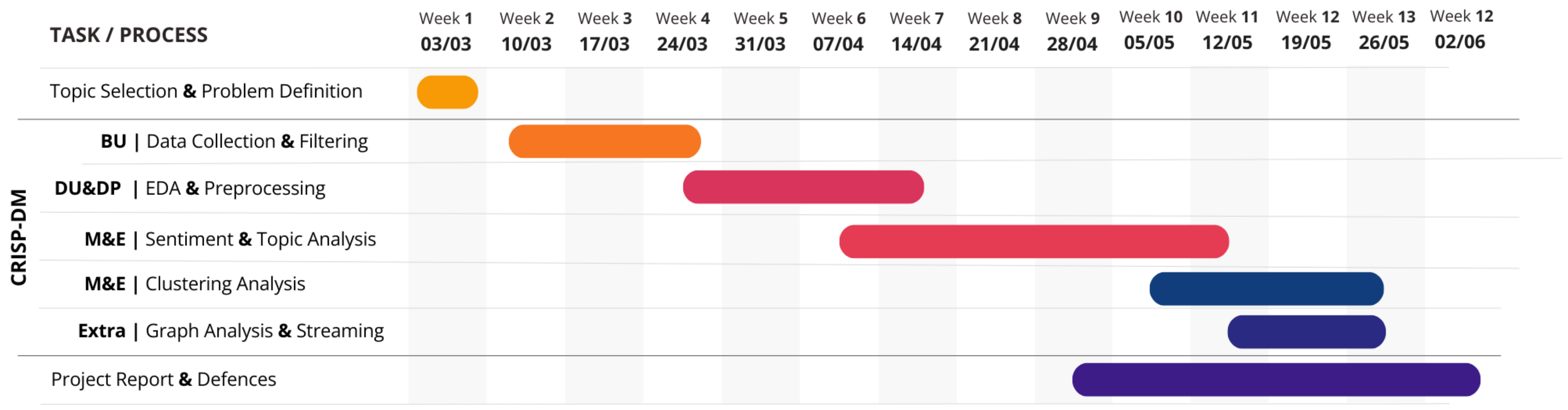


Figure B2 – Gantt Chart with Task Distribution.

APPENDIX C. DATA EXPLORATION & PREPROCESSING

Table C1 - Dataset Attributes with Descriptions.
Source: [12]

		Field	Explanation	Type
User Reviews	**1**	*rating*	Rating of the product (from 1.0 to 5.0).	*float*
	2	*title*	Title of the user review.	*str*
	3	*text*	Text body of the user review.	*str*
	4	*images*	Images that users post after they have received the product. Each image has different sizes (small, medium, large), represented by the *small_image_url*, *medium_image_url*, and *large_image_url* respectively.	*list*
	5	*asin*	ID of the product.	*str*
	6	*parent_asin*	Parent ID of the product. **Note:** Products with different colors, styles, sizes usually belong to the same parent ID. The *asin* in previous Amazon datasets is actually parent ID. Please use parent ID to find product meta.	*str*
	7	*user_id*	ID of the reviewer.	*str*
	8	*timestamp*	Time of the review (unix time).	*int*
	9	*verified_purchase*	User purchase verification.	*bool*
	10	*helpful_vote*	Helpful votes of the review.	*int*
Item Metadata	**1**	*main_category*	Main category (i.e., domain) of the product.	*str*
	2	*title*	Name of the product.	*str*
	3	*average_rating*	Rating of the product shown on the product page.	*float*
	4	*rating_number*	Number of ratings in the product.	*int*
	5	*features*	Bullet-point format features of the product.	*list*
	6	*description*	Description of the product.	*list*
	7	*price*	Price in US dollars (at time of crawling).	*float*
	8	*images*	Images of the product. Each image has different sizes (*thumb*, *large*, *hi_res*). The “variant” field shows the position of image.	*list*
	9	*videos*	Videos of the product including title and url.	*list*
	10	*store*	Store name of the product.	*str*
	11	*categories*	Hierarchical categories of the product.	*list*
	12	*details*	Product details, including materials, brand, sizes, etc.	*dict*
	13	*parent_asin*	Parent ID of the product.	*str*
	14	*bought_together*	Recommended bundles from the websites.	*list*

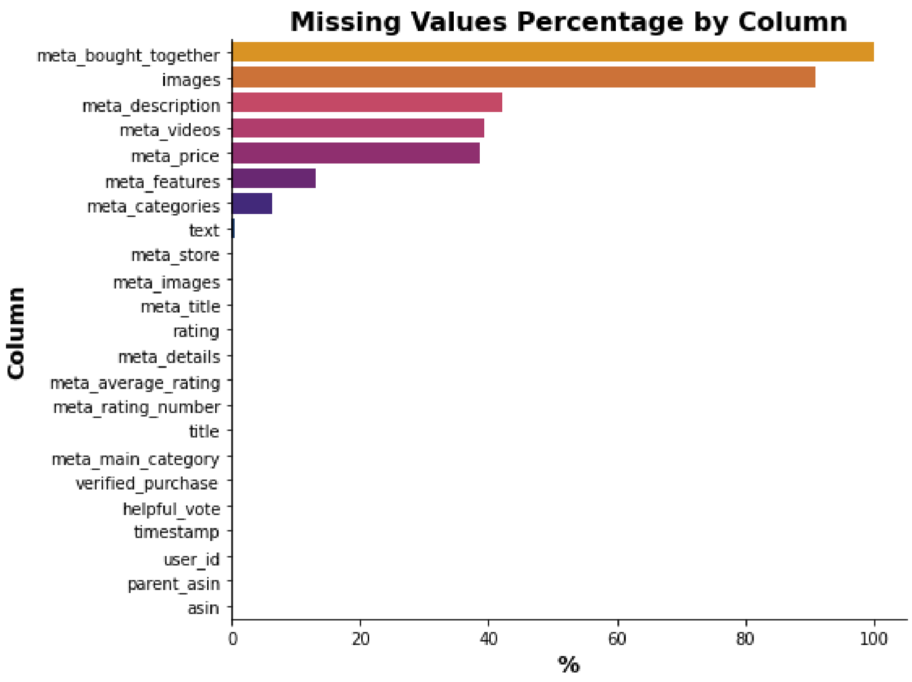


Figure C1 – Barplot of Missing Values by Column in the Product Metadata and Reviews Dataset.

We performed an *Exploratory Data Analysis* (EDA) (**Table C2**) where all insights and values presented are derived from the analysis of each variable, as detailed in the plots and tables found in the notebook *1_BU&EDA_BDAProject_Group37_Databricks.ipynb*.

Table C2 - Initial Preprocessing Analysis/Decisions.

	Variable	Exploratory Data Analysis (EDA) Summary	Preprocessing Steps & Rationale
Products/Metadata	***main_category***	Confirmed all products belong to the *"Computers"* subdomain. Review volume was high in 2022/early 2023, declining mid-2023.	**Removed** variable since it only has 1 value. We are only using only the main category *"Computers"* inside *"Electronics"* so doesn't make sense to keep the variable
	title	Identified 785 reused titles. Filtered out empty, invalid (single char, null), or very short ($<$ 3 chars) titles. A small fraction (0.09%) with suspicious character patterns was retained after review.	**Removed** products and respective reviews where the product title was empty (single char, no char or defined as null) and removed products with strange titles (less than 3 chars)
	average_rating	No missing values. Left-skewed distribution; mean 4.15, mode 4.5. Top-rated products (**5.0**) and lowest-rated (**1.0**) inspected.	No preprocessing steps applied
	rating_number	Long-tailed distribution; most products had few ratings, some $>$ 1,000. Reflects product visibility and popularity.	No preprocessing steps applied
	features	13% missing. Valid entries averaged 4.96 bullet points. High uniqueness.	No additional steps were taken since this variable will **not be used** for further analysis since it has too many unique values hence not being useful for analysis
	description	42% missing or invalid (placeholders, $<$ 5 chars). Most valid descriptions $<$ 100 words.	Due to the high number of missing values, the variable will be **removed**
	price	$\approx$ 40% missing (*NULL* values). Valid prices averaged \$167.38 (range \$1–\$14,999), highly right-skewed. Most reviews with missing price came from dropped products.	The 40k products without price were filtered from further analysis. As a very important variable, to enable meaningful analysis we filtered out all products with unknown price, focusing on the remaining products with valid prices.
	images	99.96% of products include images. Indicates image data is standard for listings.	Since this project is not focused on image analysis, no preprocessing steps were taken to address the missing values and the variable will be **dropped**
	videos	60.6% of products had videos. Presence not consistent across all listings.	Due to the high number of products without videos associated and the emphasis on our project not being on video analysis, we will **drop** this feature

	store	15,433 unique sellers. Amazon Renewed (6.4%) is most dominant. Brands like HP, Lenovo, and ASUS also frequent.	No preprocessing steps applied
	categories	Most products fell into computing categories like laptops or accessories. 6.2% had empty category fields.	Since we don't require it for further analysis and we will proceed with our own categorization, the variable will be dropped
	details	Descriptions are typically long and well populated. 63% were 301– 600 characters. However, there are 99k unique values.	Due to high number of unique values (high cardinality) and lack of usefulness considering our project scope, the variable will be dropped
	parent_asin	Parent identifier for grouped product variations. Used for joining with reviews.	No preprocessing steps applied
	bought_together	100% of entries are null. Completely missing from dataset.	Due to the variable being 100% missing, it will be **dropped**
Reviews	***rating***	62.1% of reviews rated **5** stars. Average rating is 3.97. Indicates generally positive bias in user feedback.	variable was converted from DoubleType to IntegerType in the previous section
	title	$800k+$ unique titles. Most between 11– 30 characters. Some empty reviews had only content in title.	During the analysis of the title and text variables in the reviews, we initially identified a small portion of reviews (0.39%) where the text was empty. Upon closer inspection, we discovered that many of these "empty text" reviews actually had meaningful content stored in the title field instead. To avoid losing this valuable information, we created a new variable called ***review_text*** that combines the title and text fields.
	text	Only 0.39% empty. High variability. Most reviews fall between 51 – 250 words.	
	images	91% of reviews lacked images. Image attachment not relevant for our analytical goals.	Due to the high percentage of missing values and the lack of relevance for our analysis, the images column will be **dropped**
	asin	Unique identifier for specific product variant. Redundant with *parent_asin* for join purposes.	**Dropped** the variable as it was redundant for our product-level analysis and not used for joins
	parent_asin	Shared identifier to join reviews to products. Crucial for linking metadata.	No preprocessing steps were taken. Used as the primary key for joining the reviews and metadata *DataFrames*
	user_id	Identifies each reviewer. No quality concerns identified.	No preprocessing steps were taken
	timestamp	71.4% of reviews in 2022. Review activity peaked in January and during evenings. Used for time-based insights.	In **Section 2** we had already converted the datatype from *LongType* to *DateTime*

verified_purchase	93.2% of reviews were verified purchases. Unverified reviews (6.8%) may be less reliable.	The 89,405 reviews without verified purchases were filtered from further analysis. This decision was made to prioritize feedback with more credibility to ensure a robust analysis
helpful_vote	77.8% of reviews received **0** helpful votes. Only 0.7% received more than **10** votes.	No Preprocessing steps

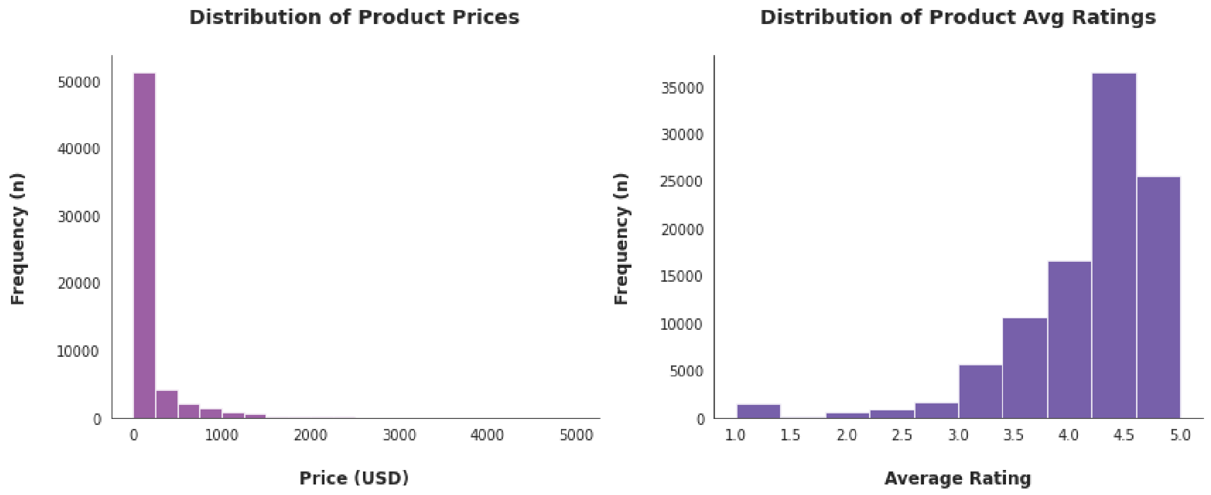


Figure C2 – Histogram with distribution of Product Prices and Average Ratings.

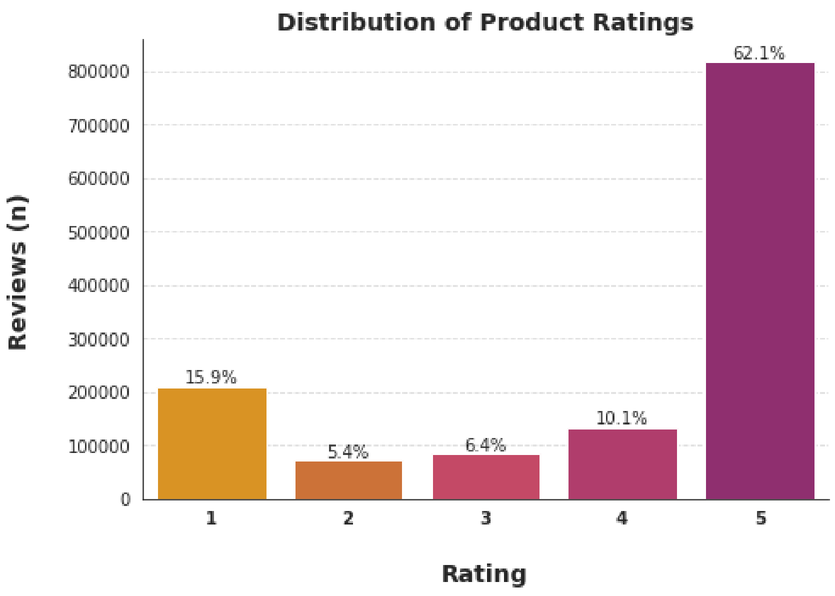
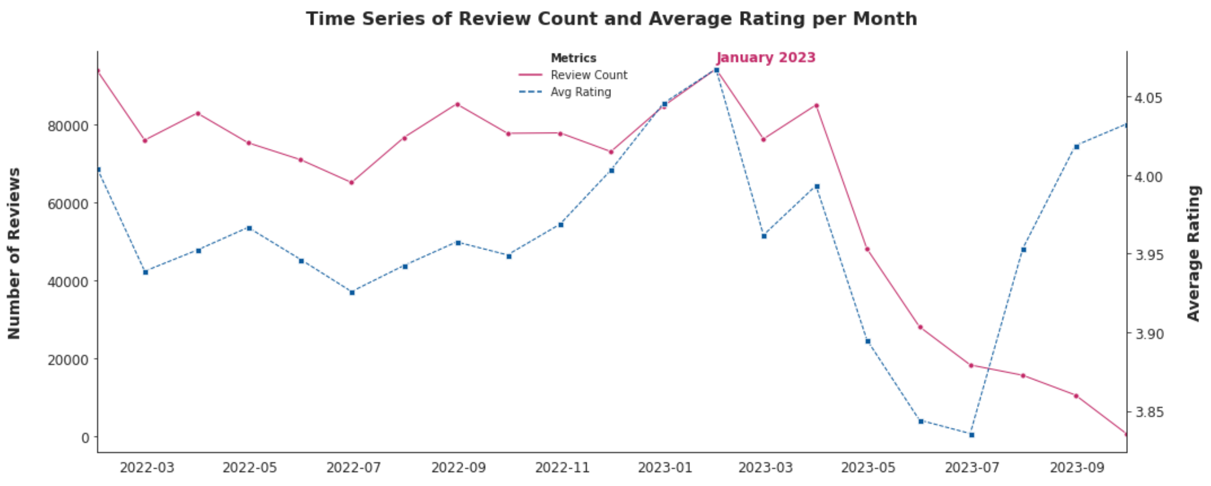


Figure C3 – Barplot with distribution of User Review Ratings.



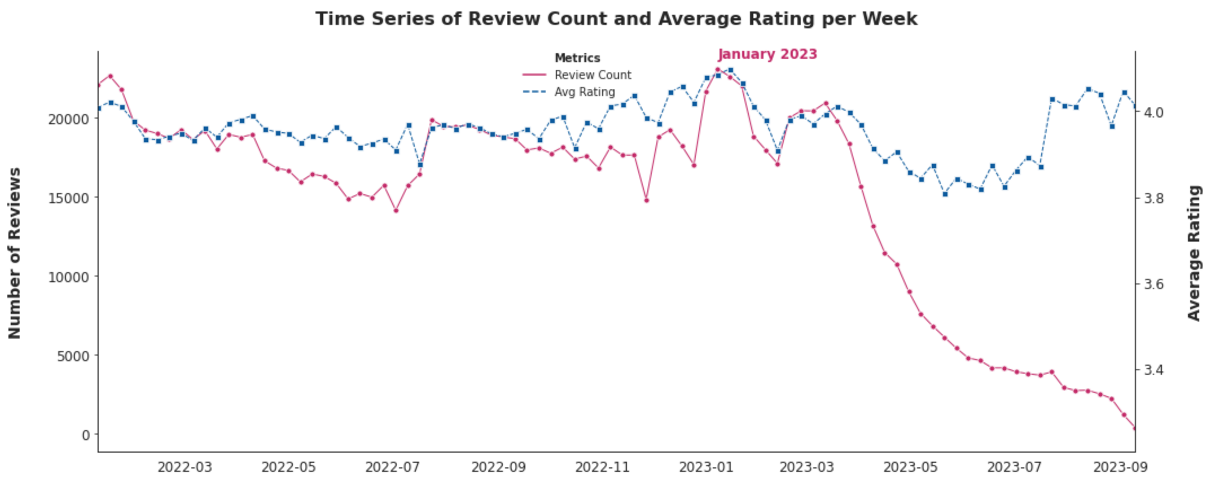


Figure C4 – Time Series of Review Count and Average Rating per Month, per Week and per Day.

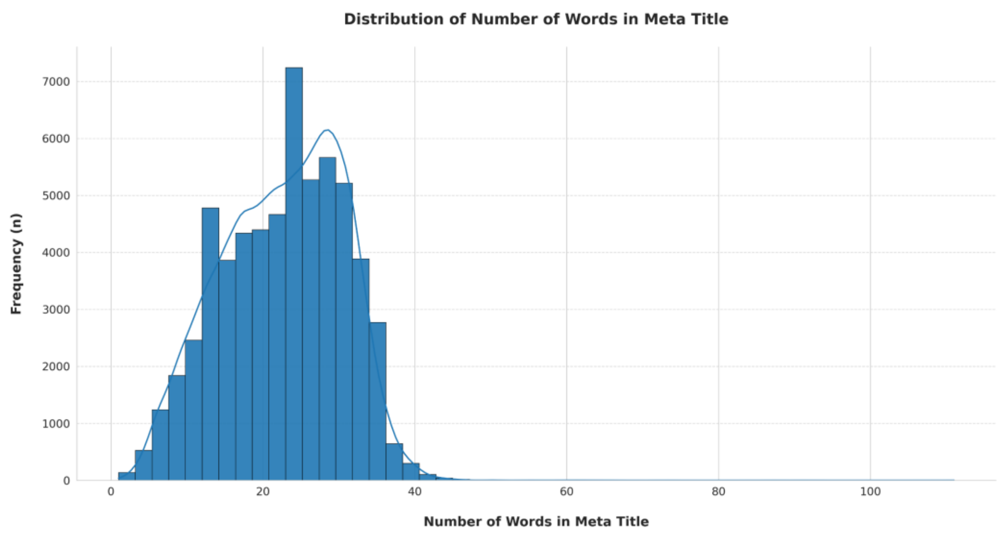
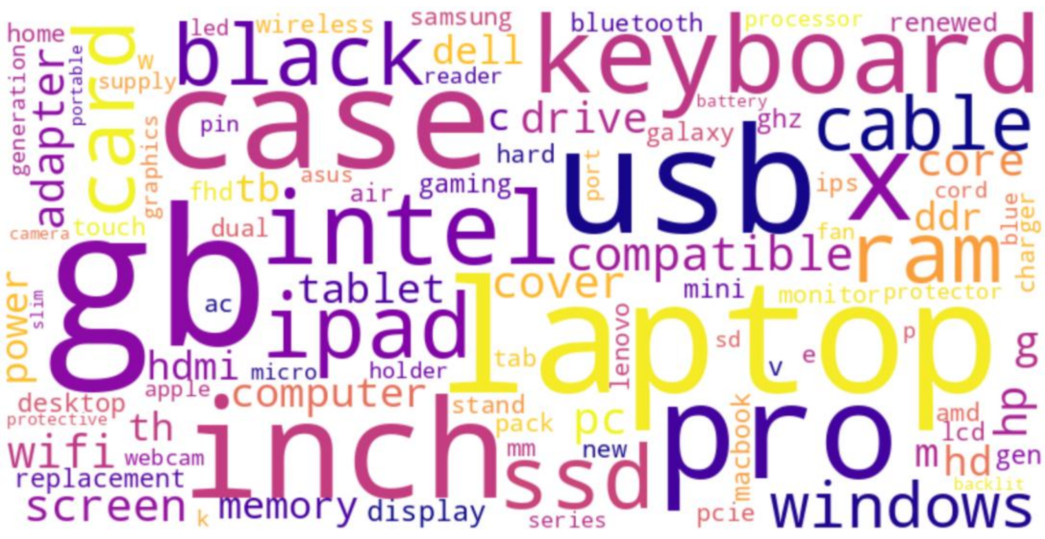


Figure C5 – Histogram with Distribution of Number of Words in *meta_title* (*n_meta_title*).



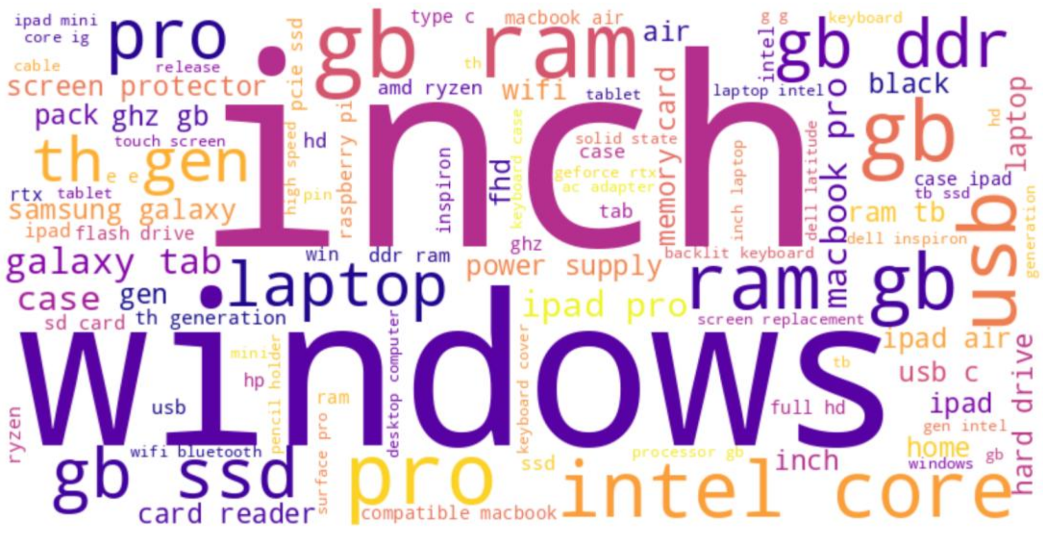


Figure C6 – Word Cloud of Top Unigrams from Product Titles (*meta_title*).

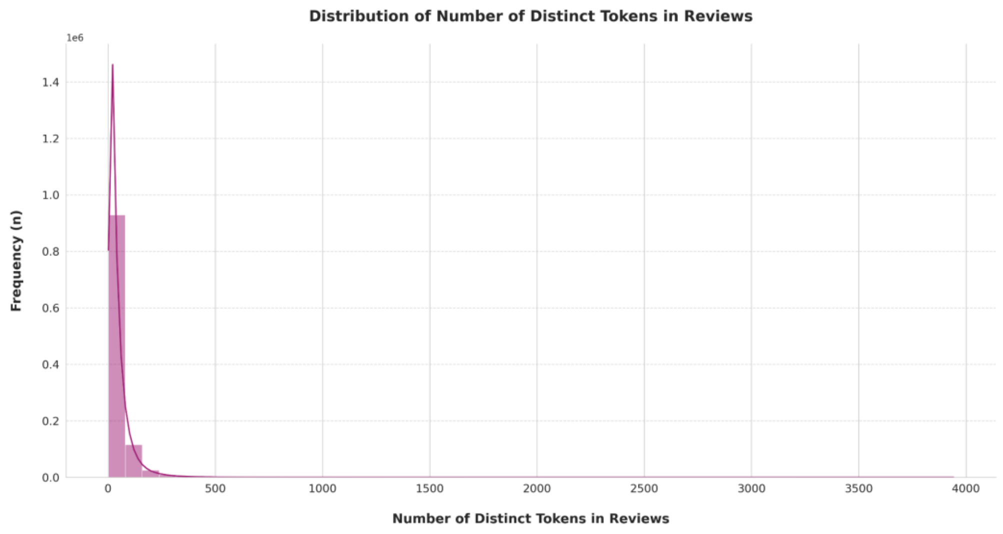


Figure C7 – Histogram with Distribution of Number of Words in *review_text* (*n_review_text*).

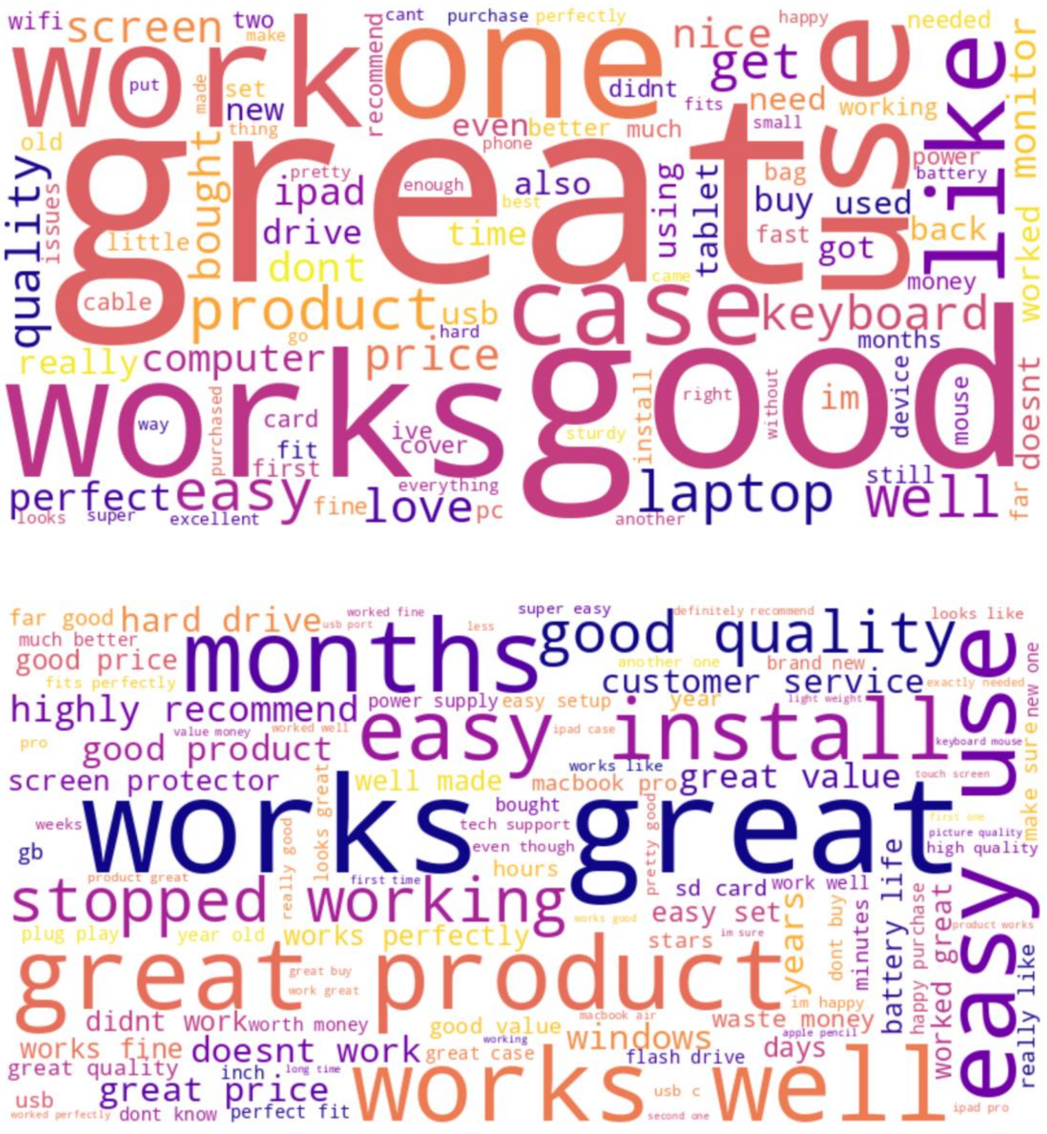


Figure C8 – Word Cloud of Top Unigrams and Bigrams from Review Texts (*review_text*).

APPENDIX D. SENTIMENT & TOPIC ANALYSIS

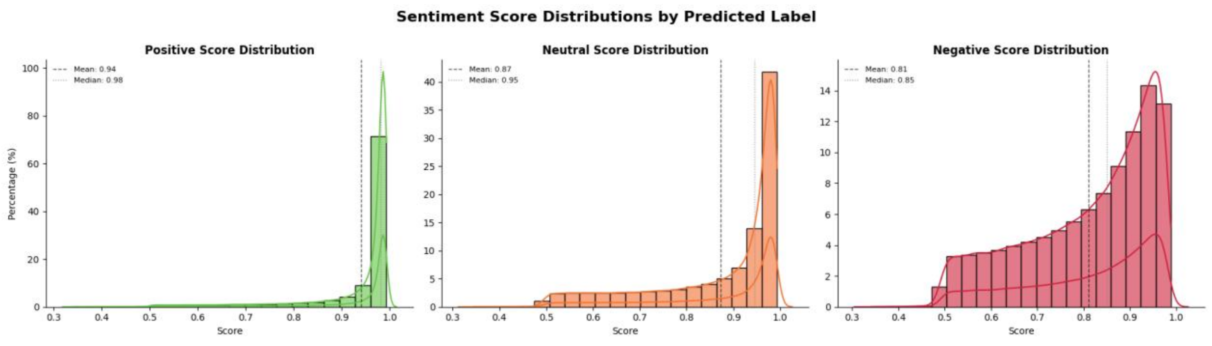


Figure D1– Score Distributions for each Predicted Sentiment Label (*TXRBSF* Model).

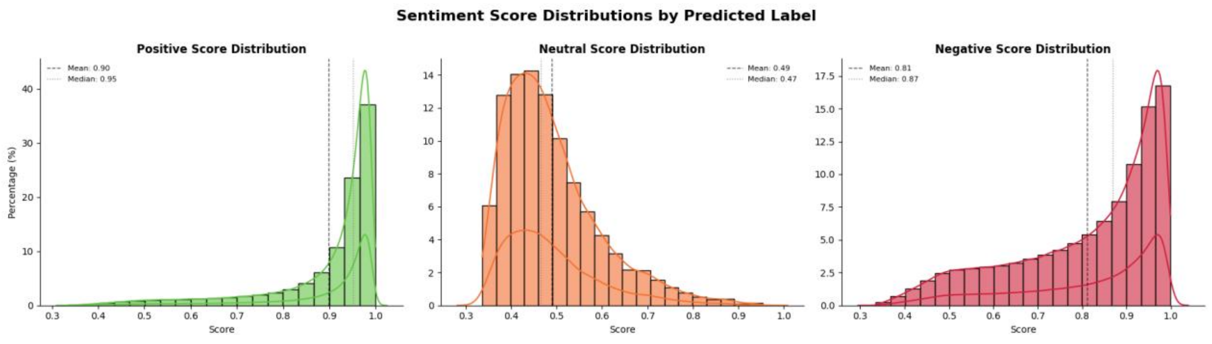


Figure D2– Score Distributions for each Predicted Sentiment Label (*mDeBERTa* Model).

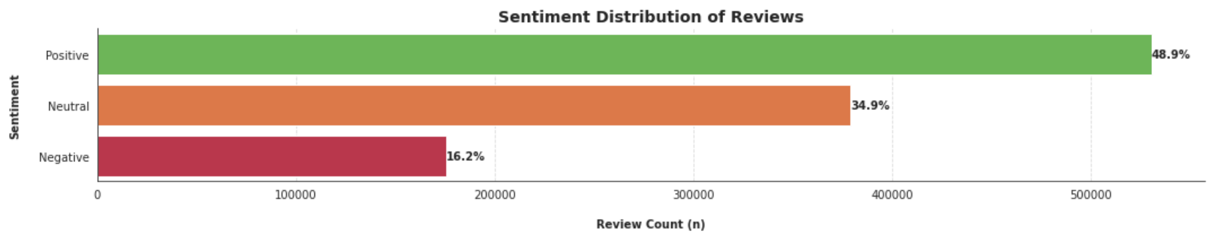


Figure D3– Sentiment Distribution of Reviews (*TXRBSF* Model).

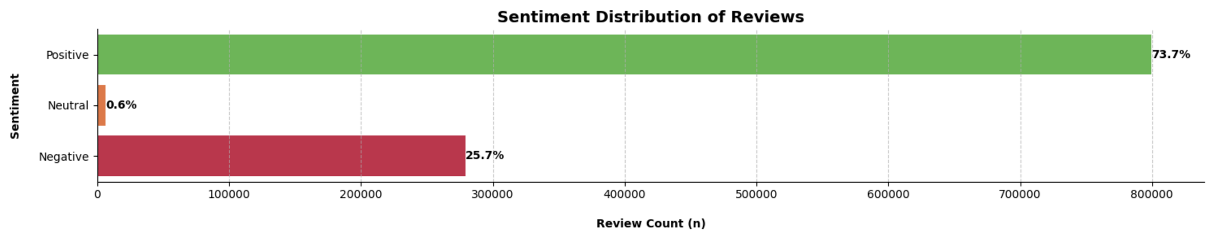
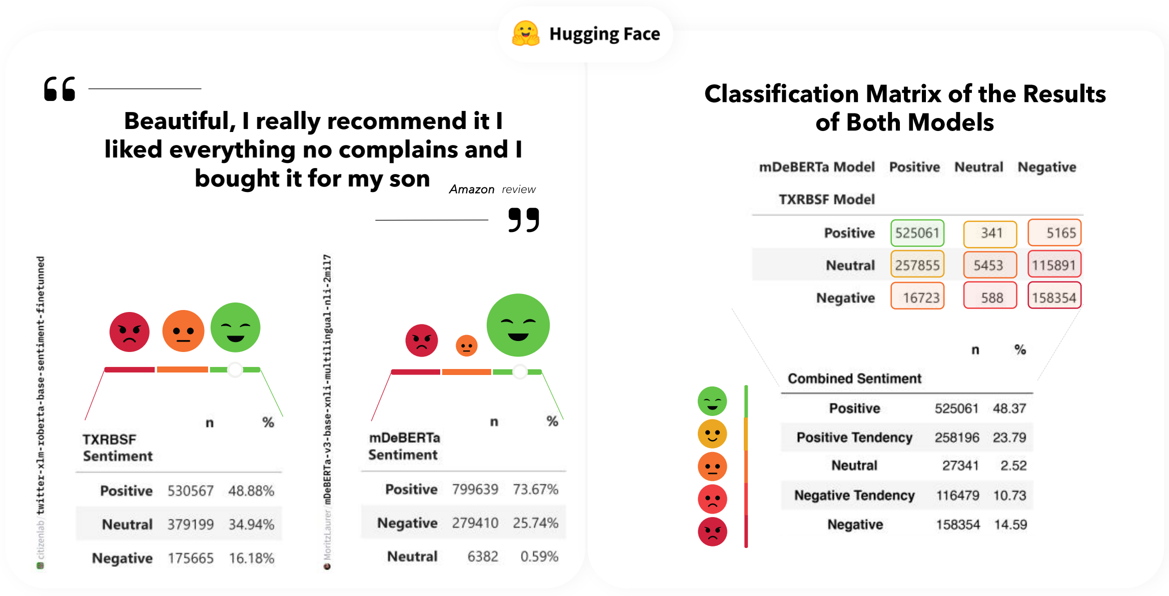


Figure D4– Sentiment Distribution of Reviews (*mDeBERTa* Model).



TXRBSF Sentiment	n	%
Positive	530567	48.88%
Neutral	379199	34.94%
Negative	175665	16.18%

mDeBERTa Sentiment	n	%
Positive	799639	73.67%
Negative	279410	25.74%
Neutral	6382	0.59%

Classification Matrix of the Results of Both Models

mDeBERTa Model / TXRBSF Model	Positive	Neutral	Negative
Positive	525061	341	5165
Neutral	257855	5453	115891
Negative	16723	588	158354

Combined Sentiment	n	%
Positive	525061	48.37
Positive Tendency	258196	23.79
Neutral	27341	2.52
Negative Tendency	116479	10.73
Negative	158354	14.59

Figure D5 – Overview of sentiment analysis results: individual model classifications for an example review (left), comparative classification matrix of the two Transformer models (top right), and the resulting combined sentiment distribution (bottom right).

Table D1 – Distribution of Predicted Combined Sentiment across True User Rating Categories (1-5 stars), showing counts and row percentages.

	Predicted Combined Sentiment				
True Rating Category	**Negative**	**Negative Tendency**	**Neutral**	**Positive Tendency**	**Positive**
1	103332 (61.9%)	49805 (29.9%)	4274 (2.6%)	8393 (5.0%)	1039 (0.6%)
2	26637 (45.8%)	20173 (34.7%)	2862 (4.9%)	7311 (12.6%)	1150 (2.0%)
3	18539 (27.1%)	21712 (31.8%)	5005 (7.3%)	18551 (27.1%)	4548 (6.7%)
4	5628 (5.3%)	12828 (12.0%)	5809 (5.5%)	43060 (40.4%)	39148 (36.8%)
5	4218 (0.6%)	11961 (1.7%)	9391 (1.4%)	180881 (26.4%)	479176 (69.9%)

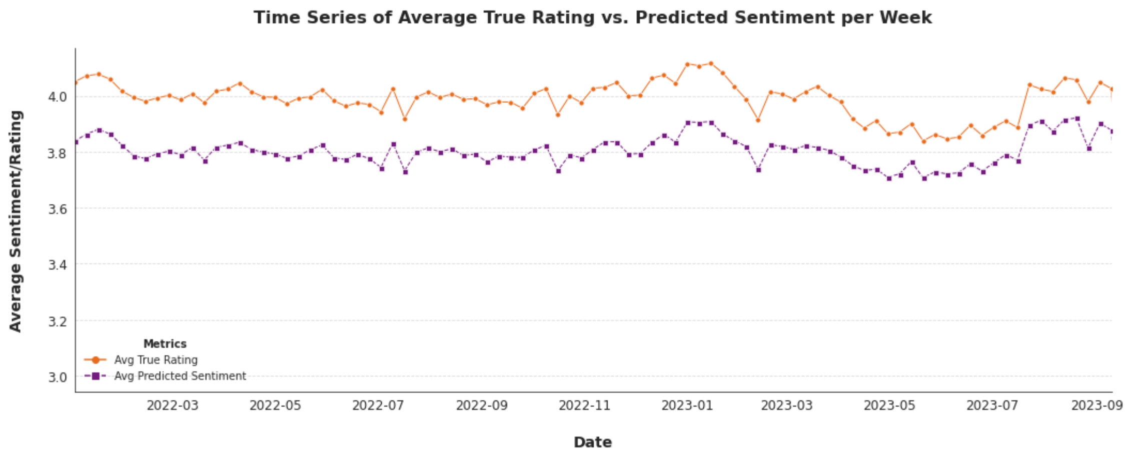


Figure D6 – Temporal Analysis of Average True Rating vs. Predicted Sentiment: Weekly trends from January 2022 to December 2023.

Table D2 – Distribution of Combined Review Sentiment Across Product Price Buckets.

	Combined Sentiment				
Price Category	**Negative**	**Negative Tendency**	**Neutral**	**Positive Tendency**	**Positive**
0 - 10	15967 (14.7%)	11152 (10.3%)	2523 (2.3%)	28919 (26.7%)	49877 (46.0%)
10 - 50	83618 (14.1%)	57187 (9.7%)	14362 (2.4%)	137872 (23.3%)	299363 (50.5%)
50 - 100	20333 (15.5%)	15430 (11.7%)	3467 (2.6%)	32399 (24.7%)	59795 (45.5%)
100 - 250	22864 (15.0%)	18923 (12.4%)	4145 (2.7%)	36604 (24.0%)	7015 (45.9%)
250 - 500	9232 (15.2%)	7755 (12.8%)	1571 (2.6%)	13864 (22.9%)	28172 (46.5%)
500+	6340 (15.9%)	6032 (15.1%)	1273 (3.2%)	8538 (21.4%)	17699 (44.4%)

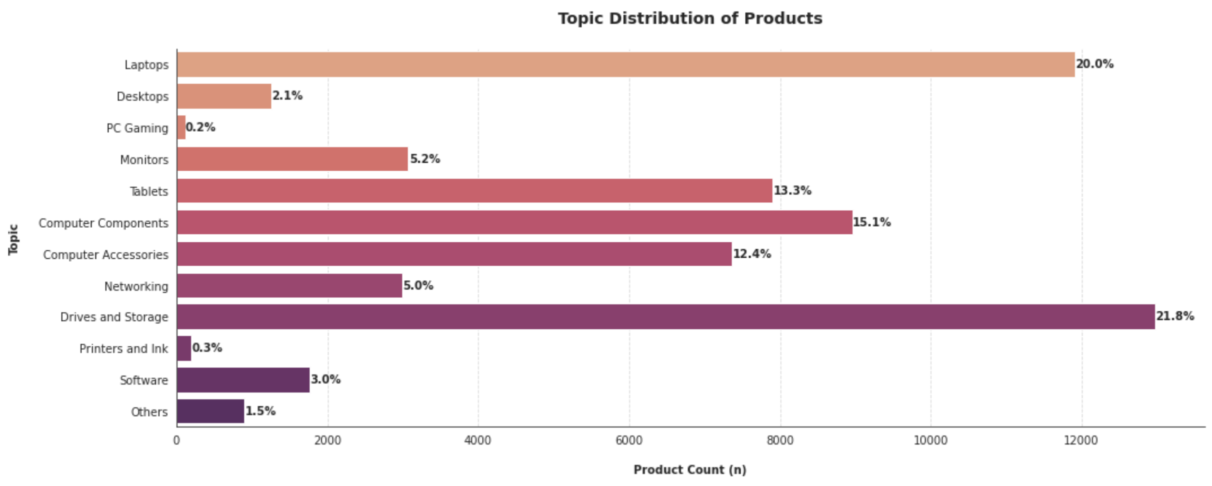


Figure D7 – Barplot with overall Distribution of Product Topics.

Table D3 – Predominant Combined Sentiment for Key Product Topics.

	Combined Sentiment				
Topic Label	**Negative**	**Negative Tendency**	**Neutral**	**Positive Tendency**	**Positive**
Laptops	23044 (17.1%)	15137 (11.3%)	3486 (2.6%)	28704 (21.3%)	64074 (47.7%)
Desktops	1745 (10.4%)	1881 (11.2%)	472 (2.8%)	4829 (28.8%)	7855 (46.8%)
PC Gaming	193 (13.0%)	182 (12.3%)	43 (2.9%)	410 (27.7%)	651 (44.0%)
Monitors	8867 (13.6%)	7192 (11.1%)	1997 (3.1%)	14483 (22.3%)	32436 (49.9%)
Tablets	32729 (15.4%)	18748 (8.8%)	5434 (2.6%)	38256 (18.0%)	117313 (55.2%)
Computer Components	11252 (11.4%)	12005 (12.1%)	2810 (2.8%)	30190 (30.5%)	42827 (43.2%)
Computer Accessories	27499 (14.3%)	18670 (9.7%)	5114 (2.7%)	42475 (22.1%)	98172 (51.1%)
Networking	13931 (15.6%)	11077 (12.4%)	1961 (2.2%)	26421 (29.5%)	36082 (40.3%)
Drives and Storage	33957 (13.9%)	27642 (11.4%)	5149 (2.1%)	64039 (26.3%)	112642 (46.3%)
Printers and Ink	447 (16.5%)	298 (11.0%)	110 (4.0%)	603 (22.2%)	1259 (46.3%)
Software	3358 (16.9%)	2647 (13.4%)	534 (2.7%)	5506 (27.8%)	7775 (39.2%)
Others	1332 (15.1%)	1000 (11.3%)	231 (2.6%)	2280 (25.9%)	3975 (45.1%)

Table D4 – Combined Sentiment Variation Across Top Product Stores.

	Combined Sentiment				
Meta Store	**Negative**	**Negative Tendency**	**Neutral**	**Positive Tendency**	**Positive**
Amazon Renewed	7716 (18.1%)	5657 (13.3%)	1068 (2.5%)	7905 (18.6%)	20258 (47.5%)
HP	3175 (19.5%)	2267 (13.9%)	497 (3.1%)	3091 (19.0%)	7263 (44.6%)
Lenovo	1697 (19.3%)	1215 (13.8%)	243 (2.8%)	1611 (18.3%)	4043 (45.9%)
ASUS	2663 (16.0%)	2628 (15.8%)	523 (3.1%)	4103 (24.6%)	6749 (40.5%)
Dell	1342 (18.8%)	1129 (15.8%)	185 (2.6%)	1567 (22.0%)	2909 (40.8%)
SanDisk	2488 (9.9%)	2356 (9.4%)	521 (2.1%)	7399 (29.6%)	12244 (49.0%)
SAMSUNG	3400 (12.7%)	2958 (11.0%)	588 (2.2%)	6863 (25.6%)	13032 (48.6%)
Generic	232 (20.4%)	138 (12.2%)	25 (2.2%)	279 (24.6%)	461 (40.6%)
MOSISO	2209 (19.2%)	1063 (9.2%)	366 (3.2%)	1890 (16.4%)	5971 (51.9%)
SCREENARAMA	24 (3.5%)	26 (3.8%)	7 (1.0%)	198 (28.6%)	438 (63.2%)

APPENDIX E. CLUSTERING ANALYSIS

We conducted cluster profiling by analysing insights derived from various plots and tables analysed (**Figure E5**, **Figure E6**, **Figure E7**, and **Table E2**). **Table E1** summarizes the three distinct clusters, each providing unique perspectives to guide customized business strategies. The following table details the segment characteristics along with corresponding marketing recommendations.

Table E1 – Segment Profiles & Recommended Marketing Approaches.

	Segment Profile & Key Characteristics	Recommended Marketing Approach
0 | Mainstream Mentions 24 173 reviews (24.0%)	- Reviews for lower-priced products (mean price ≈ $200 less than other clusters), often from Amazon Renewed, a key brand offering reconditioned items. - Features shorter product titles. - Reviews reflect good mean ratings (4.76) and consistent sentiment across models. - Associated with products having significantly more reviews (3 × higher than other clusters).	- Identify users purchasing mainstream, affordable items and offer promotions on higher-value or complementary products—items they would likely never buy on their own—using *PageRank*-enhanced suggestions at checkout. - Target customers buying trending products with themed offers, promoting top-selling items from different or related categories. - Highlight products with a large volume of positive reviews in marketing materials and invest in brand visibility through trust-based campaigns.
1 | Premium Perspectives 46 711 reviews (46.5%)	- Largest cluster, comprising nearly half of all reviews. - Reviews for products with fewer ratings, yet good ones (mean 4.79) and consistent sentiment across models. - Characterized by longer product titles, indicating detailed descriptions. - Covers a wide range of products, predominantly higher-priced items like laptops and gaming PCs from brands like HP (11.1%) and ASUS (10.6%).	- Identify users who tend to purchase higher-priced products and use recommendation systems to suggest premium upgrades. - Launch a rewards program for products with few reviews by offering, for example, points for feedback that can be redeemed for discounts to increase the number of ratings, amplifying the visibility of these well-rated products.
2 | Critical Reviews 29 592 reviews (29.5%)	- Dominated by negative reviews with a low mean rating (1.7 vs. 4.8 for others), often tied to Amazon Renewed (13.7%). - Slightly higher mean helpful votes (2.04), indicating greater influence on consumers. - Features longer reviews, reflecting detailed negative feedback.	- Identify products and users with consistently bad reviews, checking if feedback is valid or from haters. - Use personalized outreach to address complaints with tailored responses or offers to convert dissatisfied customers. - Improve product quality where needed, share transparent updates, and evaluate if products or brands should stay on sale based on their impact.

Principal Component	Variance	Cumulative Variance
1	0.267726	0.267726
2	0.209250	0.476976
3	0.180999	0.657975
4	0.124116	0.782091
5	0.110632	0.892723
6	0.080756	0.973479
7	0.026521	1.000000

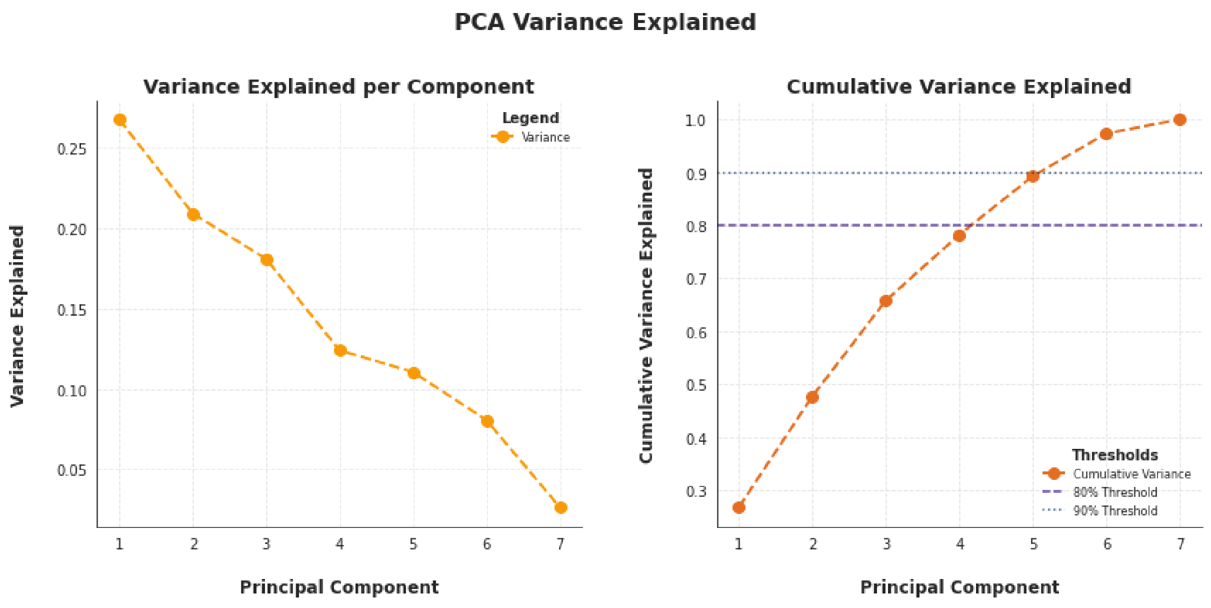


Figure E1 – Principal Component Analysis (PCA) for metric variables.

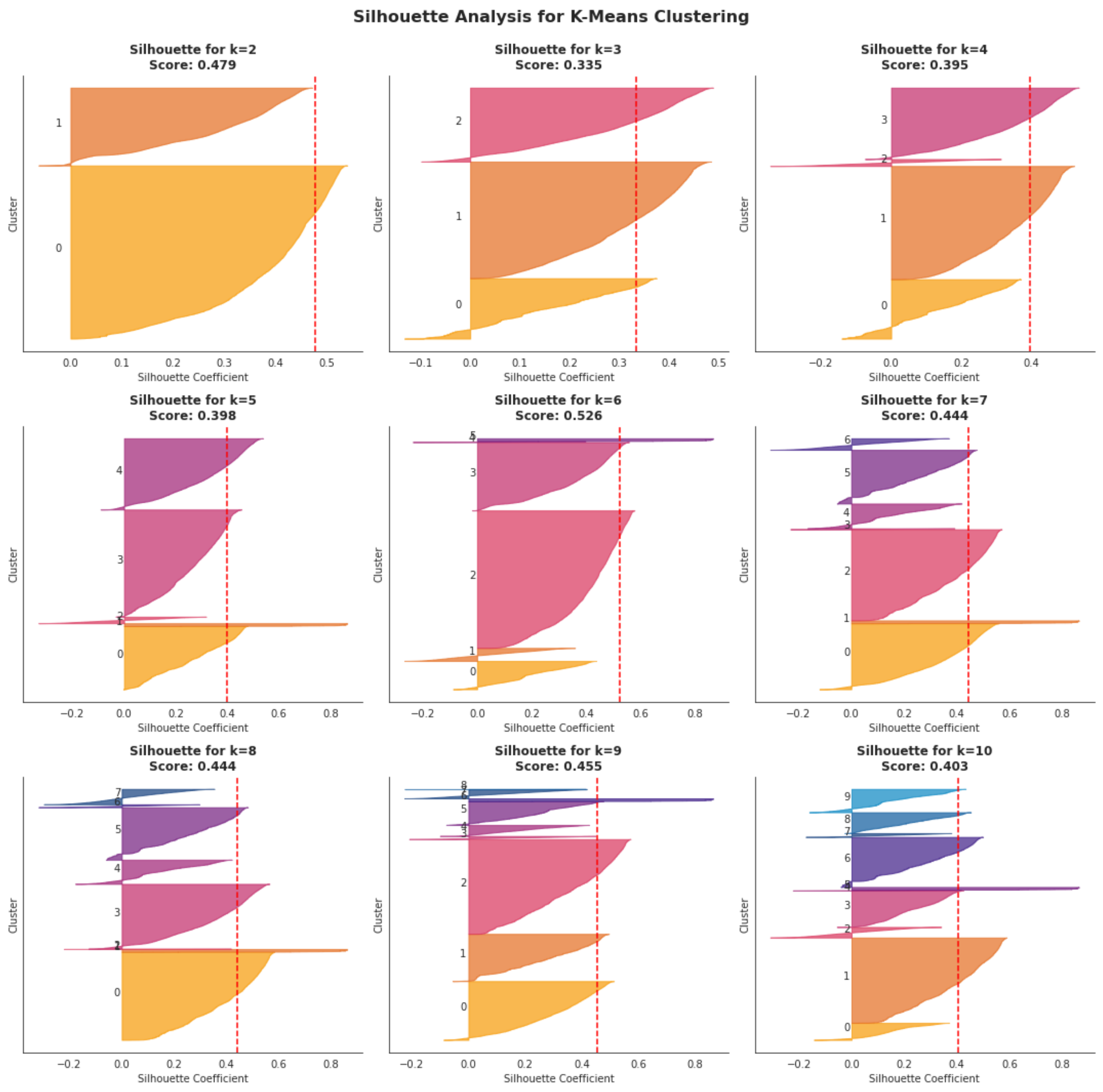


Figure E2 – Silhouette Analysis for K-Means Clustering ($k = 2$ to $k = 10$).

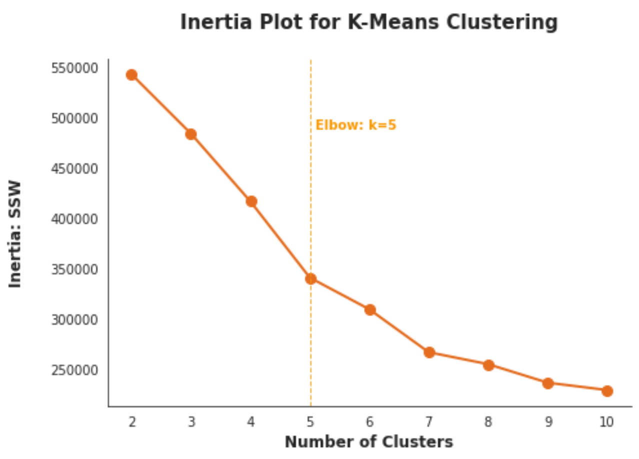


Figure E3 – Inertia Plot for K-Means Clustering (Elbow Method).

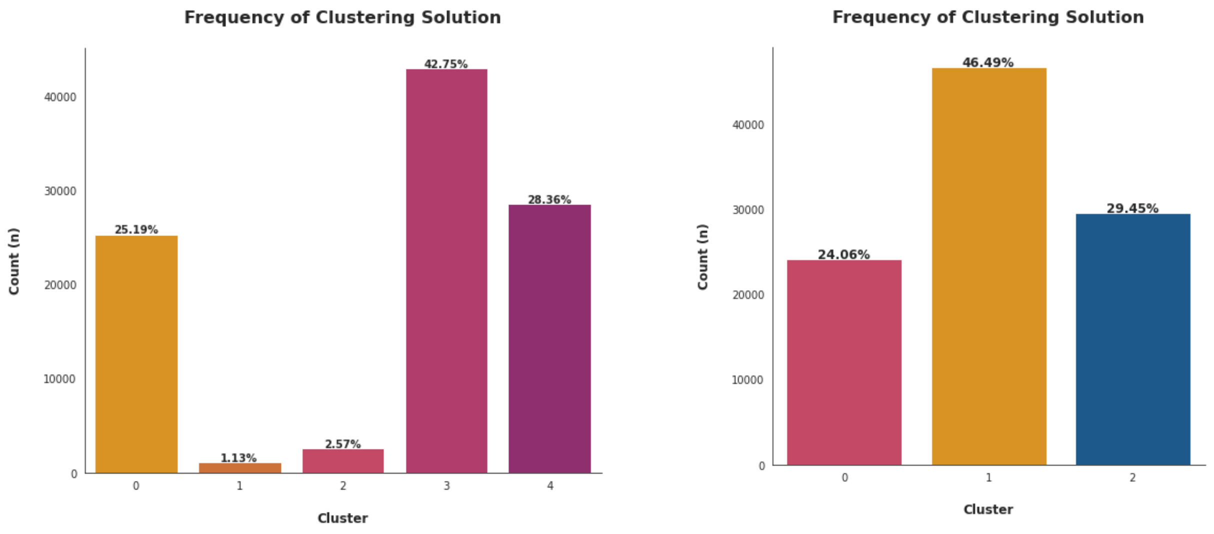


Figure E4 –Cluster Distribution for K-Means with $k = 5$ (left) and $k = 3$ (right).

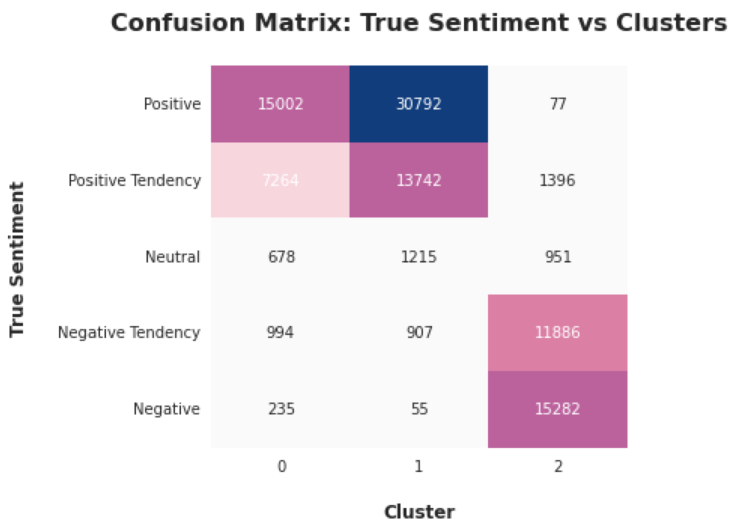


Figure E5 – Heatmap Confusion Matrix with True Sentiment vs. Clusters ($k = 3$).

Table E2 – Cluster Profiling: Average Values of Numerical Variables by Cluster.

Cluster	0	1	2
rating	4.758119	4.798506	1.738713
helpful_vote	1.245067	1.643082	2.042343
n_reviews_text	29.271253	35.039669	47.139159
meta_price	440.594457	656.416156	605.255435
n_product_title	15.437141	27.401854	23.887402
meta_rating_number	6891.566458	1004.750016	1888.559408
Combined_Sentiment_index	3.481157	3.590824	0.617870

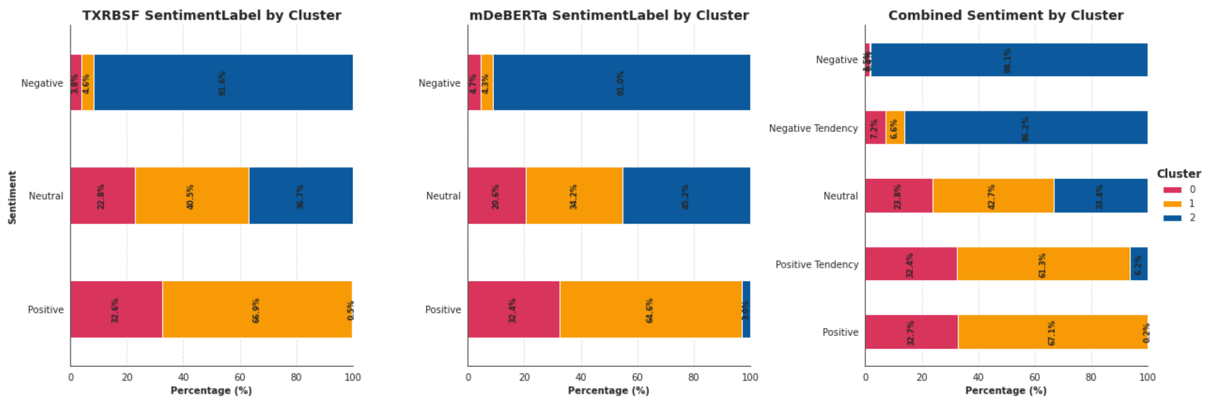


Figure E6 – Stacked Barplots of Sentiment Labels by Cluster ($k = 3$).

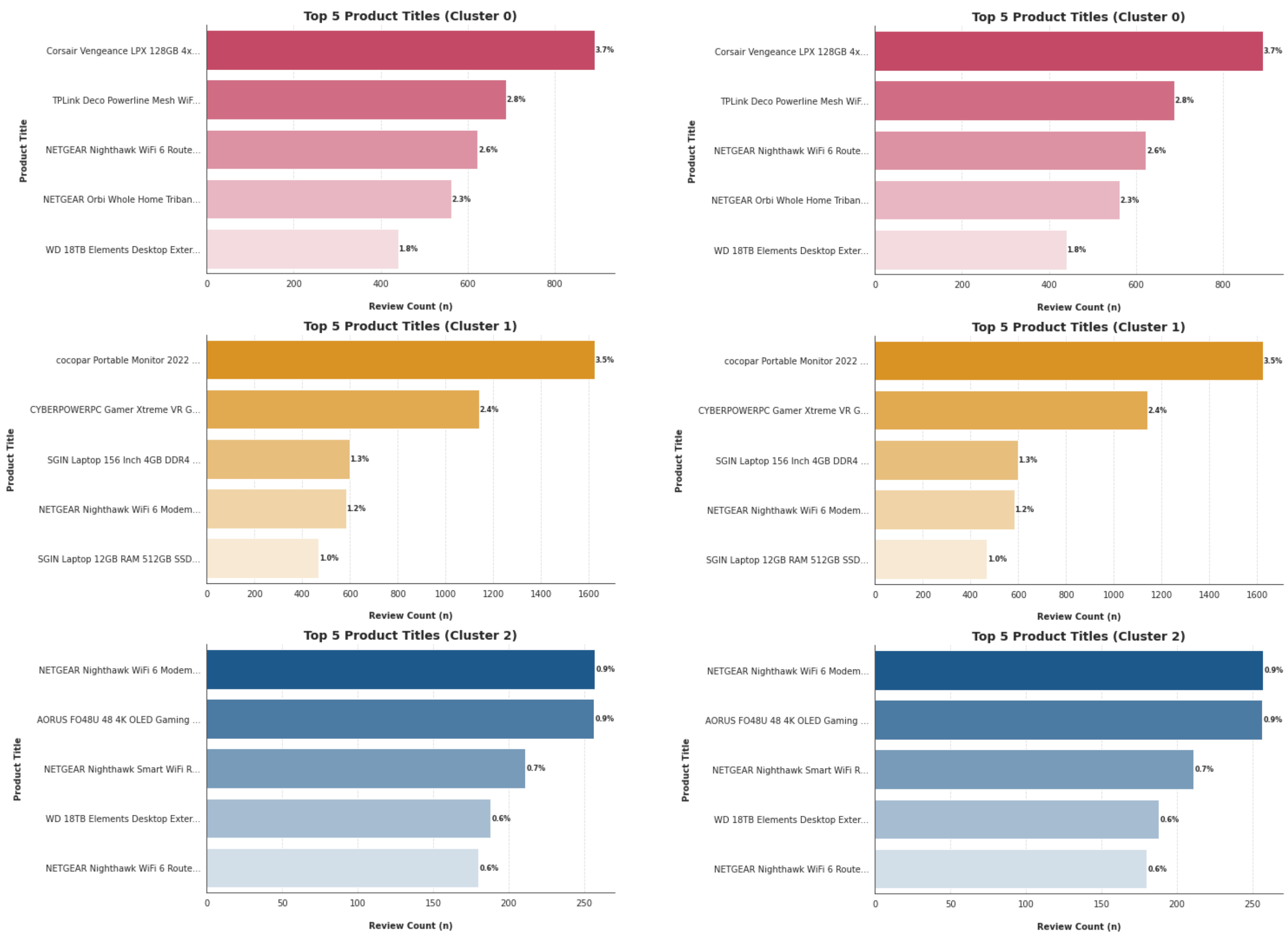


Figure E7 – Top 5 Product Titles and 5 Brands by Cluster ($k = 3$).

APPENDIX F. GRAPH ANALYSIS

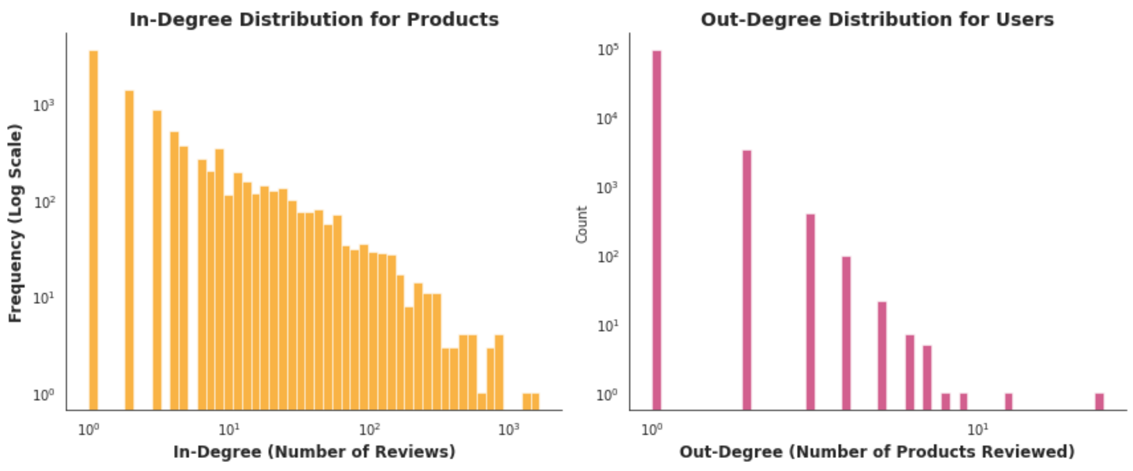


Figure F1 – Histograms showing the In-Degree Distribution for Products (Number of Reviews Received) and Out-Degree Distribution for Users (Number of Products Reviewed), Both on a $log - log$ Scale.

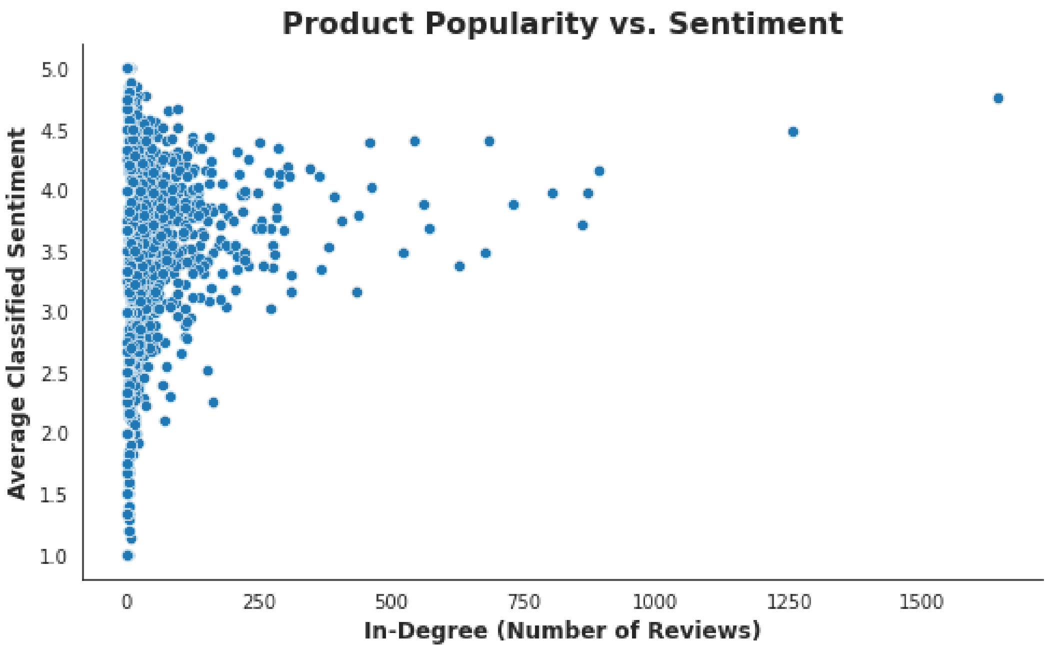


Figure F2 – Scatter Plot illustrating the Relationship between Product In-Degree (Number of Reviews) and Average Classified Sentiment.

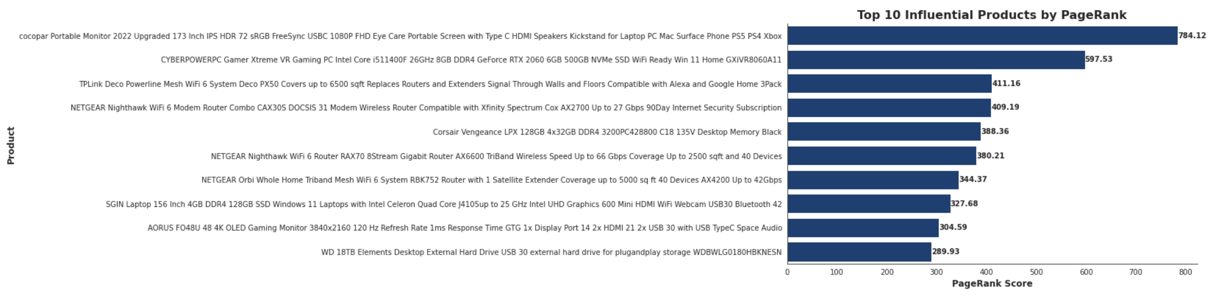


Figure F3 – Barplot Displaying the Top 10 Products Ranked by their PageRank Scores.

Table F1 – Absolute and Relative Frequency of each Community Size detected by Label Propagation Algorithm.

Community Size	n	%
1	11771	66.11
2	1768	9.93
3	900	5.06
4	530	2.98
5	372	2.09
6	272	1.53
7	203	1.14
8	192	1.08
9	134	0.75
10	115	0.65
+10	1547	8.69

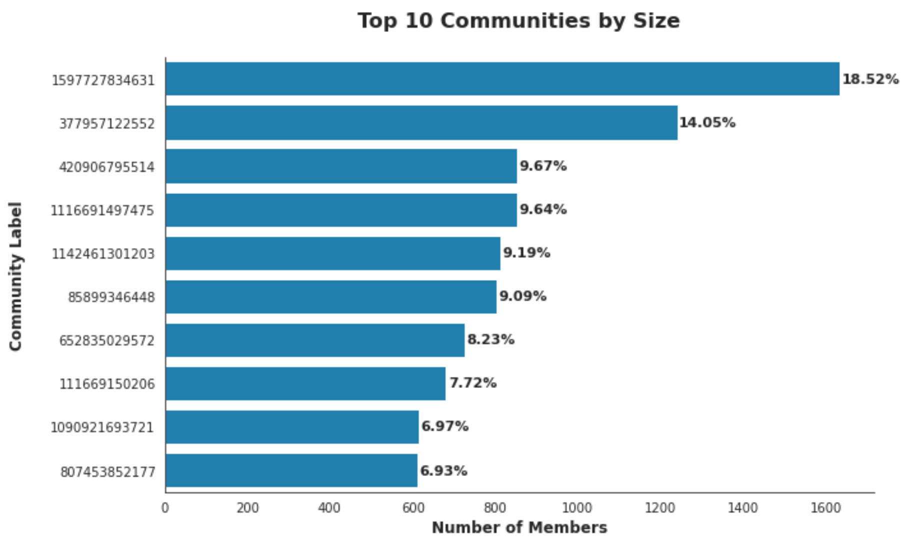


Figure F4 – Barplot showing the Number of Members (Users and Products) in the Top 10 Largest Communities detected by Label Propagation Algorithm.

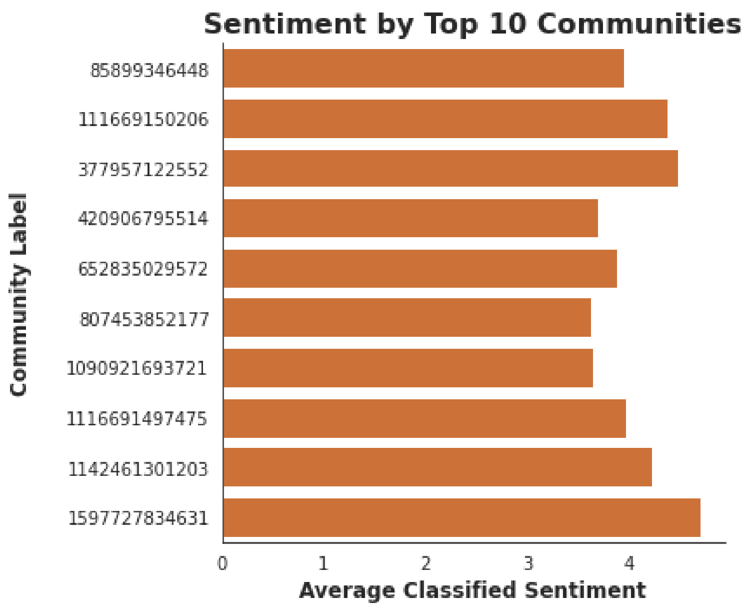


Figure F5 – Barplot displaying the Average Classified Sentiment Score for the Top 10 Largest Communities detected by Label Propagation Algorithm

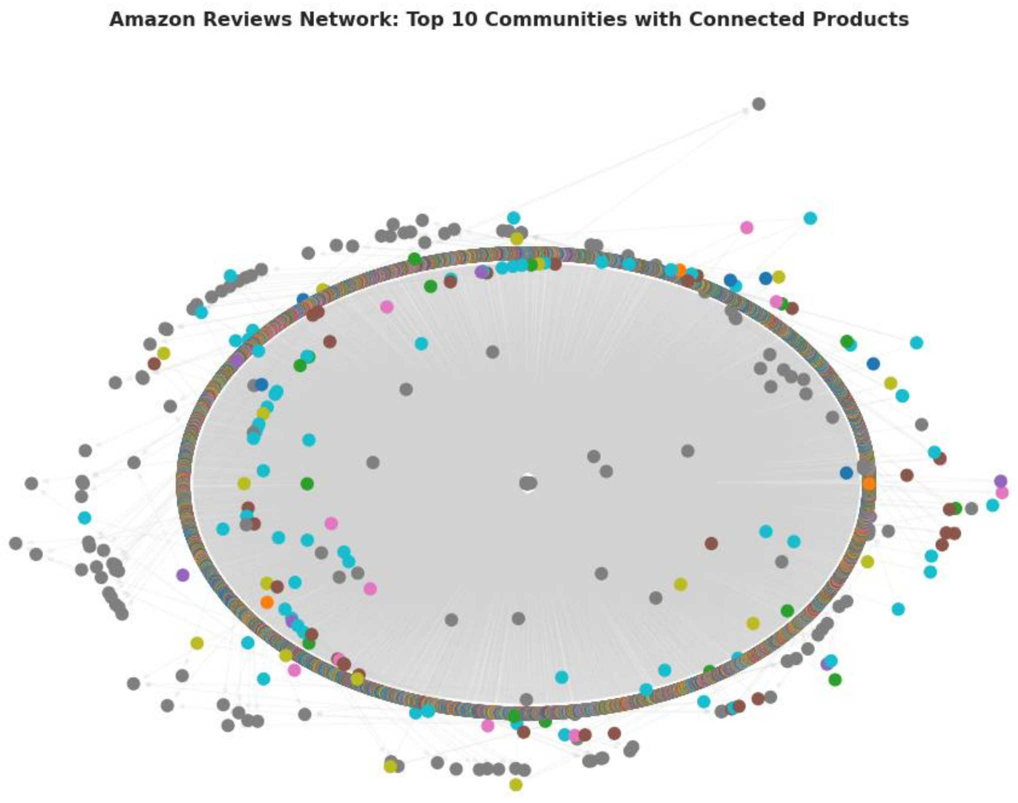
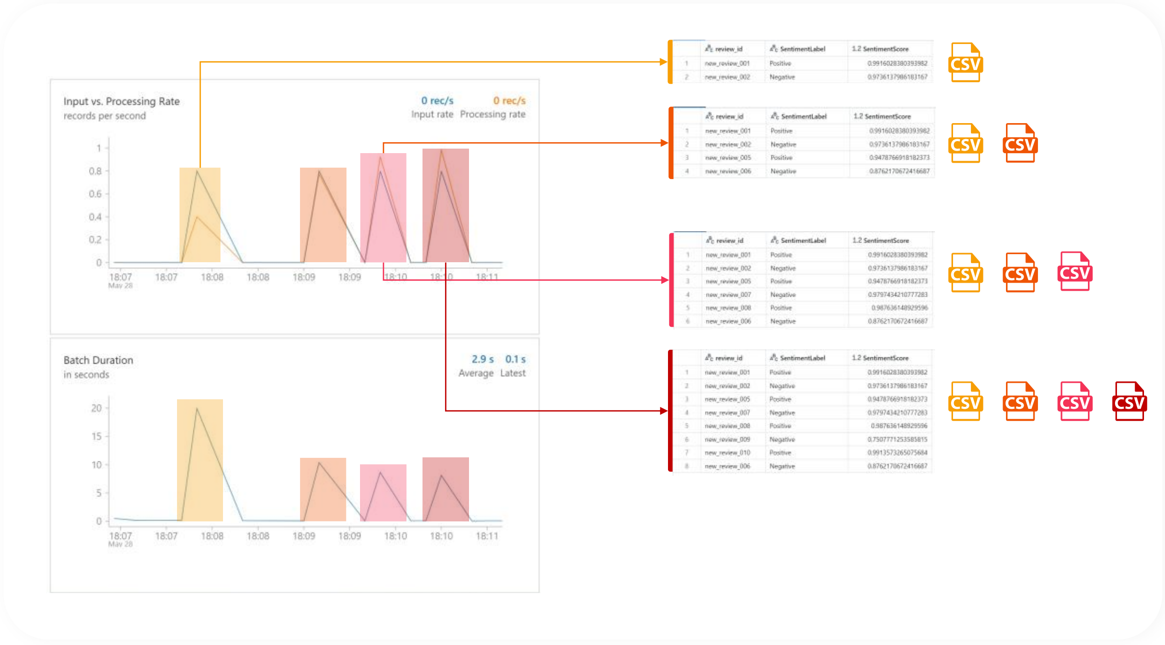


Figure F6 – Graph Visualization illustrating the Connections between Users within the Top 10 Communities and the Products they Reviewed

APPENDIX G. STREAMING SMULATION



	review_id	SentimentLabel	1.2 SentimentScore
1	new_review_001	Positive	0.9916028380393982
2	new_review_002	Negative	0.9736137986183167

	review_id	SentimentLabel	1.2 SentimentScore
1	new_review_001	Positive	0.9916028380393982
2	new_review_002	Negative	0.9736137986183167
3	new_review_005	Positive	0.9478766918182373
4	new_review_006	Negative	0.8762170672416687

	review_id	SentimentLabel	1.2 SentimentScore
1	new_review_001	Positive	0.9916028380393982
2	new_review_002	Negative	0.9736137986183167
3	new_review_005	Positive	0.9478766918182373
4	new_review_007	Negative	0.9797434210777283
5	new_review_008	Positive	0.987636148929596
6	new_review_006	Negative	0.8762170672416687

	review_id	SentimentLabel	1.2 SentimentScore
1	new_review_001	Positive	0.9916028380393982
2	new_review_002	Negative	0.9736137986183167
3	new_review_005	Positive	0.9478766918182373
4	new_review_007	Negative	0.9797434210777283
5	new_review_008	Positive	0.987636148929596
6	new_review_009	Negative	0.7507771253585815
7	new_review_010	Positive	0.9913573265075684
8	new_review_006	Negative	0.8762170672416687

Figure G1 – Databricks Dashboard Visualizing Simulated Streaming Sentiment Analysis.

ANNEX A. CRISP-DM

To address the defined objectives, the **CRISP-DM** methodology was employed. This methodology is a dynamic and fluid iterative approach (**Figure A1**) and relies on the exploration and refinement of its different stages to achieve better results, thereby supporting more effective and meaningful decision-making. By definition, this methodology comprises six phases: *Business Understanding*, *Data Understanding*, *Data Preparation*, *Modelling*, *Evaluation*, and *Deployment*, with various tasks within each phase that will be explored in this project. [10]

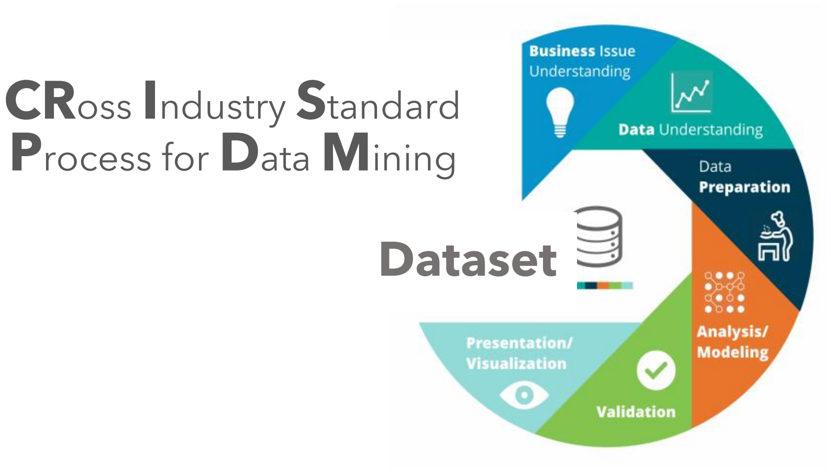

CRoss Industry Standard Process for Data Mining
Business Issue Understanding
Data Understanding
Data Preparation
Analysis/ Modeling
Validation
Presentation/ Visualization
Dataset

Figure A1 – CRISP-DM Methodology Cycle.
[Adapted from [10]]

1. Business Understanding | Understand what the business needs from the data.

The goal was to understand consumer purchasing behaviour in the Electronics – Computers segment on Amazon. This included identifying patterns in review sentiment, topics of interest, influential products/users, and detecting insights that could guide strategic decisions for product design, marketing, and vendor management.

2. Data Understanding | Collect and become familiar with the data.

We retrieved data from the Amazon Reviews 2023 collection, specifically filtering it for Electronics – Computers between 2022–2023. This phase involved examining review and product metadata structures, identifying null values, types, distributions, and assessing relevance. Exploratory Data Analysis (EDA) helped profile the attributes, assess business value, and guide preprocessing decisions.

3. Data Preparation | Prepare data for *Modelling*.

In this project: We performed deduplication, filtering (e.g., removing unverified reviews, products without prices), and missing value treatment. Data types were adjusted, and structured joins were made using *parent_asin*. Natural Language Processing (NLP) preprocessing included lowercasing, punctuation removal, tokenization, stopword removal, and n-gram extraction on product titles and review texts. Final variables were selected based on business relevance and data quality.

4. Modelling | Apply algorithms to extract patterns.

We conducted Sentiment Analysis using *XLM-RoBERTa* and *mDeBERTa* transformer models to capture textual sentiment and extract business insights. Topic classification used a zero-shot *mDeBERTa* model to label products adapted from *Amazon's* taxonomy. Clustering was applied on high-value products using K-Means to identify meaningful segments based on sentiment, price, and influence metrics. Graph Analysis used *GraphFrames* with *PageRank* and *Label Propagation* to detect influential products and communities in user-product interactions.

5. Evaluation | Assess the models to ensure they meet business needs.

Sentiment model outputs were merged to verify consistency between users' written opinions and ratings, combining their outputs into a 5-class sentiment scheme. Evaluation focused on alignment, uncovering inconsistencies, and highlighting mid-range rating insights. Topic and cluster results were assessed for interpretability and business usefulness. Clusters were analysed based on review, product, and user characteristics. Graph metrics were evaluated in terms of identifying influence and community structure.

6. Deployment | Use the results in practical applications.

A streaming simulation was developed to demonstrate how the proposed solution could operate in near real-time. Though not deployed in a production environment due to platform limitations, we outlined how businesses could use these insights to improve product offerings, vendor management, and customer engagement strategies. Additionally, results were visualized and summarised in dashboards to support decision-making in the presentation and final report.

ANNEX B. NATURAL LANGUAGE PREPROCESSING (NLP)

NLP involves computational techniques for the analysis and representation of human language. [16][17] It is fundamental to extracting structured insights from unstructured text, enabling tasks such as sentiment classification, topic modelling, and semantic clustering. This annex outlines the theoretical foundations of the preprocessing pipeline used to prepare product titles and customer reviews for downstream analysis.

Text Normalization

Text normalization refers to techniques that reduce variation in language expression without altering semantic content. This includes:

- **Lowercasing**, which standardizes tokens (e.g., *"Laptop"* and "*laptop"*) and prevents case-based token fragmentation.
- **Punctuation and HTML tag removal**, which reduces syntactic noise irrelevant for most lexical analyses ([16], Ch. 2).

Such normalization ensures that equivalent expressions are treated uniformly by the model.

Tokenization

Tokenization is the process of segmenting text into smaller units, typically words or subwords, depending on the desired level of granularity. In traditional NLP pipelines, whitespace or regex-based tokenizers are used. These methods align with classical bag-of-words assumptions, where word order is disregarded ([16], Ch. 2.2). While tokenization appears simple, it underpins all subsequent steps in NLP workflows.

Stopword Removal

Stopwords are high-frequency function words (e.g., *"the"*, *"and"*, *"is"*) that often lack informative value in isolation. Their removal is a form of dimensionality reduction that prioritizes semantically meaningful content words. Although stopword removal may not be suitable for all NLP tasks it remains standard in topic modelling and document classification contexts ([16], Ch. 2.3).

N-gram Construction

N-grams are contiguous sequences of *n* tokens. While unigrams capture individual word frequencies, bigrams and trigrams offer insights into collocations and phrase-level patterns (e.g. *"battery life"*). This shallow syntactic context can be informative in short text analysis, such as product titles or concise reviews. However, without smoothing, higher-order n-grams tend to suffer from data sparsity ([16], Ch. 3).

In the context of product reviews and metadata, this pipeline:

- Highlights domain-specific terminology (e.g., *"usb"*, *"case"*, *"laptop"*),
- Enables the aggregation of semantically related documents,
- Facilitates sentiment and topic detection by reducing lexical noise.

While deep learning models like *Transformers* require less manual preprocessing due to built-in embeddings and tokenization strategies [17], traditional pipelines remain essential for large-scale exploratory text mining and interpretable modelling tasks.

ANNEX C. TRANSFORMERS

Transformer models represent a major breakthrough in NLP. First introduced by Vaswani et al. (2017) in the paper *“Attention is All You Need”* [18], this architecture replaced traditional recurrent neural networks with a mechanism known as **self-attention**. Unlike sequential models, Transformers process entire sequences in parallel, allowing for more efficient training and better scalability.

At their core, *Transformers* assign attention weights to each token in a sentence, allowing the model to focus on the most relevant words when making predictions. These weights are learned during training, typically through a supervised objective such as predicting the next word in a sequence. Positional encodings are added to the input to help the model understand word order — crucial for capturing syntax and meaning in text.

Due to their flexibility and performance, Transformers are the foundation of modern large language models (LLMs) such as *BERT*, *GPT*, *RoBERTa*, and *DeBERTa*. They are pre-trained on massive corpora and can be fine-tuned or directly applied to a wide range of tasks, including question answering, sentiment analysis, text classification, translation, and more.

Applications of Transformer Models

Transformers are general-purpose models that can be adapted to multiple downstream tasks after initial training. Their ability to process diverse data types (text, image, speech) has enabled them to be used across domains. The **Figure C1** illustrates the versatility of foundation models when adapted to various tasks.

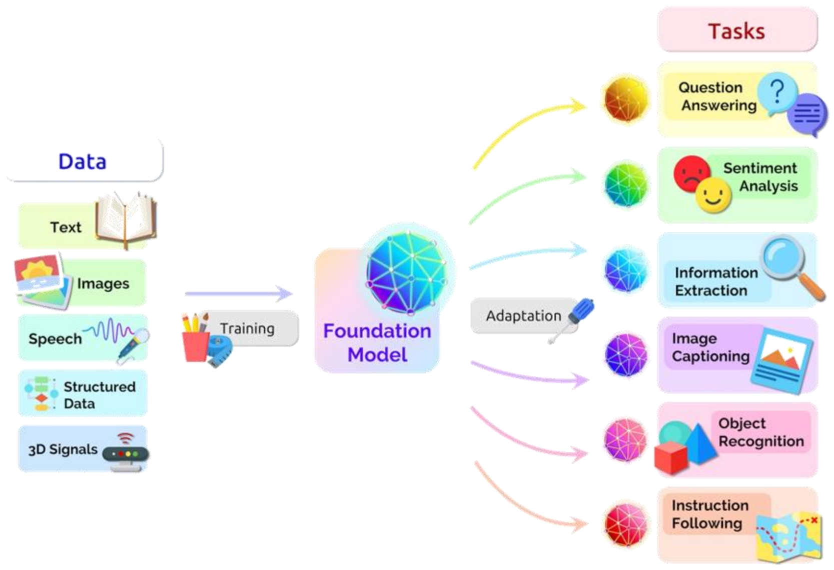


Figure C1 – Applications of Foundation Models based on the Transformer architecture.

Source: NVIDIA (2022) – [19]

Chosen Models and Task Overview

In this project, two distinct Transformer-based models were selected to address different classification scenarios (**Table C1**). This dual approach enables coverage of both **fixed-label** and **flexible-label** scenarios in multilingual, real-world social media data.

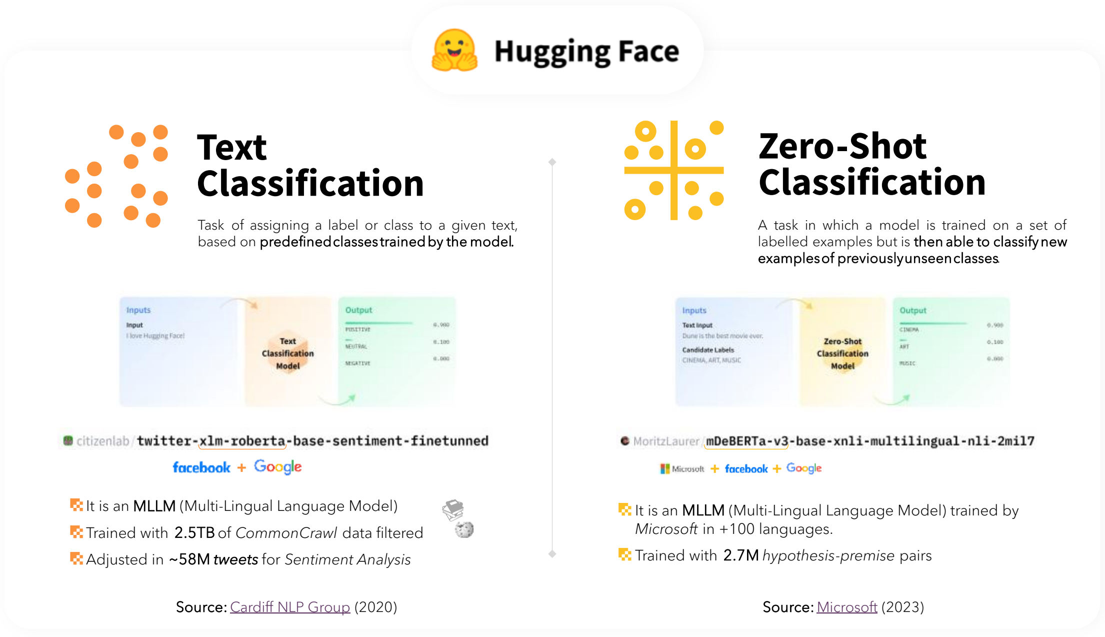
Table C1 – Transformers Models Chosen.

Task	Description	Model Name	Architecture	Notes
Text Analysis	Model is fine-tuned to classify inputs into predefined categories	citizenlab/twitter-xlm-roberta-base-sentiment-finetuned	*XLM-RoBERTa*	Fine-tuned for multilingual sentiment classification
Zero-Shot Classification	Model can assign labels it was never explicitly trained on, based on semantic similarity	MoritzLaurer/mDeBERTa-v3-base-xnli-multilingual-nli-2mil7	*mDeBERTa*	Multilingual model for zero-shot inference across diverse categories

Both models (**Figure C2**) were selected based on availability, performance, support for multilingual, and computational feasibility. **HuggingFace** was used as the model repository due to its free, open-source availability.

Task Overview and Model Deployment

Each model was integrated into a classification pipeline. CUDA acceleration was used during inference to improve performance on available NVIDIA GPU hardware (in this work available on the *Colab*).



Task of assigning a label or class to a given text, based on predefined classes trained by the model.

- It is an MLLM (Multi-Lingual Language Model)
- Trained with 2.5TB of *CommonCrawl* data filtered
- Adjusted in ~58M ***tweets*** for *Sentiment Analysis*

Source: Cardiff NLP Group (2020)

A task in which a model is trained on a set of labelled examples but is then able to classify new examples of previously unseen classes.

- It is an MLLM (Multi-Lingual Language Model) trained by *Microsoft* in +100 languages.
- Trained with 2.7M *hypothesis-premise* pairs

Source: Microsoft (2023)

Figure C2 – Overview of selected Transformer-based models for each task type: (left) sentiment classification using a multilingual fine-tuned model; (right) zero-shot topic classification using a multilingual inference model.

ANNEX D. PRINCIPAL COMPONET ANALYSIS (PCA)

PCA is a classical linear technique for **dimensionality reduction**. It transforms the original correlated variables into a new set of uncorrelated variables, called **principal components**, ordered by the amount of variance they capture from the data. The first few components typically retain most of the variability, making PCA useful for simplification, data compression, and noise reduction. Mathematically, PCA performs an eigen decomposition of the data covariance matrix [23], projecting data into a new orthogonal basis where the directions of greatest variance are prioritized.

Rules of Thumb for Selecting Principal Components

Several heuristic rules have been proposed to guide the selection of the optimal number of principal components to balance dimensionality reduction with information preservation.

1. **First Rule of Thumb – Cumulative Variance Threshold**

A common approach is to retain the smallest number of components that together explain a sufficiently large proportion of the total variance in the data. A typical threshold is **80% to 90%**.

- This rule assumes that most of the important structures in the data are captured within the first few components.
- Mathematically, one selects the smallest integer *k* such that the sum of the first *k* explained variance ratios satisfies:

$$\sum_{i=1}^{k} \frac{\lambda_i}{\sum_{j=1}^{p} \lambda_j} \geq 0.80$$

where λ_i is the variance (or eigenvalue) of the i −th principal component, and p is the total number of components.

2. **Second Rule of Thumb – Kaiser's Criterion**

According to Kaiser (1960), only components with **eigenvalues greater than 1** should be retained. This rule is particularly common when the data has been standardized (i.e., features have unit variance).

- The rationale is that a principal component should explain at least as much variance as an individual original variable (which has unit variance).
- Therefore, components with $\lambda_i > 1$ are considered meaningful, while those with lower eigenvalues may represent noise.

3. **Third Rule of Thumb – Scree Plot (Elbow Method)**

The Scree Plot is a graphical tool that plots the eigenvalues (or explained variance) of the components in descending order.

- The user visually inspects the plot to identify the **"elbow" point**, where the rate of decrease in eigenvalues slows down significantly.
- Components before this point capture the most important patterns in the data, while components after it tend to represent minor variations or noise.
- This method is subjective and may not always yield a clear decision point.

These rules serve as guidelines rather than strict criteria, with the final choice often involving trade-offs between **model simplicity**, **efficiency**, and **interpretability**, typically using multiple rules together.

Key Limitation of PCA

The main drawback is **loss of interpretability**. While PCA improves computational efficiency and may reduce overfitting, the principal components no longer correspond to the original variables, complicating understanding and actionable insights, especially where transparency is critical.

ANNEX E. CLUSTERING (K-MEANS)

K-Means [24] is one of the most widely used unsupervised clustering algorithms [25][26] due to its simplicity, efficiency, and scalability. The algorithm partitions the data into k distinct clusters in such a way that the **within-cluster variance** is minimized. Each cluster is represented by its centroid, and each data point is assigned to the nearest centroid using Euclidean distance. While computationally efficient, K-Means assumes clusters of spherical shape and equal size, which may not reflect real-world data distributions. Another limitation is the requirement to **predefine the number of clusters**, which is not always straightforward without prior domain knowledge.

Determining the Optimal Number of Clusters

To choose the appropriate number of clusters (k), two of the most common heuristics are the **Elbow Method** and the **Silhouette Coefficient**.

1. **Elbow Method (Inertia)**

   The **Elbow Method** involves plotting the **inertia**, also known as the **Within-Cluster Sum of Squares (WCSS)**, against different values of k. Inertia is defined as the total squared distance between each point and the centroid of its assigned cluster. The formula is:

$$\text{Inertia} = \sum_{j=1}^{C} \sum_{i=1}^{n_j} \|x_i - \mu_j\|^2$$

   Where:

   - C: total number of clusters
   - n_j: number of points in cluster j
   - x_i: point i in cluster j
   - μ_j: centroid of cluster j

   The optimal k is typically found at the **"elbow" point** of the plot, where the marginal gain in variance reduction starts to diminish. Adding more clusters beyond this point yields only slight improvements in explaining the data variability, suggesting a good trade-off between model complexity and performance.

2. **Silhouette Coefficient**

   The **Silhouette Coefficient** measures how well a data point fits into its assigned cluster compared to other clusters. It considers both **cohesion** (how close the point is to others in the same cluster) and **separation** (how far the point is from points in other clusters).

   For a single sample, the silhouette score s is given by:

$$s = \frac{b - a}{\max(a, b)}$$

   Where:

   - a: mean intra-cluster distance (cohesion)
   - b: mean nearest-cluster distance (separation)

Interpretation:

- $s \approx +1$: The point is well matched to its own cluster and poorly matched to neighboring clusters.
- $s \approx 0$: The point lies on the boundary between two clusters.
- $s < 0$: The point may have been assigned to the wrong cluster.

By computing the **average silhouette score** for all samples for each value of *k*, one can identify the number of clusters that yields the **best overall structure** in the data — generally the higher the average silhouette score, the better the clustering configuration.

Both the Elbow and Silhouette methods are useful and often applied in combination to validate the choice of *k*, though they are ultimately heuristics and not guaranteed to produce a definitive answer in all datasets.

ANNEX F. GRAPH ANALYSIS

To analyse structural patterns and relational importance within the Amazon review network, we applied graph algorithms using the *GraphFrames* package in *PySpark*. This annex explains the theoretical principles behind the two main algorithms used: **PageRank** and **Label Propagation**.

Unlike ***GraphX***, which is Spark’s original graph processing API built on RDDs (*Resilient Distributed Datasets*), ***GraphFrames*** offers a more flexible and expressive interface by building on top of *DataFrames*. *GraphFrames* also provides native support for SQL-like queries and seamless integration with *PySpark*, which was essential for our project, given that our entire data pipeline and processing logic were based on the *PySpark* ecosystem using *Python*.

PageRank

PageRank is an algorithm developed by Page and Brin (1999) to rank the importance of nodes in a directed graph. It simulates a **random walk** process, where at each step a user either:

- Follows an outgoing link from the current node, or
- "Teleports" to a random node in the graph, with a probability defined by a **reset probability** (commonly set to 0.15).

Each node receives a score that reflects its relative importance, based on both the quantity and quality of incoming links. In our case, we modelled the graph as directed from users to products. Thus, PageRank scores are mainly influenced by the in-degree of products, i.e., the number of reviews received. A high score indicates high visibility or popularity within the network.

Key characteristics of PageRank:

- **Input:** Directed graph
- **Key parameter:** *resetProbability* controls the teleportation rate
- **Importance factor: In-degree** (nodes with many incoming edges rank higher)
- **Output:** Numerical score per node
- **Application:** Identifies influential or central products

2. Label Propagation

Label Propagation Algorithm (LPA) is a method for **community detection** in large-scale graphs. It works by initially assigning each node a unique label. At each iteration, nodes update their label to the most frequent label among their neighbours. This process continues until labels stabilise or the maximum number of iterations (*maxIter*) is reached.

One of the strengths of LPA is that it does not require the number of communities to be specified in advance. Instead, communities emerge naturally from the label propagation process. However, the number of iterations indirectly influences the number of communities:

- **Fewer iterations** may result in more fragmented, smaller communities.
- **More iterations** allow broader label diffusion, potentially merging communities.

In *PySpark GraphFrames*, *maxIter* defines how many times this propagation step is repeated. LPA is non-deterministic due to its reliance on random initialisation and update order.

Key characteristics of Label Propagation:

- **Input:** Undirected or directed graph
- **Key parameter:** *maxIter* (indirectly affects community count)
- **Importance factor:** Neighbour labels
- **Output:** Community label per node
- **Application:** Identifies groups of nodes with dense internal connections

Comparison of Core Aspects

Feature	PageRank	Label Propagation
Goal	Identify important/influential nodes	Detect communities based on structure
Type of output	Numerical score per node	Cluster label per node
Requires graph direction	Yes	No (can be applied to undirected graphs)
Control over output size	No	Indirect via *maxIter*
Deterministic?	Yes (with same config)	No (non-deterministic)
Main influence factor	In-degree (user → product)	Neighbourhood structure