National College of Ireland

Leveraging BERT-GRU for Sentiment-Driven Demand Forecasting & Product Insight in E-Commerce

MSc Data Analytics Research Project

MSCDAD_B

Renjitha Methanath

Student ID : x23308532

School of Computing

National College of Ireland

Supervisor : Bharat Agarwal

National College of Ireland

MSc Project Submission Sheet

School of Computing

Student Name: ……Renjitha Methanath……

Student ID: ……x23308532……

Programme: ……MSc Data Analytics…… **Year:** ……2024-25……

Module: ……Research Practicum……

Supervisor: ……Bharat Agarwal……

Submission Due Date: ……11-08-2025……

Project Title: ……Leveraging BERT-GRU for Sentiment-Driven Demand Forecasting and Product Insight in E-Commerce……

Word Count: ……10,000…… **Page Count**……21……..

I hereby certify that the information contained in this (my submission) is information pertaining to research I conducted for this project. All information other than my own contribution will be fully referenced and listed in the relevant bibliography section at the rear of the project. ALL internet material must be referenced in the bibliography section. Students are required to use the Referencing Standard specified in the report template. To use other author's written or electronic work is illegal (plagiarism) and may result in disciplinary action.

Signature: ……Renjitha Methanath……

Date: ……10-08-2025……

PLEASE READ THE FOLLOWING INSTRUCTIONS AND CHECKLIST

Attach a completed copy of this sheet to each project (including multiple copies)	☑
Attach a Moodle submission receipt of the online project submission, to each project (including multiple copies).	☑
You must ensure that you retain a HARD COPY of the project, both for your own reference and in case a project is lost or mislaid. It is not sufficient to keep a copy on computer.	☑

Office Use Only	
Signature:	
Date:	
Penalty Applied (if applicable):	

Leveraging BERT-GRU for Sentiment-Driven Demand Forecasting and Product Insight in E-Commerce

Renjitha Methanath
23308532
MSCDAD_B
National College of Ireland

Abstract

As e-Commerce continues to expand, getting customer sentiment right has become an imperative requirement for both reliable demand forecasting and inventory management. Traditional methods of forecasting mostly use historical sales data or social media hints alone, with no consideration of the authenticity of reviews and with minimal contextual information. This paper develops a sentiment-based demand forecasting framework by combining a BERT-GRU deep learning model with current state-of-the-art time series forecasting techniques like SARIMAX. Based on supervised Amazon review data, the system identifies fine-grained sentiment, filters out low-quality or artificially created text, and links sentiment polarity, buyer ratings, and review trends to demand forecasting with greater accuracy. The fine-tuned BERT-GRU model achieved an impressive 95% accuracy with consistent performance across all classes. Furthermore, incorporating sentiment as an exogenous variable in SARIMAX yielded a significant improvement in model fit, reducing the AIC from 189 to 181. In closing the loop between actionable forecasting and sentiment insights, this approach addresses concerns like spam reviews and context mismatch, offering a scalable, real-world solution to smarter, optimized decision-making in e-commerce.

Keywords— Sentiment Analysis, BERT-GRU, Transformer, Review Authenticity, Demand Forecasting, SARIMAX

1 Introduction

In the rapidly evolving and highly competitive online shopping marketplace of today, vendors are getting swamped by the sheer volume of customer reviews, with millions of reviews pouring in every day. These reviews hold rich, actionable insights into what customers prefer, what infuriates them, and how products can be improved. But it is practically impossible for vendors to scan and interpret all reviews manually to find out the drivers of positive or negative views. The ability to understand this emotion is crucial not just to product refinement and addressing customer pain points but also to staying in business by constantly responding to changing consumer expectations.

At the same time, relying on strict historical sales records alone to forecast future demand is increasingly insufficient. Customer sentiment — specifically negative sentiments — can quickly influence purchasing decisions and cause sudden fluctuations in demand that do not appear in conventional forecasts. Adding sentiment from reviews to demand forecasting allows sellers to have a richer and more proactive view of how customers' attitudes directly influence future sales trends. Apart from that, with the rise of the incidents of AI-generated reviews and spam comments, there is an urgent need to classify the low-quality feedback to deliver only high-quality, human-generated feedback for business decision-making.

This gap in the state of the art motivates the primary research question of this work:

" To what extent can verified customer sentiment be effectively integrated into demand forecasting models to enhance prediction accuracy and support more informed inventory decisions in e-commerce? "

This research responds to this question by proposing a sentiment-based forecasting model that combines robust review authenticity filtering with a hybrid BERT-GRU deep learning model capable of extracting nuanced sentiment directly from Amazon product reviews. These validated sentiment scores are then used in advanced time-series forecasting models such as SARIMAX, in combination with other relevant variables such as customer recommendation rates, to produce accurate, context-sensitive forecasts. Through model benchmarking against both with and without sentiment inclusion, this study illustrates how businesses can move beyond static historical data and gain more reliable, actionable insight to make smarter product and inventory decisions.

This paper consists of a literature review, methodology, design specification, Implementation and evaluation of experiments carried out, discussion of findings, and finally, conclusions, limitations, and recommendations for further enhancements. The structure ensures a logical flow from background studies to practical and critical observations.

2 Literature Review

2.1 Sentiment analysis & its importance

It has become essential for companies that are looking to prosper in today's highly competitive e-commerce world to comprehend customer attitudes. Systematic analysis of user perceptions gives companies valuable information about how customers perceive a specific product directly, and they guide product design and strategic decision-making (Alzahrani et al., 2022). Unlike overall sales information, customer opinions frequently address several product characteristics including quality, price, design and report strengths and weaknesses that may go unnoticed otherwise (Obiedat et al., 2021). Having an in-depth understanding of this prescience enables firms to target areas for improvement, better focus product features, and remain competitive in a dynamic market (Cambria et al., 2017).

The internet now allows customers to freely leave reviews for products and services, providing valuable information to enable new consumers to make informed decisions while also offering companies immediate access to authentic user sentiment (Geetha and Renuka, 2021). This openness has obvious benefits for online business, including better delivery efficiency, increased customer loyalty, cost reduction, and improved profitability (Alamdari et al., 2020).

Essentially, sentiment analysis is a contextual text mining process to identify subjective knowledge from texts to enable companies to gain a deeper insight into the social sentiment of their product or service. Customers also leave their authentic experience and emotions in comments or reviews, which are very current "word of mouth." It can affect other potential customers' buying behavior (Sanyal and Wamique, 2019). Persuasive reviews build trust in products and companies, directly affecting conversion rates. Even so, the number of reviews can be too tedious for clients to go through, and there is a necessity to employ automated sentiment analysis techniques that are effective in extracting valuable insights (Chintalapudi et al., 2021;Iddrisu et al., 2023). Sentiment analysis helps

identify and examine the underlying emotions in customer feedback and yield valuable information to guide purchasing decisions and business strategy development (Savci and Das, 2023).

Sentiment analysis is usually studied at three main levels (Birjali et al., 2021). Document-Level Sentiment Analysis (DSA) analyzes the overall sentiment of an entire review or document to obtain the general emotional tone expressed by the writer (Hussein, 2018). Sentence-Level Sentiment Analysis (SSA) takes it further by looking at sentences on a one-by-one basis to see if specific statements convey positive or negative feelings, offering a more nuanced understanding of customer impressions (J. Zhang et al., 2021; Tubishat et al., 2021). The most fine-grained approach, Aspect-Level Sentiment Analysis (ASA), locates sentiments that relate to particular attributes or aspects of a product review. For instance, terms such as "love" in "I love the Amazon product" aptly exhibit positive emotions attached to a particular product feature (Tubishat et al., 2021; Ahmed et al., 2020). This degree of specificity is indispensable when trying to ascertain which product features appeal to customers and need to be changed.

Sentiment analysis is now an essential tool for e-commerce businesses that enables them to gain meaningful insights from massive amounts of user-generated content. It assists in ensuring that businesses can cope with changing customer demands, remain pertinent with products, and compete effectively in the online space.

2.2 Machine Learning for sentiment analysis

Early research by Pang et al., 2002 laid the groundwork for sentiment analysis using conventional machine learning models. In their seminal paper, they employed supervised classifiers such as Naive Bayes, Maximum Entropy, and Support Vector Machines (SVM) for use with the movie review dataset. Pang et al., while the first to employ these models, found that sentiment classification posed unique challenges to the typical text classification problem. The primary limitation came from the Bag of Words (BOW) approach, which does not consider grammatical structure and semantic context — features critical in capturing sentiment nuances. Their trials showed that while SVM marginally outperformed Naive Bayes and Maximum Entropy models, each of these models was incapable of identifying subtle sentiment cues that human beings intuitively register, foreshadowing that traditional feature representations intrinsically carry limitations in dealing with subjectivity.

Expanding on this, Jayakody et al., 2021 conducted sentiment analysis on actual social media data. On Twitter product reviews, the research compared different ML algorithms — Support Vector Machines, Logistic Regression, and k-Nearest Neighbors — with text vectorization techniques like Count Vectorizer and TF-IDF. The comparative result showed that Logistic Regression with Count Vectorizer had the best accuracy of 88.26%, demonstrating that the preprocessing pipeline and the vectorization approach used can greatly influence the performance of the model. This study highlights the importance of carefully optimizing the representation of unstructured text before using ML algorithms, particularly where there is noisy and informal language as seen on social media.

Working in a more comparative vein, Y. Zhang et al., 2024 provided an insightful comparison between traditional machine learning models — Logistic Regression, Random Forest, and Naive Bayes — and deep learning models such as CNNs and RNNs for Amazon product review prediction. The result was persuasive: Logistic Regression and Random Forest both achieved an impressive accuracy of 99%, surpassing even the deep networks, which scored slightly lower (RNN at 98% and CNN at 93%). This finding validates the claim that well-optimized ML models, especially ensemble algorithms like Random Forests, can rival more complex architectures, provided that the data are well-organized enough and the features well-designed. Additionally, this study highlighted the fact that lexicon-based pre-trained tools like NLTK and TextBlob lag behind supervised ML methods in accuracy since they consistently fall short on catching the contextual nuances and domain-tailored language patterns inherent in user content.

Lastly, Antony et al., 2025 survey gave an in-depth overview of the landscape not only over the usual ML classifiers like Naive Bayes and SVMs but also introduced their combination with lexicon-based methods and hybrid approaches. The ongoing challenges identified by Cruz were the identification of sarcasm and irony or the handling of domain-specialized vocabularies not optimized by standard models. The study verified that while classical ML models remain widely effective due to their simplicity and explicability, their predictive accuracy can be greatly enhanced by using them in conjunction with domain lexicons or semantic resources. The integration often bridges the gap between rule-based pattern recognition and statistical model predictive accuracy.

Study	**Models Used**	**Purpose**	**Key Results**
Pang et al. (2002)	Naive Bayes, Maximum Entropy, SVM with Bag-of-Words	Early exploration of ML for sentiment classification on movie reviews	SVM outperformed others slightly; Bag-of-Words failed to capture context and semantics.
Jayakody (2021)	SVM, Logistic Regression, KNN with Count Vectorizer and TF-IDF	Analyzing Twitter product reviews to find optimal ML–vectorization combo	Logistic Regression with Count Vectorizer achieved 88.26% accuracy.
Yan Zhang (2024)	Logistic Regression, Random Forest, Naive Bayes, CNN, RNN	Comparative study on Amazon reviews using ML vs DL	Random Forest & Logistic Regression scored 99%, outperforming RNN (98%) and CNN (93%).
Cruz Antony J (2025)	Naive Bayes, SVM, hybrid with lexicon-based methods	Survey of sentiment analysis methods & challenges (sarcasm, irony, domain-specific vocabularies)	Highlighted hybrid models enhance accuracy and contextual understanding.

Table 1: Summary of Studies on ML Approaches for Sentiment Analysis

Together, these works collectively explain how machine learning techniques — from initial Naive Bayes and SVM usage to more recent ensemble approaches — are still a crucial part of sentiment analysis in a diverse range of applications. While these approaches have proven to be dependable, especially when carefully adjusted and paired with proper text representation techniques, they also highlight ongoing shortcomings, most significantly in the capacity to interpret context, nuance, and mixed sentiment. These results present a clear course of action for future research to create more context-aware models or combine ML with more profound semantic comprehension to better detect sentiment.

2.3 Hybrid Deep Learning for Sentiment Analysis

More recent studies have expanded the horizon for sentiment analysis by investigating a range of cutting-edge deep learning and hybrid methodologies, each addressing weaknesses in current approaches alongside application-oriented issues.

Chen, 2024 proposes a new model that fuses a multiscale Convolutional Neural Network (CNN) and a Bidirectional Long Short-Term Memory (BiLSTM) network to better examine sentiment expressed on Weibo, one of China's largest microblogging sites. Their method obtains local and global features by multiscale CNN kernels and models long-distance text relationships by BiLSTM. In their experiments, the MSCNN-BiLSTM ensemble worked better than the traditional CNN and LSTM individually, with F1 scores of 80% against 77% for CNN and 76% for BiLSTM individually. This shows the extent to which hybridization of BiLSTM and CNN models has come a long way in unraveling complex emotional content, especially with respect to short-text social media data, which is unconstrained.

Nimmi et al., 2022 introduced the AVEDL (Average Voting Ensemble Deep Learning) model, which utilized multiple transformer-based models, namely BERT, DistilBERT, and RoBERTa, to undertake sentiment analysis of COVID-19 emergency response calls. This ensemble voting strategy yielded a robust Macro-average F1-score of 85.2% and accuracy of 86.46%, outperforming many standard machine learning baselines. The model nevertheless failed to fully address colloquial language and speech nuances, proving that context-specific slang and speech behaviors remain a task yet to be tackled in real-world deployment.

Sirisha and Bolem, 2022 took it a step further by combining Aspect-Based Sentiment Analysis (ABSA) with RoBERTa and LSTM layers using Twitter data for the war in Ukraine. The outcome with this approach was a whopping 94.7% accuracy, which shows how combining transformers with sequential deep models is more capable of recognizing fine-grained sentiment features in volatile settings.

Bansal and Srivastava, 2019 developed an attribute-based hybrid method that exploits POS tagging to identify more specific consumer insights with lower computational requirements. Although the suggested method demonstrated potential, the work needed more generalized application examples aside from short text or hand-labeled lexicons to retain scalability and automation. Other researchers have also experimented with hybrid architectures for better semantic understanding. For example, AlBadani et al., 2022) offered a new ULMFit-SVM hybrid approach that combined Universal Language Model Fine-tuning with SVM, which achieved notable accuracies — 99.78% for Twitter US Airlines, 99.71% for IMDB, and 95.78% for GOP debate data. The sentiment recognition was only up to document-level granularity, not considering aspect-level analysis.

Assiri and Gumaei, 2024's study, which was the foundation for our project, showcased a novel integration of DeBERTa (Decoding-enhanced BERT) with GRU layers, leveraging the strengths of both transformer parallelism and sequential models to better capture long-distance semantic dependencies. Tested on the Twitter LLM dataset, this hybrid model reached an F1-score of 97%, setting a new benchmark for combining parallelized attention mechanisms with efficient recurrent units.

Study	**Models Used**	**Purpose**	**Key Results**
Yan Chen & Hong (2024)	Multiscale CNN + BiLSTM	Sentiment analysis of Weibo posts capturing local & global features	MSCNN-BiLSTM achieved 80% F1, outperforming standalone CNN (77%) and BiLSTM (76%).
Nimmi (2022)	AVEDL ensemble (BERT, DistilBERT, RoBERTa)	Sentiment analysis of emergency response calls during COVID-19	Achieved 86.46% accuracy and F1-score of 85.2%; struggled with slang and speech nuances.
Uddagiri (2022)	RoBERTa + LSTM + Aspect-Based Sentiment Analysis (ABSA)	Sentiment analysis on Ukraine conflict tweets with aspect focus	Accuracy reached 94.7%, showing effectiveness in fine-grained, context-sensitive tasks.
Bansal (2019)	Attribute-based hybrid (POS tags, feature selection)	Consumer intelligence analysis with low computation cost	Good accuracy but lacked scalability; required manual labeling and short-text focus.
Barakat (2022)	ULMFit + SVM	Hybrid document-level sentiment analysis on Twitter Airlines, IMDB, GOP debate	Achieved 99.78%, 99.71%, and 95.78% accuracy respectively; limited to document-level.
Adel Assiri (2024)	DeBERTa + GRU	Twitter sentiment analysis capturing long-distance semantic dependencies	Achieved 97% F1-score, demonstrating the power of combining transformers with recurrent layers.

Table 2: Summary of Studies on Hybrid DL Approaches for Sentiment Analysis

Together, these studies reflect the expanding boundaries of hybrid and deep models in sentiment analysis, and the fact that even though transformer-based models like BERT and RoBERTa are always better than the earlier models, still, models that can handle slang, multilingual inputs, real-time inference, and aspect-based nuances in different application contexts are still needed.

2.4 Applications for Sentiment Analysis

Recent research shows that sentiment analysis is no longer just a text mining technique but also an actionable input for uses like demand forecasting, trend spotting, and financial prediction. Singh et al., 2024, for example, proposed an AI-powered multi-modal framework that integrates traditional time series models like ARIMA and Exponential Smoothing with advanced machine learning models like LSTM, Random Forest, and XGBoost. What is novel with their approach is the integration of real-time social media sentiment on Twitter and Reddit with lexicon-based (VADER) and BERT-based classifiers. Integrating these sentiment signals with historical sales data consistently reduced prediction errors by 20–30% when compared to conventional ARIMA models, demonstrating the potential for consumer sentiment as an early signal for demand change, especially in emotionally driven or high-movement products. Yet, issues persist regarding handling noisy, high-volume text data and ensuring that certain sentiment signals are normalized to the same level across platforms and categories.

Along the same vein, Darraz, 2023 demonstrated the integration of customer review-based sentiment scores with trend detection for use in informing product strategy in online retailing. Coupling VADER sentiment scores with decision tree classifiers and cluster analysis,

the study was able to predict likely bestsellers to 93% accuracy, often beating sales data in detecting the trend. Such integration of NLP-driven opinion mining with predictive analytics allows firms to pre-emptively manage inventory and marketing, although the static lexicon of VADER and simplicity of decision trees may limit sensitivity to subtle, domain-specific opinions, suturing a requirement for domain-tailored or hybrid models.

In the financial domain, Akhtar et al., 2017 suggested an ensemble model that integrates CNN, LSTM, and GRU networks trained on financial news headlines and Twitter messages with domain-specific embeddings and lexicon features. The sentiment representation is fed into a Support Vector Regression (SVR) layer to forecast market trends and improve current state-of-the-art models by up to 4.1 points on the SemEval-2017 benchmark. This work illustrates that complex, aspect-based sentiment indicators can improve predictions of stock motion. However, the computational costs and use of continuously updated domain-dependent dictionaries required by the ensemble model raise questions about its suitability for high-frequency trading environments where speed and interpretability are critical.

Study	**Models Used**	**Purpose**	**Key Results**
Singh et al. (2024)	ARIMA, Exponential Smoothing, LSTM, Random Forest, XGBoost with BERT sentiment	Multi-modal demand forecasting integrating sentiment with historical sales	Sentiment integration reduced forecast errors by 20–30%; LSTM with sentiment performed best.
Darraz (2022)	VADER sentiment + Decision Tree + Clustering	Trend detection from customer reviews to identify potential bestsellers	Achieved 93% accuracy in predicting trending products; limited by VADER's static lexicon.
Md Shad Akhtar (2022)	CNN, LSTM, GRU ensemble + Support Vector Regression	Financial sentiment analysis using news & Twitter data	Outperformed state-of-the-art by 2–4.1 points; high computational cost limits real-time use.

Table 3: Summary of Studies on Application-Level Sentiment Analysis

All together, these studies record a strong trend: sentiment analysis, when paired with robust time series and machine learning infrastructure, can fundamentally improve forecast power across domains. They further emphasize that the practical success of such systems depends upon surmounting challenges like data preprocessing at scale, context-dependent sentiment disambiguation, computational efficiency, and balancing model complexity with interpretability. Follow-up research in this field can focus on advancing multi-modal and cross-lingual sentiment fusion, domain-specific lexicon building, as well as explainable AI to ensure that sentiment signals not only deliver accuracy but actionable insights to decision-makers.

2.5 Addressing Literature Gaps: Our Proposed Framework

Although several have tried sentiment analysis from traditional machine learning models, they come at the cost of much richer contextual relationships and more nuanced senses in text. For example, Pang et al., 2002 and Jayakody et al., 2021 demonstrated the application of conventional classifiers like SVM, Naive Bayes, and Logistic Regression with bag-of-words or TF-IDF representation without word order or semantic context, which leads to less comprehension of complex sentiments like sarcasm or fine-grained emotion transitions. Although models like Logistic Regression or Random Forest performed well in Chen, 2024's comparison regarding accuracy, they could not comprehend context-dependent sentiment expressions entirely. Likewise, Antony et al., 2025's review shows that lexicon-based and traditional ML methods on their own cannot match up against challenges such as domain-specific language or irony. On the other hand, while more recent work has used deeper models like BERT-GRU (Assiri and Gumaei, 2024) to capture more context, these methods overlooked the problem of removing fake or AI-generated reviews and exposing the models to manipulated sentiment signals. Also, in Singh et al., 2024's work on demand forecasting, sentiment was successfully integrated with economic indicators, but their adoption of VADER, a lexicon-based method, doesn't allow it to catch sarcasm and compound phrases. By combining a state-of-the-art BERT-GRU model for sophisticated contextual sentiment detection with a backup AI fake-review filter, our project addresses these weaknesses head-on — not just ensuring that sentiment is informative and well-captured, but that only genuine user-generated content is used to train downstream tasks such as demand forecasting. In this way, we transcend the bounds of conventional ML sentiment models and previous integration approaches like Singh's and present a better, context-aware, and viable solution for practical implementation.

3 Research Methodology

3.1 Data Gathering & Initial Cleaning

The dataset for this project was obtained from Kaggle's publicly accessible product review collection of Amazon, which consists of 30K product Reviews. The review data underwent a preliminary extraction to locate its parts. During this step, only necessary columns like the product ID, review, rating, and recommendation flag were retained to ensure that the analysis was not diluted by extraneous factors. A rudimentary form of data preprocessing was applied to the dataset in terms of basic cleanup to remove offending entries and irrelevant data that did not conform to the desired structure and format.

3.2 Sentiment Labeling & Class Imbalance Analysis

Sentiment labels had to be created to enable supervised learning because the raw dataset did not have predefined categories of sentiments. A simple rule-based approach was employed: positive sentiment was ascribed to a rating over 4, a score of 3 was classified as neutral, and negative sentiment was attributed to scores under 3. This provided the target variable necessary for training the baseline logistic regression model. Nonetheless, this approach has its shortcomings since customer ratings are not always in tandem with written reviews—users may rate a comment highly while providing a low score, or the other way around. Regardless of this constraint, in this case, rule-based labeling was enough to create an initial model and then refine the understanding of the dataset's behavior. Any introduction of model inefficiencies at this stage is later remedied with a BERT-GRU model that captures true contextual sentiment more accurately. An analysis of the distribution of classes was done to identify any imbalances, and this information was helpful in deciding how to deal with disproportionate representation of certain classes during training.

3.3 Text Preprocessing & Feature Engineering

To train the baseline model, an elaborate text preprocessing pipeline was developed. This was important to remove noise, which could obscure meaningful patterns. This step cleaned reviews by addressing non-alphabetic characters, redundant formats, alongside words that do not add significant meaning.

The cleaning process involved the following steps.

- **Removing non-alphabetic characters**: Remove punctuation, numbers and special signs which redirect attention from words, letters and meanings.
- **Lowercasing text**: Uniformity in letter casing is crucial for words. All forms should be treated equally, hence converted to lower case forms.
- **Stopword removal**: Common words such as "and", "the", and "is" were removed using the NLTK stopword list to reduce noise and improve focus.
- **Lemmatization**: Every word has a dictionary or base version for all phrases. "Running" in this context is transformed to "Run".
- **Stemming**: Further trims words to their root by removing suffixes ("Player" or "Play"). Applying both lemmatization and stemming powerfully clears many word variations.

Once cleaned, the next step was feature engineering using TF-IDF (Term Frequency–Inverse Document Frequency), which transforms text into numerical features that a machine learning model can use. The process works as follows:

- **Term Frequency (TF)**: Measures how frequently a word appears in a single document relative to the total number of words in that document.

$$TF(t,d) = \frac{\text{Number of times term } t \text{ appears in document } d}{\text{Total number of terms in document } d} \tag{1}$$

- **Inverse Document Frequency (IDF)**: Measures how unique or informative a word is across all documents. Common words that appear in almost every document get a low IDF score, while rare, meaningful words get higher scores.

$$IDF(t) = \log\left(\frac{N}{n_t}\right) \tag{2}$$

  where:

  - N = total number of documents
  - n_t = number of documents containing term t

- **TF-IDF Score**: For each word, the TF and IDF are multiplied to produce a final score that balances local importance with global uniqueness.

$$TF\text{-}IDF(t,d) = TF(t,d) \times IDF(t) \tag{3}$$

This is how the practical implementation goes:

- **Tokenization**: Every review is separated into words (tokens).
- **Vocabulary building**: All unique tokens from the training set are evaluated, and the top 5000 tokens with the most significant TF-IDF scores are retained. This limit enhances computational efficiency while concentrating on the most informative words.
- **Vectorization**: Each review is converted into a numerical vector. If a word from the review is present in the top 5000 vocabulary, its corresponding index in the vector is set to its TF-IDF score; otherwise, it is omitted.
- **Fit and transform**: A TF-IDF vectorizer fits on the training set to learn the vocabulary and IDF values and then is used to transform both training and test sets. This prevents test data from leaking into the training process.

With this, the thorough text preprocessing and feature engineering pipeline refined the raw input data, allowing the baseline logistic regression model to learn meaningful patterns in the reviews' sentiments.

3.4 Baseline Model: Logistic Regression

To define a baseline and sanity-check the effectiveness of the rating-based sentiment labels, a baseline model was implemented with Logistic Regression. The data was split into training and testing sets with an 80-20 division, which ensured that the model was evaluated on unseen data for generalization assessment. Despite its name, Logistic Regression is a classification algorithm that performs remarkably well for multi-class problems, given that it is extended with a softmax function. As with any machine learning algorithm, the model has a beginning stage of initialization where it assigns random weights to each input feature. For each input review represented with a TF-IDF feature vector, the model computes a weighted sum to apply softmax over the weighted sum to turn it into a probability distribution over the three sentiment classes (positive, neutral and negative). The softmax function guarantees that all predicted probabilities across the classes will sum to one.

The model during training computes a loss based on the predicted probabilities and the actual class labels. An incorrect prediction prompts the model to adjust its weights and biases with gradient descent to reduce this loss, step by step learning the optimal words or features for each sentiment class. To ensure convergence in this case, the cap was set to a maximum of 1000 iterations. This max_iter parameter determines the total number of full passes the model attempts to process the training dataset while optimizing the weights towards a favorable configuration.

Analyzing the output from this baseline logistic regression approach allows us to ascertain whether the simple signal rating-based sentiment labels suffice for predicting sentiment with reasonable accuracy. This validation is crucial before advancing to the more advanced BERT-GRU model designed to capture deeper contextual interrelations within the text.

3.5 AI Content Filtering for BERT-GRU Input

Before training the central BERT-GRU sentiment classification model, an interim manual revision of reviews was carried out to omit low-quality reviews and those that appeared to have been generated by an AI algorithm. This type of review-cleansing AI algorithm was created using heuristics and was based on the following practical criteria:

- **Short Length**: Reviews of five words or fewer were flagged since such sentiments are too shallow and do not capture meaningful evaluation.
- **Overused Marketing Phrases**: Reviews that included phrases "highly recommend" or "great product", which are generic in nature and repeated many times, were flagged.
- **Low Word Uniqueness Ratio**: A ratio of unique words to total words was calculated for each review. If this ratio were below 0.3, then the review was flagged. This helps detect spam or bot-generated content, which often unnaturally repeats words.

By applying these criteria, the reviews in question were omitted from the training dataset to make sure that only high-quality, human-like text entered the BERT-GRU model, improving its ability to learn the true contextual sentiment patterns.

3.6 Hybrid BERT-GRU Model Building

For our main deep learning pipeline, we chose to implement BERT (Bidirectional Encoder Representations from Transformers) owing to its highly documented capacity for identifying contextual meaning in text. Contrary to normal text preprocessing, BERT performs best when utilized on natural, unperturbed text. This is because stemming or extensive manual token cleaning distorts word meanings and causes the pretrained embeddings to misinterpret precious context. Therefore, after passing the reviews through the AI-content filtering step reviews were then passed straight into the tokenizer of BERT. We utilized the bert-base-uncased tokenizer so that it's consistent and case-normalized. The labels and reviews were first translated into Python lists since BERT's tokenizer and PyTorch handle list-like input structures. The sentiment labels were then labeled numerically as 0, 1, or 2 so PyTorch would be able to calculate loss values properly during training.

The tokenizer performs several things simultaneously:

- **Tokenization**: Splits text into sub-word items (tokens) that may be fed to BERT.
- **Inserting Special Tokens**: Adds [CLS] at the beginning for classification issues and [SEP] for segment separation marking.
- **Padding & Truncation**: It ensures all input sequences are of equal length by inserting [PAD] tokens or cutting down on excessive-length sequences. Here, max_length=128 was employed to make the maximum input 128 tokens.
- **Attention Mask**: A binary mask that indicates what tokens are actual input and what tokens are padding. This way, BERT will focus its attention mechanism on the right places.

Since PyTorch needs input data to be wrapped in a specific form to batch and shuffle effectively, a ReviewDataset() class was established.

- The __init__ function stores input tensors (input_ids, attention_mask) and labeled encodings.
- __len__ provides the number of total samples.
- __getitem__ provides one data point at a time as a dictionary so that PyTorch's DataLoader can handle batches dynamically.

After creating the dataset, it was split into 80% training and 20% validation to check the generalizability of models. A DataLoader was used with a batch size of 16, shuffling training data every epoch to mitigate overfitting.

The core for sentiment classification is implemented as a custom BERT_GRU_Classifier() class that inherits from torch.nn.Module. With a custom class in such a manner offers a modular and versatile architecture: it enables us to stack pretrained parts such as BERT with extra layers (GRU, dropout, and a classifier) with all the handy aspects of PyTorch's neural network API being reused, including simple parameter management, automatic differentiation, and tidy forward propagation semantics. Deriving from nn.Module guarantees that all subcomponent parameters are registered and updated automatically during training.

At the heart of this system is the pretrained BERT model, bert-base-uncased, which has 12 transformer layers. BERT tokenizes a review into single tokens and gives each token a 768-dimensional embedding. The embeddings capture not only the word meaning but also its position within the sentence and its segment information. From BERT, we only take the last hidden state output, which provides contextualized representations for each token based on the deep transformer layers.

These BERT embeddings are input into a bidirectional GRU (Gated Recurrent Unit) layer. This GRU is dimensioned with input size=768 (in parallel with BERT's output size), hidden size=128, so that both sides of the GRU will learn 128 features, thus a total 256-dimensional output for each token. Using a bidirectional GRU, the model can look back as well as forward along the sequence and pick up on dependencies not picked up using one direction alone. Within, the GRU uses reset gates and update gates to control information flow: the reset gate determines how much older data to discard, and the update gate controls how much new data to add. This gating action helps GRU learning long-term relationships without suffering from vanishing gradients.

Subsequently, a dropout layer of rate 0.3 is employed to add regularization. Dropout shuts down some of the neurons randomly during training, which does not make the model overly reliant on any one feature and places a lower risk of overfitting. Finally, a linear fully connected classification layer maps GRU's 256-dimensional output to three class scores for positive, neutral, and negative sentiment. These raw scores are converted into class probabilities through a softmax function, so that the output probabilities are summed to one.

Mathematically, for each class c:

$$P(y = c|x) = \frac{\exp(z_c)}{\sum_j \exp(z_j)} \tag{4}$$

Where:
z_c is the raw score (logit) for class c.
The denominator sums the exponentials of logits for all classes to normalize the output into valid probabilities.

One major design decision here was to initially freeze the weights of BERT by making requires_grad=False on its parameters. This means that BERT only does nothing but act as a feature extractor during the first training phase, significantly hampering training time and computation. Once the GRU and classifier layers had learned enough to make sense of BERT's embeddings, the BERT layers

were also unfrozen so that fine-tuning could happen. That is, their weights were made trainable once again so that the entire network — BERT, GRU, and classifiers could learn together for optimal performance in the task of sentiment classification.

As the dataset was class-imbalanced, class weights were computed using compute_class_weight() and passed on to the CrossEntropyLoss function. Cross-entropy measures the difference between the predicted probability distribution generated by the model and the real distribution. The weighted version penalizes underclass misclassifications more harshly, encouraging balanced prediction.

The cross-entropy loss for a single prediction is given by:

$$L = -\sum_{c} y_c \log(\hat{y}_c) \tag{5}$$

Where:
y_c is the true label (1 for the correct class, 0 otherwise).
$\hat{y}_c$ is the predicted probability for class c from the softmax function.

The model uses the AdamW optimizer, which is a modification of the Adam optimizer with weight decay to avoid overfitting using penalization of large weights. Optimizers play a crucial role during training as they use parameters of the model to minimize the loss function. AdamW uses the benefits of adaptive learning rates (2e-5) and momentum while providing improved generalization compared to the standard Adam.

Briefly, the pipeline is: raw text is tokenized by BERT, which splits it into tokens and converts each token to a 768-dimensional vector. The embeddings pass through a bidirectional GRU that encodes sequential dependencies in both directions to yield a dense 256-dimensional representation per sequence. The representation is regularized through dropout, and raw class scores are generated by the dense layer. The CrossEntropyLoss function then computes how much the predictions are away from the true labels and calculates the error. The optimizer, in backpropagation, computes the gradients and updates the model weights: GRU and classifier weights are always updated, and after being unfrozen, BERT's weights are also updated. It enables the network to learn the optimal token patterns to represent positive, neutral, or negative sentiment. The BERT-GRU was trained on 3 epochs, which found a balance between sufficient learning and computational affordability given the dataset size and limited resources. Training and validation loss monitoring helped to prevent overfitting of the model across these epochs.

This hybrid architecture — with deep transformers, recurrence structure, and robust regularization — ensures that the model creates rich, context-dependent representations to make high-quality sentiment predictions, directly feeding into more precise downstream demand forecasting.

Component	**Configuration**
Input	Tokenized Amazon review text
Embedding Layer	Pre-trained `bert-base-uncased` (fine-tuned)
Sequence Modeling	Bidirectional GRU
GRU Hidden Size	128 units
Dropout	0.3
Output Layer	Fully connected layer with Softmax activation (3 classes)
Optimizer	AdamW
Learning Rate	2e-5
Loss Function	Weighted Cross-Entropy
Batch Size	16
Epochs	3
Train-Validation Split	80:20

Table 4: BERT-GRU Classification Model Configuration

3.7 Demand Forecasting Methodology

Upon finishing the sentiment classification step, the next step was to develop an appropriate demand forecasting model. As a beginning, the original data was preprocessed again by converting review dates to a valid datetime format and dropping rows with missing values to ensure data consistency. The reviews were also aggregated monthly by creating a year_month feature and aggregating by ASIN and this monthly interval to calculate the reviews per product per month. ASINs with at least six months of continuous review history were chosen based on the total combined data, such that the selected time series would represent the period to forecast. One qualifying ASIN was then chosen to provide an example of the time series pipeline modeling.

Before any time-series model estimation, stationarity needed to be determined by the Augmented Dickey-Fuller (ADF) test. A stationary series is a series with the same statistical properties over time, which is a requirement for ARIMA-family models to give reliable forecasts. If the p-value is less than 0.05, then the null hypothesis for a unit root is rejected, and stationarity can be assumed. The test was utilized to check if differencing was necessary or not.

For finding appropriate parameters for ARIMA, Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots were checked. ACF helps observe how strongly every lag of the variable correlates with present values, and for determining the MA (moving average) term, whereas the PAC emphasizes the immediate impact of historical lags, dictating the selection of the AR (autoregressive) term. These plots dictated the set of ARIMA specifications that were tested.

A baseline ARIMA was initially built using solely the past review numbers for demand prediction. Different orders (p,d,q) for ARIMA were attempted to establish the best fit. To incorporate additional data, the method was extended to SARIMAX models. The rate of recommendation by customers was incorporated as an exogenous variable first. The model was then capable of accounting for the immediate effect of positive customer word-of-mouth on future demand. Finally, sentiment scores generated by the BERT-GRU classifier were aggregated as another exogenous variable along with recommendation rates. Aggregating sentiment was to check whether incorporating proven customer opinion trends would enhance prediction accuracy as much as depending on sales data only. By comparing the outputs, the effect of sentiment and recommendations in enhancing prediction performance could be readily demonstrated.

3.8 Evaluation Methodology

3.8.1 Classification Model Evaluation

To comprehensively analyze the performance of both the baseline Logistic Regression model and the BERT-GRU sentiment classifier, a range of common classification metrics that are widely utilized were used: Accuracy, Precision, Recall, F1 Score, Cohen's Kappa, and Confusion Matrix.

- **Accuracy** is the number of correct predictions divided by all that were made. It offers a rapid overview of how often the model is accurate.

$$\text{Accuracy} = \frac{TP + TN}{TP + TN + FP + FN} \tag{6}$$

- **Precision** quantifies the number of samples that were predicted as a particular class and were indeed correct. It is very important in the case of class imbalance.

$$\text{Precision} = \frac{TP}{TP + FP} \tag{7}$$

- **Recall** (Sensitivity) computes how many actual samples of a particular class were correctly identified. High recall is very important when losing true instances is costly.

$$\text{Recall} = \frac{TP}{TP + FN} \tag{8}$$

- **F1 Score** is the harmonic mean of Precision and Recall. It balances both measures, providing a single score that considers false positives and false negatives.

$$\text{F1} = 2 \times \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \tag{9}$$

- **Cohen's Kappa** measures the consistency between predicted labels and true labels after chance agreement correction. It is highly suitable for imbalanced multi-class problems.

$$\kappa = \frac{p_o - p_e}{1 - p_e} \tag{10}$$

- **Confusion Matrix** gives a thorough breakdown of correct and incorrect predictions by displaying counts for every combination of predicted and true labels. This enables the computation of all the metrics below and provides a clear indication of which classes are confused.

3.8.2 Forecasting Model Evaluation

For the demand forecast models (ARIMA, SARIMAX), attention was paid to how well the model fits and forecasts the time series. They were assessed using:

- **Root Mean Squared Error (RMSE)** computes the square root of the mean squared difference between actual and predicted values. More severe errors are penalized more, and it is sensitive to outliers.

$$\text{RMSE} = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2} \tag{11}$$

- **Mean Absolute Error (MAE)** gives the mean of absolute differences between predicted and actual values. It is easy to interpret since it is the average prediction error in the same units as the data.

$$\text{MAE} = \frac{1}{n}\sum_{i=1}^{n}|y_i - \hat{y}_i| \tag{12}$$

- **Akaike Information Criterion (AIC)** is a model selection metric that simultaneously balances goodness-of-fit and model complexity. A lower AIC suggests a better model when comparing multiple competing models.

$$\text{AIC} = 2k - 2\ln(L) \tag{13}$$

Where:
k=number of parameters; L=likelihood of the model

4 Design Specification

This project's end-to-end sentiment-driven demand forecasting system is constructed based on a stacked integration of libraries, tools, and system resources to enable a reproducible and efficient process from data gathering to end forecasting insights.

4.1 Data Source & Storage Requirements

- **Dataset Source:** The Amazon product review dataset was accessed via Kaggle. Kaggle access and an appropriate account are prerequisites.
- **Storage:** Data is stored and processed locally on disk. There is a directory structure for raw data, processed outputs, model checkpoints, and results.
- **Compute:** All compute are executed on a local machine with CPU or GPU (if available) via PyTorch's device management.

4.2 Software & Notebook Environment

- **IDE:** The entire pipeline was authored in Jupyter Notebook, which supports step-by-step debugging, visualization, and iteration.
- **Environment:** Virtual environment or Conda with Python 3.x is recommended to handle dependencies.

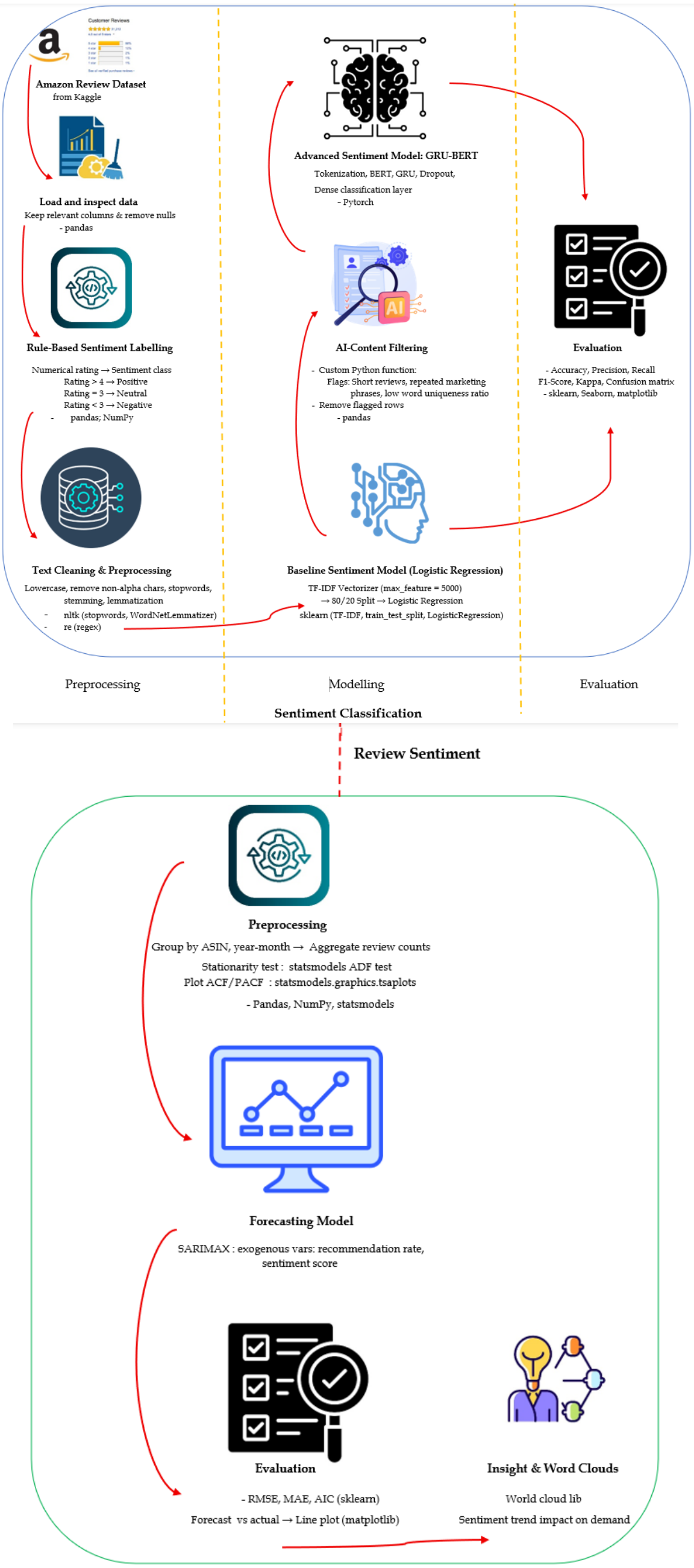

Figure 1: Architecture Diagram

4.3 Local System Requirements

- **Hardware:** Minimum 8 GB RAM (16+ GB recommended), with optional GPU (NVIDIA CUDA) support for BERT fine-tuning.
- **Disk Storage:** Sufficient space to store raw data, intermediate files, and model outputs.

Category	Libraries/Tools	Functionality
General Purpose & Data Handling	numpy, pandas	Numerical computation, data manipulation, and loading datasets (e.g., from Kaggle).
Text Preprocessing	nltk, re	Stopword removal, stemming, lemmatization, and text pattern matching using regular expressions.
Vectorization & Baseline Modeling	scikit-learn	Feature extraction with TfidfVectorizer, label encoding, class weight balancing, and baseline models like Logistic Regression.
Deep Learning for Sentiment Analysis	torch, transformers	Custom BERT-GRU model built using torch.nn, BertTokenizer, and BertModel; optimizer: Adam; loss function: weighted CrossEntropyLoss.
Time Series Forecasting	Statsmodels	ARIMA/SARIMAX modeling, ADF Test (adfuller), and autocorrelation analysis (plot_acf, plot_pacf).
Visualization & Insights	matplotlib, seaborn, wordcloud	Data trend plotting and word cloud generation for sentiment-based insights.
Evaluation Metrics	sklearn.metrics	Classification: Accuracy, Precision, Recall, F1-Score, Cohen Kappa. Forecasting: MAE, MSE, MAPE.

Table 5: Tools & Libraries required for the implementation

5 Implementation

This explains the final, step-by-step implementation that was adopted to develop the solution that worked for sentiment-based demand forecasting, from data cleaning through to sentiment classification with the BERT-GRU model and up to the union of sentiment and recommendation rates within the SARIMAX demand forecasting model.

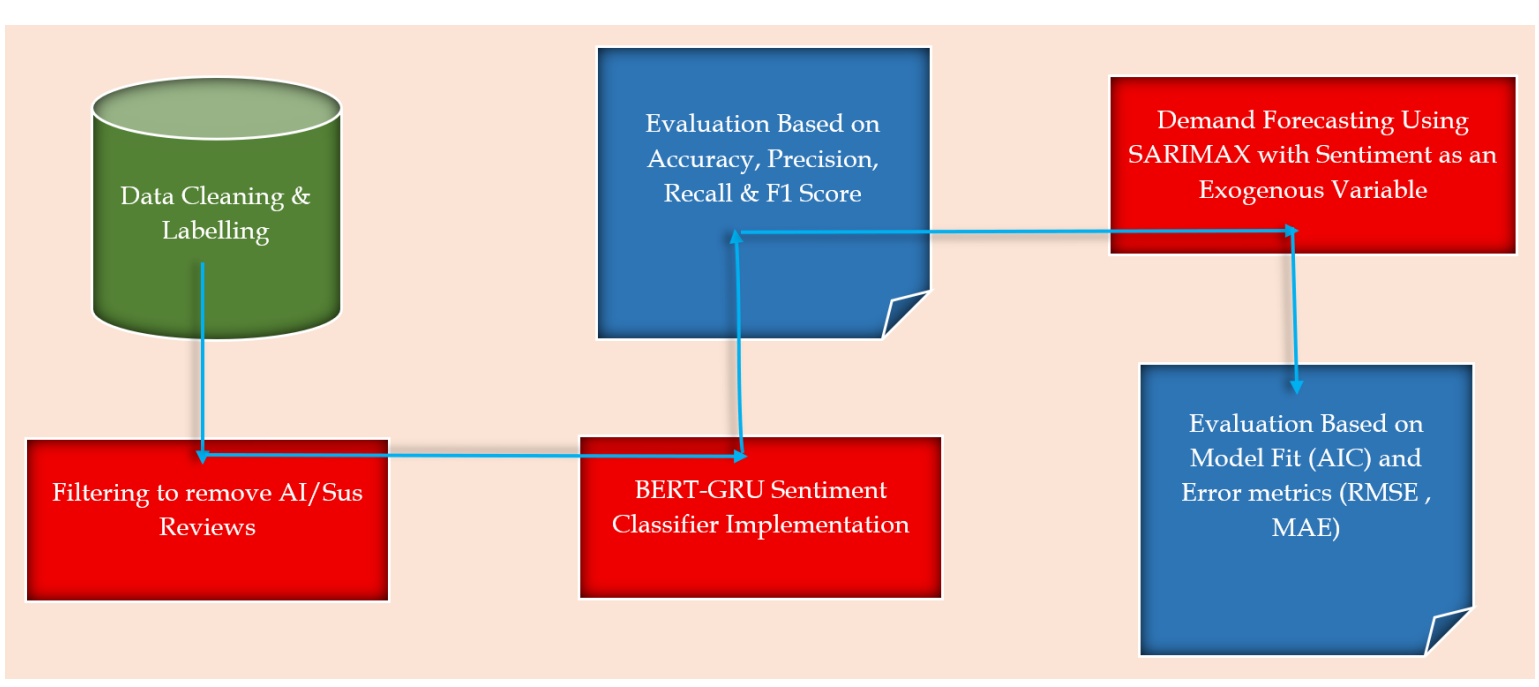

Figure 2: Process Flow for the Implementation of the Final Model

5.1 About the Data

Source: Kaggle – Amazon Product Review Dataset (link)

Resolution: Contains 39,660 records and 21 columns.

Time Range: Reviews span from *July 9, 2010* to *April 18, 2018*.

Sampling Rate: Data is recorded daily (though not perfectly uniform).

Geographical Location: Fields for user location (`userCity`, `userProvince`) are completely empty, so no geographical insights were used.

Missing / Erroneous Data Handling

High missingness was observed in the following fields:

- `reviews.userCity` & `reviews.userProvince` – 100% missing
- `reviews.dateAdded` – 39.4% missing
- `reviews.date` – 12.7% missing
- `reviews.doRecommend`, `reviews.numHelpful`, `reviews.title` – 14%–17% missing

```
 'missing_values': id                      5000
 name                    6760
 asins                      2
 brand                   5000
 categories              5000
 keys                    5000
 manufacturer            5000
 reviews.date            5039
 reviews.dateAdded      15621
 reviews.dateSeen        5000
 reviews.didPurchase    39659
 reviews.doRecommend     5594
 reviews.id             39659
 reviews.numHelpful      5529
 reviews.rating            33
 reviews.sourceURLs      5000
 reviews.text               1
 reviews.title           5006
 reviews.userCity       39660
 reviews.userProvince   39660
 reviews.username        5007
 dtype: int64,
 'date_range': (Timestamp('2011-11-18 00:00:00+0000', tz='UTC'),
  Timestamp('2018-04-18 00:00:00+0000', tz='UTC')),
 'sampling_rate': 'daily',
 'geo_info': {'user_city_non_null': 0, 'user_province_non_null': 0}},
id                  12.607161
name                17.044881
asins                0.005043
brand               12.607161
categories          12.607161
keys                12.607161
manufacturer        12.607161
reviews.date        12.705497
reviews.dateAdded   39.387292
reviews.dateSeen    12.607161
```

Figure 3: Summary of Dataset Attributes & Missing Value Statistics

5.2 Data cleaning & Labelling

In the initial step, the pandas library was utilized to load the Amazon product reviews dataset from a CSV file (Review_Dataset.csv). After keeping only the required columns and initial cleaning, finally returned the cleaned dataset to 39,626 rows and 4 columns, affirming that the process is successful and providing a good basis for all further processing steps. A look at the cleaned dataset shows that each entry now has a valid product ASIN, review text, numeric rating, and recommendation status.

Then, generated sentiment labels from the numeric review_ratings. The get_sentiment() function translates the numeric scores into sentiment categories. The Python code uses Pandas to carry out this translation for all reviews in the data set. Then we graphed the distribution of these assigned sentiment labels to check if there was class imbalance or not. Here we used the seaborn and matplotlib libraries. We use the countplot() function to graph the number of reviews per sentiment class. The bar chart clearly illustrates that there are many more positive reviews compared to neutral and negative ones, making the argument for the selection of using class weighting while training the model to reduce bias against the majority class.

```
Cleaned dataset shape: (39626, 4)
```

	asins	reviews.text	reviews.rating	reviews.doRecommend
0	B01AHB9CN2	This product so far has not disappointed. My c...	5.0	True
1	B01AHB9CN2	great for beginner or experienced person. Boug...	5.0	True
2	B01AHB9CN2	Inexpensive tablet for him to use and learn on...	5.0	True
3	B01AHB9CN2	I've had my Fire HD 8 two weeks now and I love...	4.0	True
4	B01AHB9CN2	I bought this for my grand daughter when she c...	5.0	True

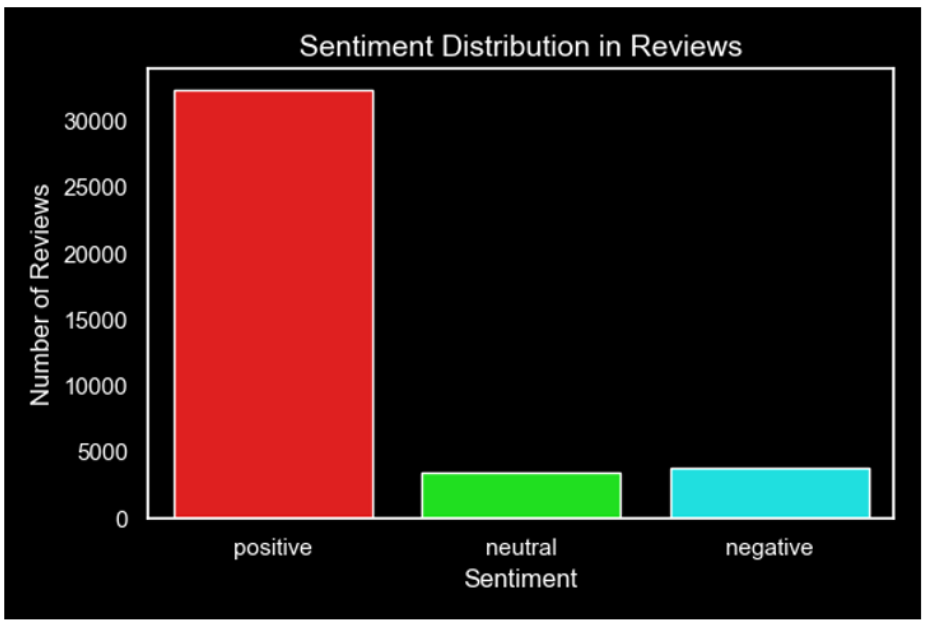

Figure 4: Sample of the cleaned Amazon review dataset & sentiment distribution

5.3 AI Content Filtering to Remove Low-Quality or Suspicious Reviews

Here, we implemented a rule-based filtering procedure to identify and eliminate reviews that are likely to be AI-generated, spam, or of low quality and lacking meaningful sentiment context. This was done through a Python function, is_suspicious_review, which utilized basic string and list operations to flag suspicious entries. Specifically, reviews with fewer than five words were identified, which is not useful input for sentiment modeling. Unnecessary repetitive usage of generic marketing words like "highly recommend" or "great product" was looked at as indicators of spam or artificially generated content. Also, reviews that contain a low uniqueness ratio were penalized because repetition would always direct towards auto-generated or non-human content. Suspicious reviews (True) were removed, while only better quality and more authentic reviews remained for the BERT-GRU model to work on. The BERT-GRU

model was provided with 39,327 out of 39,626 original reviews following filtering, which shows that hardly any low-quality or suspicious reviews had been removed, equilibrating the data amount with quality.

```
Original reviews: 39626, After filtering: 39327
```

Figure 5: Review count before & after AI-based filtering

5.4 BERT-GRU Sentiment Classifier Implementation

5.4.1 Review Tokenization

The deployment starts with the loading of a pre-trained BERT tokenizer (bert-base-uncased) from Hugging Face. The tokenizer splits raw, unparsed reviews into tokens, converts them into lower case, and assigns every token its BERT vocabulary equivalent ID. Sentiment labels (positive, neutral, negative) are also transformed into integers using LabelEncoder, which PyTorch needs for loss calculation during training. It pads and trims each review so that every input sequence has the same length (up to 128 tokens). The tokenizer also generates an attention mask, which tells the model which tokens are actual words and which are padding. The resulting tokenized dataset consists of input IDs, attention masks, and sentiment label encodings.

```
# Output basic tokenization info
tokenized['input_ids'].shape, tokenized['attention_mask'].shape, len(encoded_labels)
```

```
(torch.Size([39327, 128]), torch.Size([39327, 128]), 39327)
```

Figure 6: Tokenization output showing input IDs and attention masks

5.4.2 Data Preprocessing

A ReviewDataset() class is defined by subclassing torch.utils.data.Dataset. The class wraps the tokenized input_IDs, attention_masks, and labels, which PyTorch's DataLoader can then efficiently batch data and shuffle during training.__getitem__ is employed to yield one sample at a time as a dictionary with all necessary tensors. This renders the model capable of reading data in mini-batches efficiently, thus training faster and causing the model to generalize better. The entire dataset is then split into 80% training and 20% validation sets using PyTorch's random split functionality. The subsets are wrapped by DataLoader and batching (batch size 16), and shuffling is automatically performed.

```
# Confirm sizes
print(f"Train samples: {len(train_dataset)}, Validation samples: {len(val_dataset)}")
```

```
Train samples: 31461, Validation samples: 7866
```

Figure 7: Training & validation dataset split confirmation

5.4.3 Sentiment Model Workflow: BERT + GRU + Classifier

The BERT_GRU_FineTune() class is declared by extending torch.nn.Module to make the model utilize PyTorch's neural network capabilities. The fundamental architecture consists of three major components:

- **Pretrained BERT Model (bert-base-uncased)**: Pre-trained BERT Model (bert-base-uncased): This maps all input tokens to 768-dimensional embeddings that retain the semantic meaning, position, and segment information. Since BERT is fine-tuned in this case, its weights are not frozen (requires_ grad=True), and thus the model can adjust these embeddings for the specific sentiment classification task.
- **Bidirectional GRU Layer**: The BERT embeddings are passed into a GRU (Gated Recurrent Unit) layer of hidden_size=128. The GRU learns 128 features in both directions, giving a 256-dimensional vector for each sequence when bidirectional=True. GRUs are a good choice for sequential data because they have internal reset and update gates that control how much past information to keep, capturing the text dependencies.
- **Dropout and Classification Layer**: There is a dropout layer with a rate of 0.3 used for regularization, dropping certain neurons randomly during training to avoid overfitting. A dense, fully connected classification layer then projects the output of the GRU to three logits, one for each sentiment category, and the one with the best score is considered as the final sentiment.

Loss Function & Optimizer: The CrossEntropyLoss loss function is utilized, with class weights computed via compute_class_weight to address class imbalance. This serves to punish the model more for misclassifying a minority class and thus compelling balanced predictions for all classes. AdamW is used for optimization with a learning rate of 2e-5, which is a common initial value when fine-tuning BERT models. During training, the optimizer updates both BERT and GRU parameters as well as classifier weights.

Model Training Loop: To train the last BERT-GRU sentiment classification model, a custom training loop was used to incrementally update the model's parameters and evaluate its performance. Initially, at the beginning of each epoch, the model is first converted into training mode by calling model.train(), such that any layers that need to behave differently while training (for example, dropout) are enabled. In the past, the training data were processed in mini-batches by a PyTorch DataLoader, which keeps small batches of data in memory efficiently and provides shuffling for improved generalization.

In a batch, the tokenized input IDs, attention mask, and sentiment labels are passed to the device, where they are going to be processed — GPU if it is available, or CPU otherwise. The optimizer gradients are cleared at the start of each batch through the optimizer.zero grad() so that any residual gradients from earlier do not affect the present update. The model performs a forward pass to generate output logits on each example, these being scores for each sentiment class. The cross-entropy loss function then calculates how closely the logits match the real labels to compute how well the model is performing. The loss.backward() function performs the backpropagation, approximating gradients per parameter, and the optimizer.step() updates the weights in turn. This optimizes the BERT, GRU, and classification layers to close the gap between true and predicted sentiment labels. After all training batches are done, true and predicted labels are aligned to calculate the training accuracy of the epoch, and the average Training loss is also computed. To ensure that the model is doing well to make good generalizations on test data, a validation is also performed at every epoch. model.eval() is used to put the model into evaluation mode, which deactivates dropout along with other train-specific

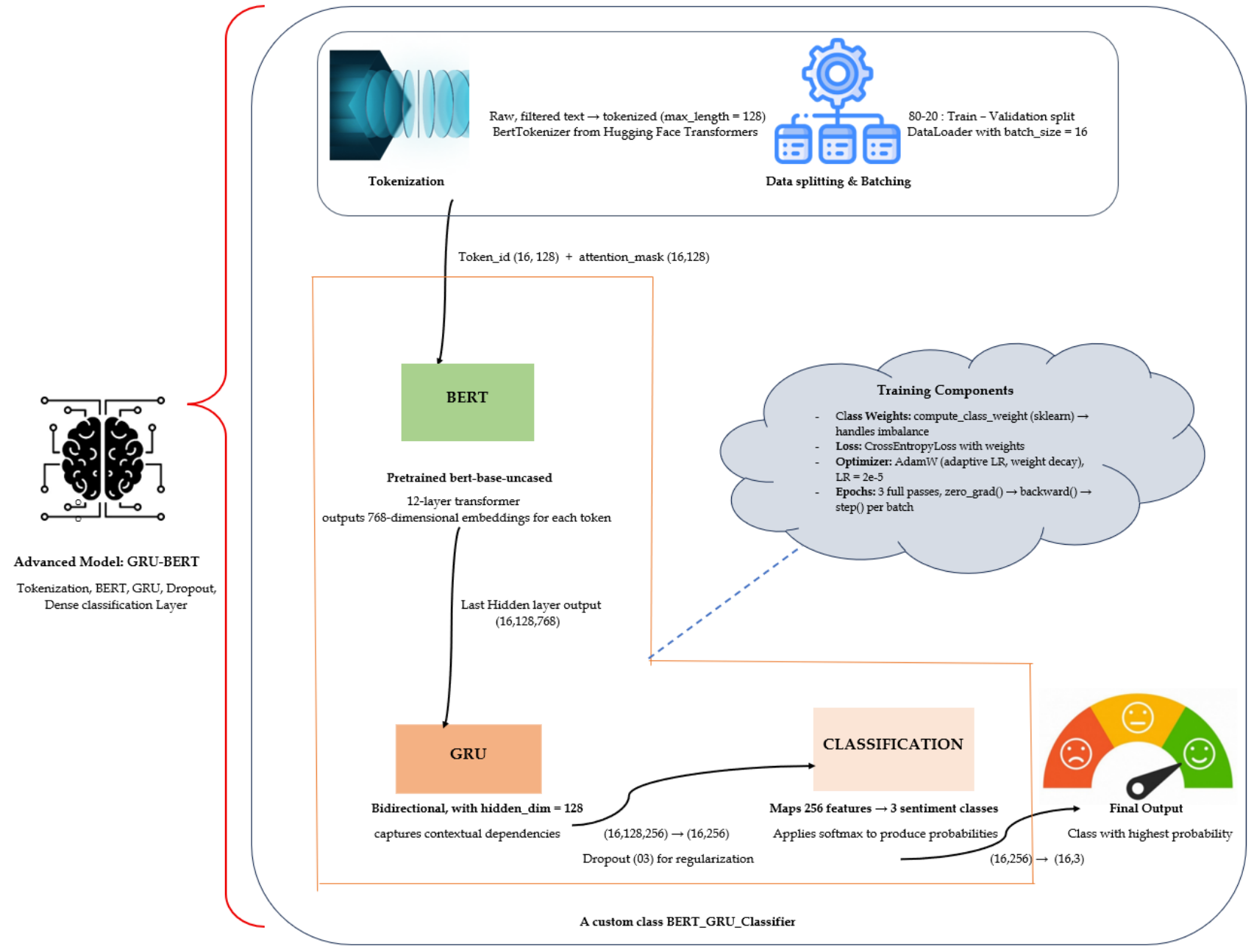

Figure 8: BERT + GRU + Classifier Model Work flow

procedures to ensure that there are stable predictions. torch.no grad() is also invoked to prevent the computation of gradients during validation to save memory and computation time since the weights are not updated during it.

When validated, the model iterates through batches of the validation set sequentially, predicts, and calculates loss per batch. The predicted classes are determined by finding the top-scoring output for each example with torch.argmax. These are compared to true validation labels so that training accuracy and mean validation loss can be calculated. Outputting loss and accuracy on training and validation per epoch provides a good idea as to whether the model is learning well and whether there's any sign of overfitting, eg, training loss decreasing when validation loss increases.

This careful architecture of this training loop, with its distinct separation of training and validation stages, ensures that the model can learn to recognize important patterns without sacrificing the ability for generalization to unseen material. It combines significant machine learning building blocks such as epoch-wise monitoring, backpropagation, and mini-batch gradient descent so that there is effective fine-tuning of the BERT-GRU model.

5.5 Demand forecasting

This final step is about employing the preprocessed and sentiment-augmented dataset to generate meaningful demand forecasts. The objective is to illustrate how validated sentiment and recommendation trends can be used to meaningfully improve time series forecasting.

5.5.1 Preprocessing & ASIN Selection

The first step in preparation for time series forecasting was to convert the reviews.date column into a proper datetime format using pandas. The reviews that had invalid or missing dates were disregarded to maintain the integrity of the monthly grouping. Then the

	asins	year_month	review_count
259	B018Y229OU	2016-01	1748
271	B018Y229OU	2017-01	1408
258	B018Y229OU	2015-12	1399
270	B018Y229OU	2016-12	1181
71	B00L9EPT8O,B01E6AO69U	2017-01	1105
298	B01AHB9CN2	2017-01	920
79	B00L9EPT8O,B01E6AO69U	2017-09	783
70	B00L9EPT8O,B01E6AO69U	2016-12	744
297	B01AHB9CN2	2016-12	719
78	B00L9EPT8O,B01E6AO69U	2017-08	612

```
Eligible ASINs: ['B00IOY8XWQ', 'B00OQVZDJM', 'B00U3FPN4U', 'B01BH83OOM', 'B018Y229OU', 'B00IOYAM4I', 'B00L9EPT8O,B01E6AO69U', 'B01BFIBRIE', 'B00QFQRELG', '
B018Y225IA', 'B00VINDBJK', 'B00ZV9PXP2', 'B00TSUGXKE', 'B0189XYY0Q', 'B018Y23MNM', 'B018SZT3BK', 'B01J2G4VBG', 'B01J4ORNHU', 'B01AHB9CN2', 'B01AHB9CYG', 'B
00UH4D8G2', 'B00QL1ZN3G']
```

Figure 9: ASIN Selection for Time-series Forecasting

data was grouped by the unique ASINs and year_month period, and a new column was created containing the number of reviews per

product per month. This grouping allowed for visual inspection and identification of products with typical review activity. To have sufficient data for modeling for the product that would be chosen, only those ASINs that had at least six months of review data were kept. A qualifying ASIN was then chosen as the case study for the time series forecasting.

5.5.2 Stationarity Test

After preparing the time series, it was necessary to verify if the series was stationary—a central requirement for ARIMA-based models. The Augmented Dickey-Fuller (ADF) test of the statsmodels library was utilized for this. The ADF test generates a test statistic and p-value; if the p-value is less than 0.05, one can reject the null hypothesis that the series is non-stationary (has a unit root). As seen in the output, the selected ASIN's review count series was stable, with the test statistic well below the critical values of the 1%, 5%, and 10% confidence levels. This suggests that no further differencing transformations were required before fitting the ARIMA or SARIMAX models.

```
ADF Statistic: -4.682175918134875
p-value: 9.088576120920526e-05
Critical Value (1%): -3.639224104416853
Critical Value (5%): -2.9512301791166293
Critical Value (10%): -2.614446989619377
Series is stationary.
```

Figure 10: ADF Test Results for Stationarity

5.5.3 Autocorrelation & Partial Autocorrelation Analysis

To find the right ARIMA parameters, Autocorrelation Function (ACF) and Partial Autocorrelation Function (PACF) plots were drawn.

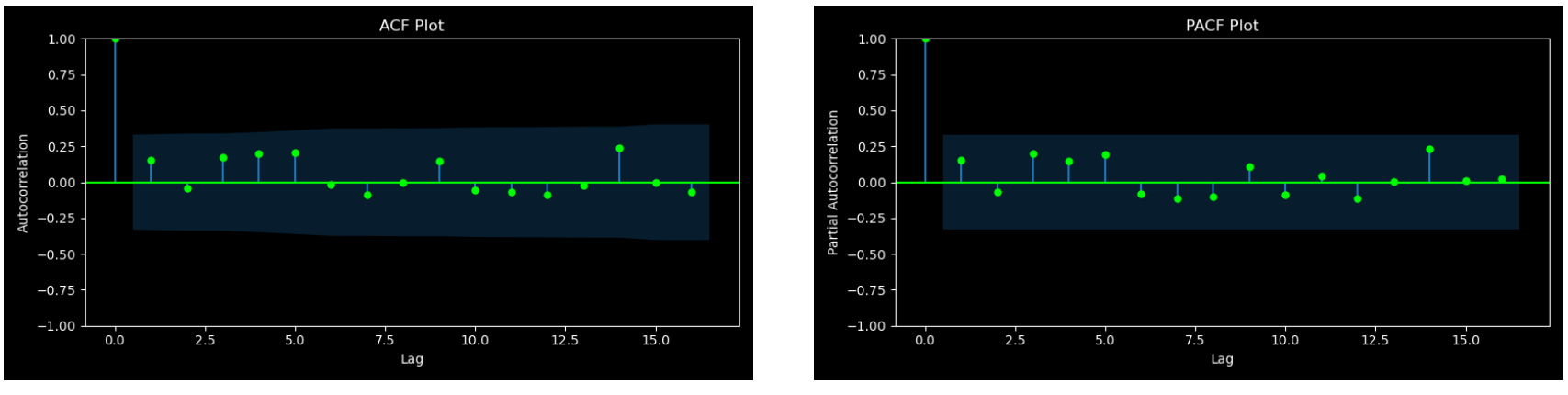

Figure 11: ACF & PACF Plots for Model parameter selection

These plots, using statsmodels.graphics.tsaplots, help in determining the correct order of moving average (MA) and autoregressive (AR) terms by showing the correlation of every lag with the original series. In the generated plots, there is a strong spike at lag 1 for both ACF and PACF, with all other lags falling inside the confidence intervals, reflecting little significance. This suggests an ARIMA model form of (1, 0, 1), where AR and MA terms are both 1, and differencing is not required (d).

5.5.4 SARIMAX Model Building for Demand Forecasting

Finally, the sentiment scores, cleaned review counts, and buyer recommendation rates were combined to build a sentiment-augmented SARIMAX model using statsmodels.tsa.statespace.sarimax. The model was constructed to train several different ARIMA order combinations. In each configuration, the model was trained against the training period (80% of the data) and then tested against the holdout period. The exogenous variables—recommendation rate and the sentiment score—were included to see if they improved forecast performance.

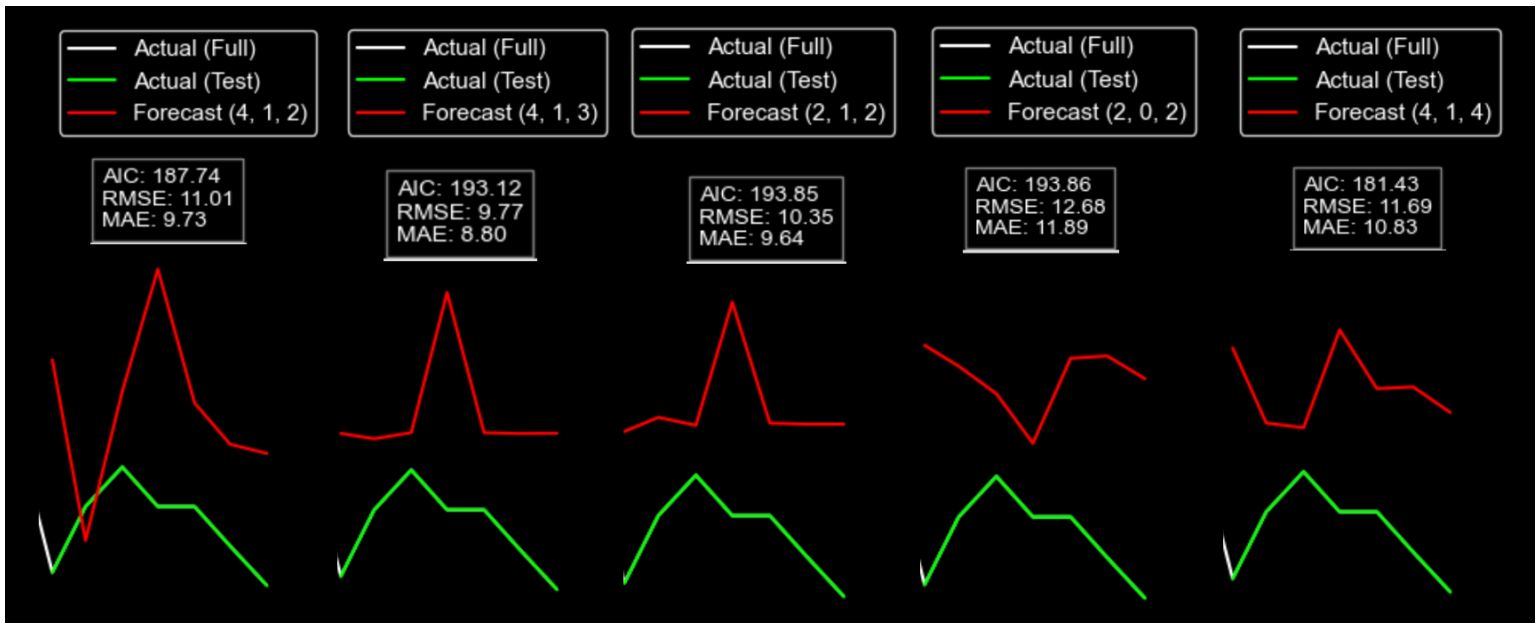

Figure 12: Actual Vs Forecasted plot for different SARIMAX model orders

6 Evaluation

6.1 Experiment 1 — Logistic Regression Baseline

To establish a baseline for the quality with which the rule-based sentiment labels (derived from numeric ratings) would be able to support supervised sentiment prediction, a basic multinomial logistic regression model was developed and evaluated. The baseline system was a checkpoint that was necessary to estimate the intrinsic predictive ability of the rating-to-label mapping before moving on to more complex architectures.

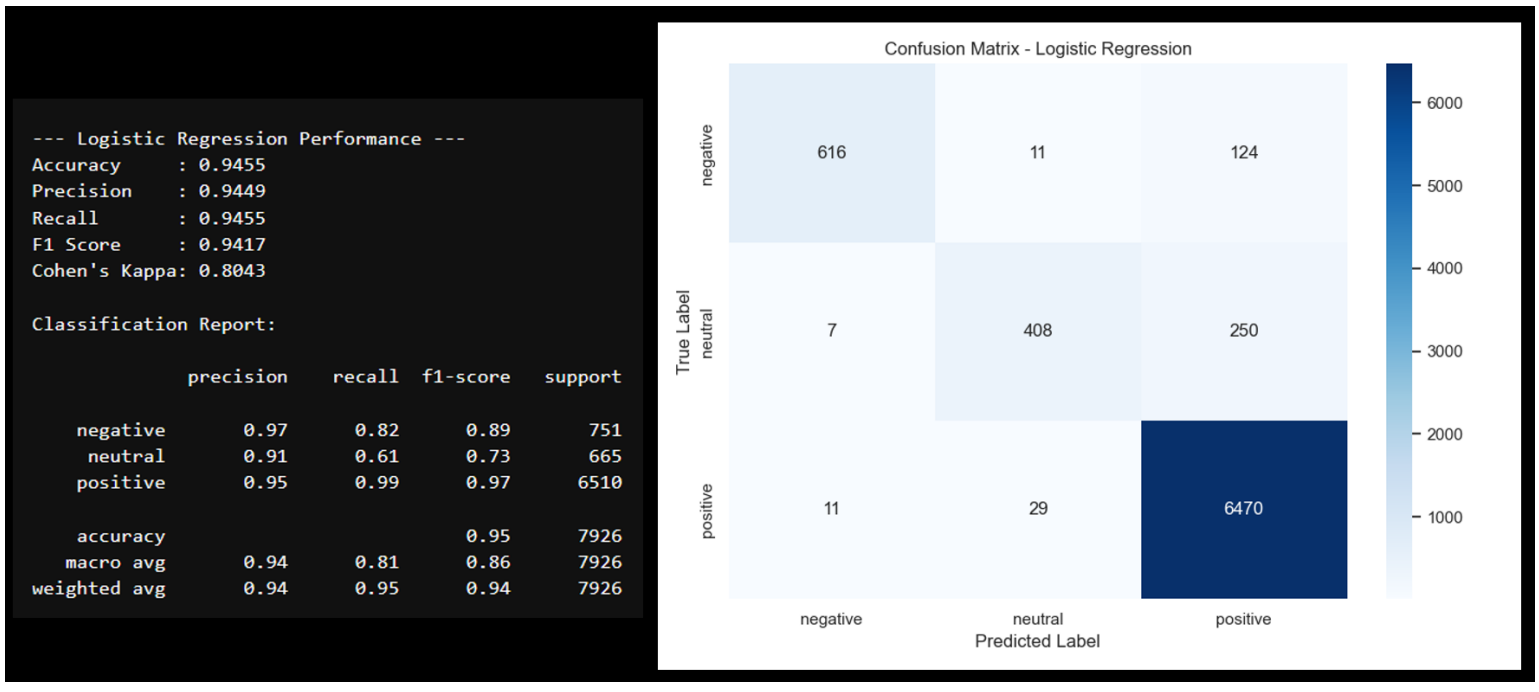

Figure 13: Performance metrics for Baseline Logistic Regression model

The results, as encapsulated in the below classification report and confusion matrix, are that the model achieved an overall accuracy of approximately 94.5%, with precision and recall scores very closely positioned at approximately 94%. The 0.80 Cohen's Kappa statistic expresses strong agreement beyond chance, meaning the labels provided a reasonable structure to make predictions. However, closer inspection of the per-class scores reveals some imbalance: while the model performs perfectly on the dominant positive class (F1-score of 0.97), its recall drops for the neutral class (0.61), reflecting the impact of class imbalance on the generalization of the model.

The confusion matrix showed how most reviews were correctly mapped to their respective sentiment categories, yet with evident confusion between neutral and positive or negative categories. The observation reinforced the understanding that while rating-based sentiment can tell a baseline model, it cannot capture finer contextual distinctions that a rating alone cannot express.

This baseline had two significant limitations. First, logistic regression with TF-IDF relies on a bag-of-words representation that models words in isolation and ignores the order and more complex contextual meaning of words in a sentence. As a result, it cannot capture subtleties such as sarcasm, subtle changes in tone, or contradictory statements within a single review. Second, the seemingly high overall performance creates a misleading impression of high accuracy because the rule-based labels (derived solely from ratings) don't necessarily match the sentiment expressed in the text — i.e., a reviewer might provide a high star rating but leave a negative comment, which won't be picked up by the model based on word frequency alone.

Overall, this initial experiment validated that the dataset structure was amenable to sentiment modeling but suggested that a context-aware model would be needed to handle the more subtle language of real review text. This finding explicitly guided the decision to develop an improved BERT-GRU model that exploits the full context of the review rather than just numeric ratings.

6.2 Experiment 2 — BERT-GRU sentiment classification with Frozen BERT

In this experiment, the standard logistic regression for sentiment classification was enhanced by training a more complicated hybrid deep learning model: the BERT-GRU model with frozen BERT layers. Freezing BERT means that the pre-trained language model's weights

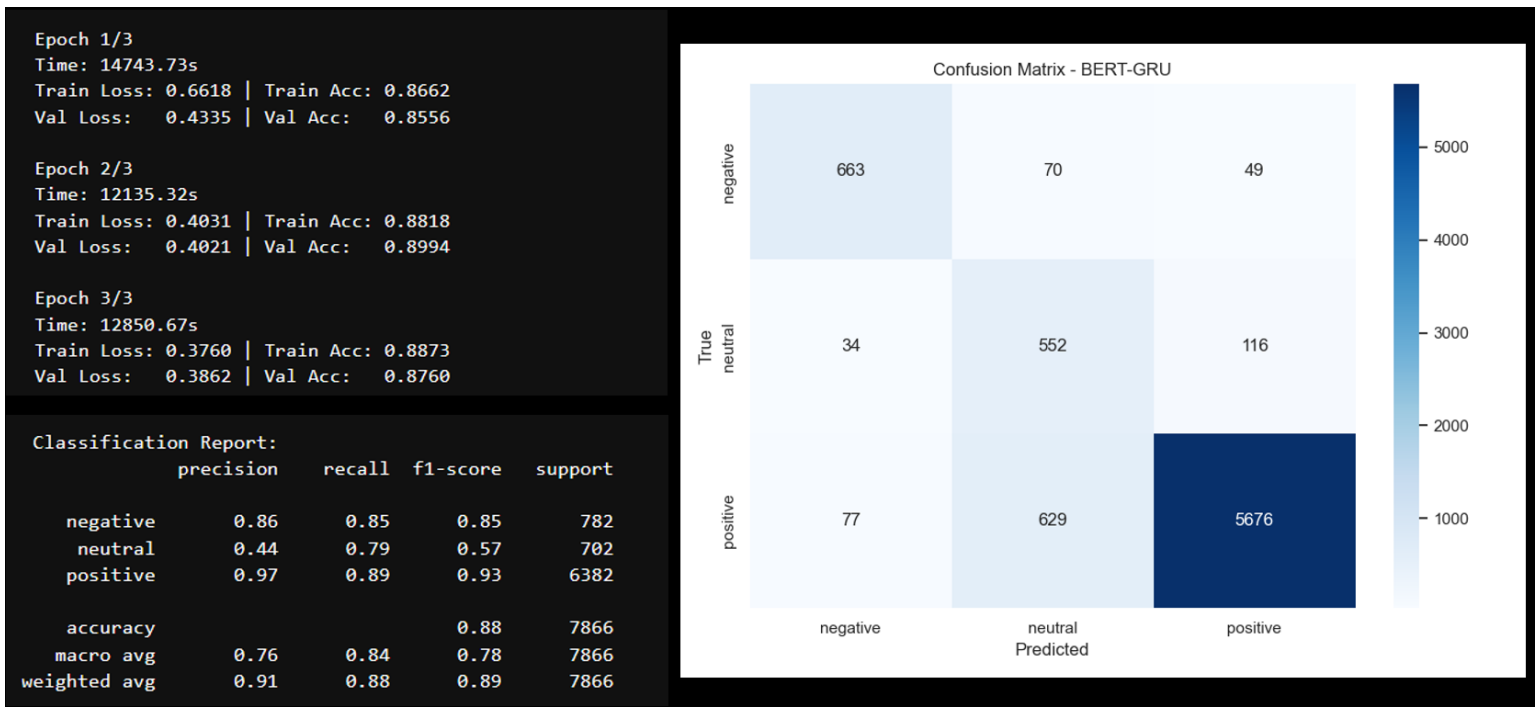

Figure 14: Performance metrics for Frozen BERT-GRU Model

are not updated during training. This model choice allows BERT to serve solely as a feature extractor for high-quality, contextualized embeddings of the input reviews. By doing so, training can guide computational resources towards learning the sequential dependencies

and sentiment-specific complexities through the GRU and an ultimate dense classification layer. This approach also conserves training time and prevents the risk of overfitting when only limited labelled data is available.

The training procedure was run for three epochs, as shown by the epoch summary output. There was steady improvement in validation and training accuracy across epochs, with the last validation accuracy reaching approximately 87.6%. This shows that even without fine-tuning BERT's deep transformer layers, the combination with the GRU was able to capture patterns beneficial for the sentiment labels.

The confusion matrix shows the model performed well on positive and negative classes, while the neutral class remains more challenging — no surprise given as its lower representation and probable overlap with both positive and negative sentiment expression. The following classification report also highlights these trends: precision and recall for positive reviews were high (0.97 and 0.89 respectively), whereas for neutral reviews, the precision dropped to 0.44, with a recall of 0.79 and an F1 score of 0.57. This indicates that while the frozen BERT-GRU was a step up from logistic regression when it came to context-sensitive predictions, some nuances — especially in borderline neutral reviews — are better captured when BERT's transformer layers have room to adapt through fine-tuning.

Overall, this frozen BERT-GRU model shows how the application of powerful pre-trained language representations with a sequential GRU layer on top can result in significant performance gains for review sentiment comprehension. This task was relevant to the research question in that it demonstrated that the simple application of BERT as a frozen feature extractor can already increase classification performance over baseline models, paving the way for establishing the incremental value of fine-tuning the entire model in the next phase.

6.3 Experiment 3 — BERT-GRU sentiment classification with Unfrozen BERT

In this final sentiment classification experiment, the BERT-GRU model was further enhanced by unfreezing pre-trained layers of BERT. Contrary to the case of the frozen one, where BERT was just a static feature extractor, this fine-tuning setup allowed BERT transformer weights to be learned throughout training. That is, the model can calibrate its contextual embeddings specifically for the sentiment classification task, leveraging the semantic patterns of the product reviews to catch more subtle linguistic cues that may not be picked up by rule-based labels.

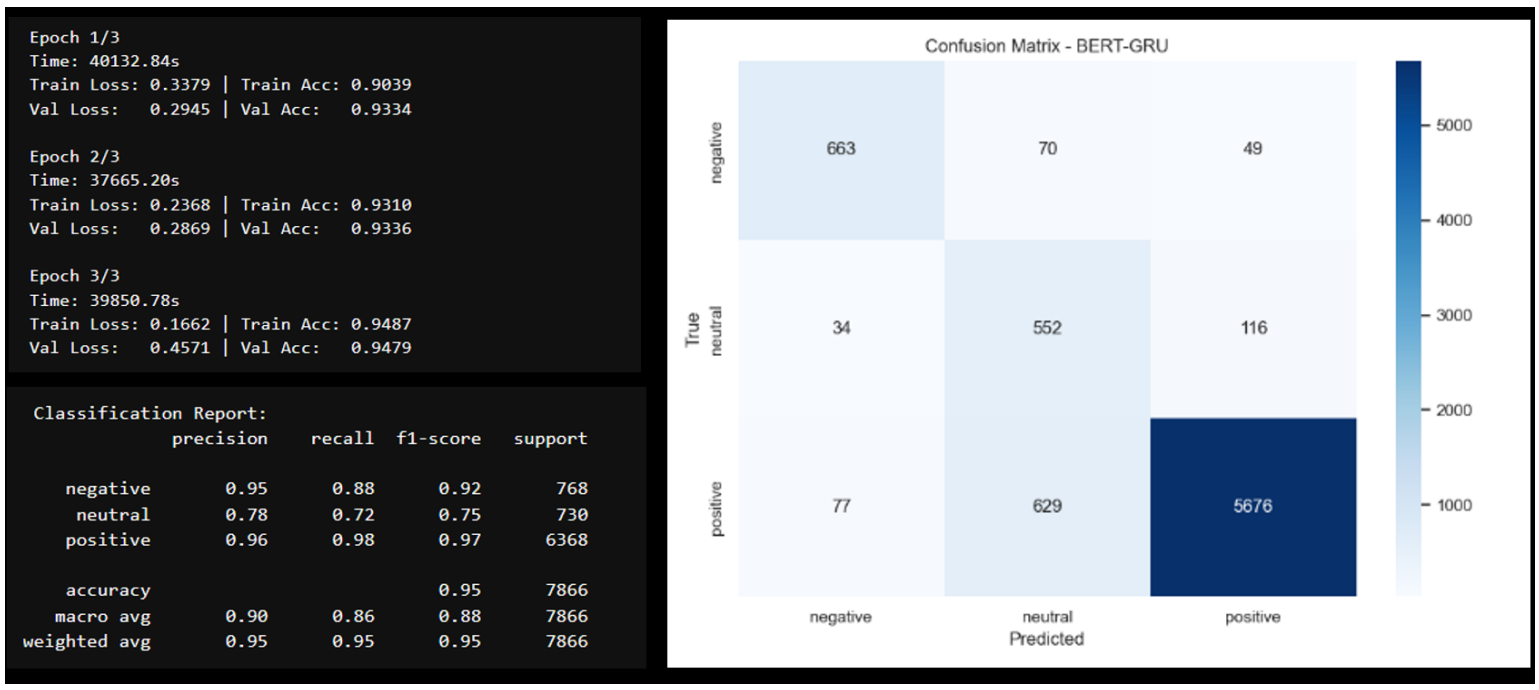

Figure 15: Performance metrics for Fine-tuned BERT-GRU Model

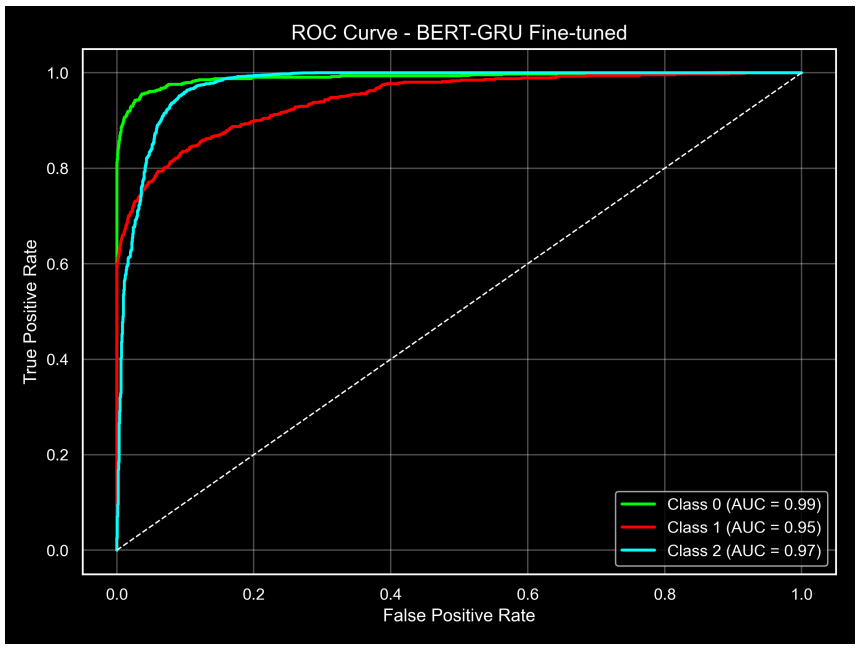

Figure 16: ROC Curve for Fine-tuned BERT-GRU Model

The model had good convergence in three training epochs, with validation accuracy reaching almost 95%. Training and validation loss reduced systematically, indicating consistent learning without signs of overfitting. The final metrics of classification, shown in the report given below, illustrate a superior macro average precision, recall, and F1-score of 0.90, 0.86, and 0.88, respectively, with all the weighted averages being very close to the global accuracy. This indicates that the model performed effectively for all three classes of sentiment, even when there was class imbalance.

The confusion matrix shows that the fine-tuned model resulted in an impressive reduction of misclassifications for the negative and neutral classes compared to the frozen setup. The neutral class, which is often the most challenging to classify accurately due to its bland sentiment, achieved an F1-score of 0.75, reflecting the benefit of fine-tuning BERT's layers.

The ROC curve shows good performance for all three classes. Class 0, Class 1, and Class 2 curves are close to the top-left corner, indicating a very high true positive rate and a very low false positive rate. The Area Under the Curve (AUC) values are 0.99 for Class 0, 0.95 for Class 1, and 0.97 for Class 2. These AUC values reflect excellent discriminative ability, with almost perfect discrimination by Class 0. Overall, the model performs well in terms of predictive performance and is effective in separating different sentiment classes.

The overall evaluation confirms that allowing BERT's transformer to update its embeddings to the specific dataset context meaningfully improved the model's ability to learn nuanced sentiments. Overall, the fine-tuned BERT-GRU model possessed a healthy blend of high accuracy and enhanced generalization, confirming its practical applicability to sentiment analysis where contextual understanding is paramount.

6.4 Ablation Study: Frozen vs Unfrozen BERT

Following training and evaluation of both versions of the BERT-GRU architecture, a direct comparison of the outcomes underlines the value added by fine-tuning the BERT layers for this sentiment analysis task. The frozen version used BERT as a static feature extractor, with its pretrained weights fixed, only training the GRU and classification layers. On the contrary, the non-frozen variant allowed the BERT encoder parameters to be trained throughout the training process, and therefore, the model could adapt its deep contextual embeddings more precisely to the uniqueness of this dataset.

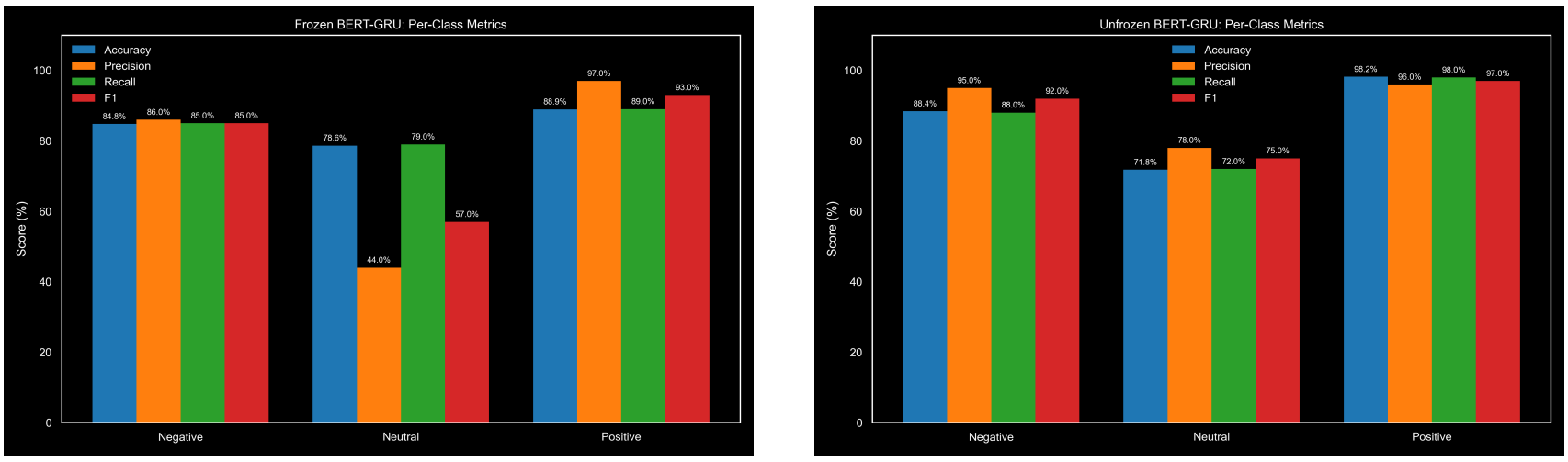

Figure 17: Performance Comparison of Frozen & Fine-tuned BERT-GRU Model

Model	Class	Precision	Recall	F1-score	Accuracy	Summary
Logistic Regression	Negative	97%	82%	89%	82%	Performs well for positive and negative reviews but struggles with neutral sentiment due to limited contextual understanding.
	Neutral	91%	61%	73%	61%	
	Positive	95%	99%	97%	99%	
BERT-GRU (Frozen BERT)	Negative	86%	85%	85%	85%	Improves contextual learning compared to Logistic Regression, boosting performance for negative and positive classes, but still weak for neutral due to frozen BERT layers.
	Neutral	44%	79%	57%	79%	
	Positive	97%	89%	93%	89%	
BERT-GRU (Fine-tuned)	Negative	95%	88%	92%	88%	**Best overall**: fine-tuning BERT captures nuanced sentiments, significantly improving neutral detection while maintaining high accuracy across all classes.
	Neutral	78%	72%	75%	72%	
	Positive	96%	98%	97%	98%	

Table 6: Performance Comparison of Sentiment Models

From bar plots, unfreezing BERT resulted in quantitatively improved scores on various metrics, particularly for the more context-dependent sentiment classes. Accuracy for the negative class went from around 84.8% (frozen) to 88.4% (unfrozen), with precision and F1-score also increasing sharply. The positive class, which is highly prevalent in the dataset, was also positively affected: accuracy jumped above 98%, showing that fine-tuning BERT enables even subtle contextual information in positive reviews to be captured.

Notably, the performance of the neutral class was inconsistent: its accuracy fell marginally in the unfrozen model compared to the frozen state, but precision and F1-score went up. It suggests that while the unfrozen model became even better at distinguishing actual neutral cases from borderline positive or negative reviews, it could still be thwarted by the prevalent uncertainty and the lower number of neutral examples.

Overall, the enhanced precision, recall, and F1-score of the unfrozen model show that employing a fully fine-tuned BERT layer allows the model to capture deeper contextual patterns, resolve ambiguity, and make more symmetrical predictions across sentiment classes. The outcome implies the effectiveness of fine-tuning for specific-domain tasks when working with noisy or subtle language data where fixed embeddings alone would be inadequate.

6.5 Experiment 4 — ARIMA-Based Demand Forecasting

Within this experiment, a simple ARIMA model was used as a baseline approach to forecast the monthly review count of an ASIN. The ARIMA model was built solely using the historical time series data — i.e., the review volume at the monthly aggregation level — without incorporating any external factors. Certain orders of ARIMA were attempted systematically (as shown in the summary table), and the best AIC was 210.02 for the model ARIMA(2,1, 1), along with an RMSE of 10.28, MAE of 9.85, and a relatively high MAPE of 187.62%. These measures of error demonstrate that even though the model tries to fit the broad seasonality and trends, the ability of the model to make correct predictions of absolute future values is limited when relying only on the history of review quantities. This variance is largely explained by the ARIMA model's supposition that all future demand is a function of its past values alone — and not other real-world drivers that may influence consumer behavior significantly. The visualization of the forecast illustrates that although the ARIMA(2, 1, 1) model attempts to capture the general trend of demand, it neglects to respond to sudden spikes and dips in the real test data. This experiment demonstrates that while ARIMA is a good starting point for demand forecasting, its predictive

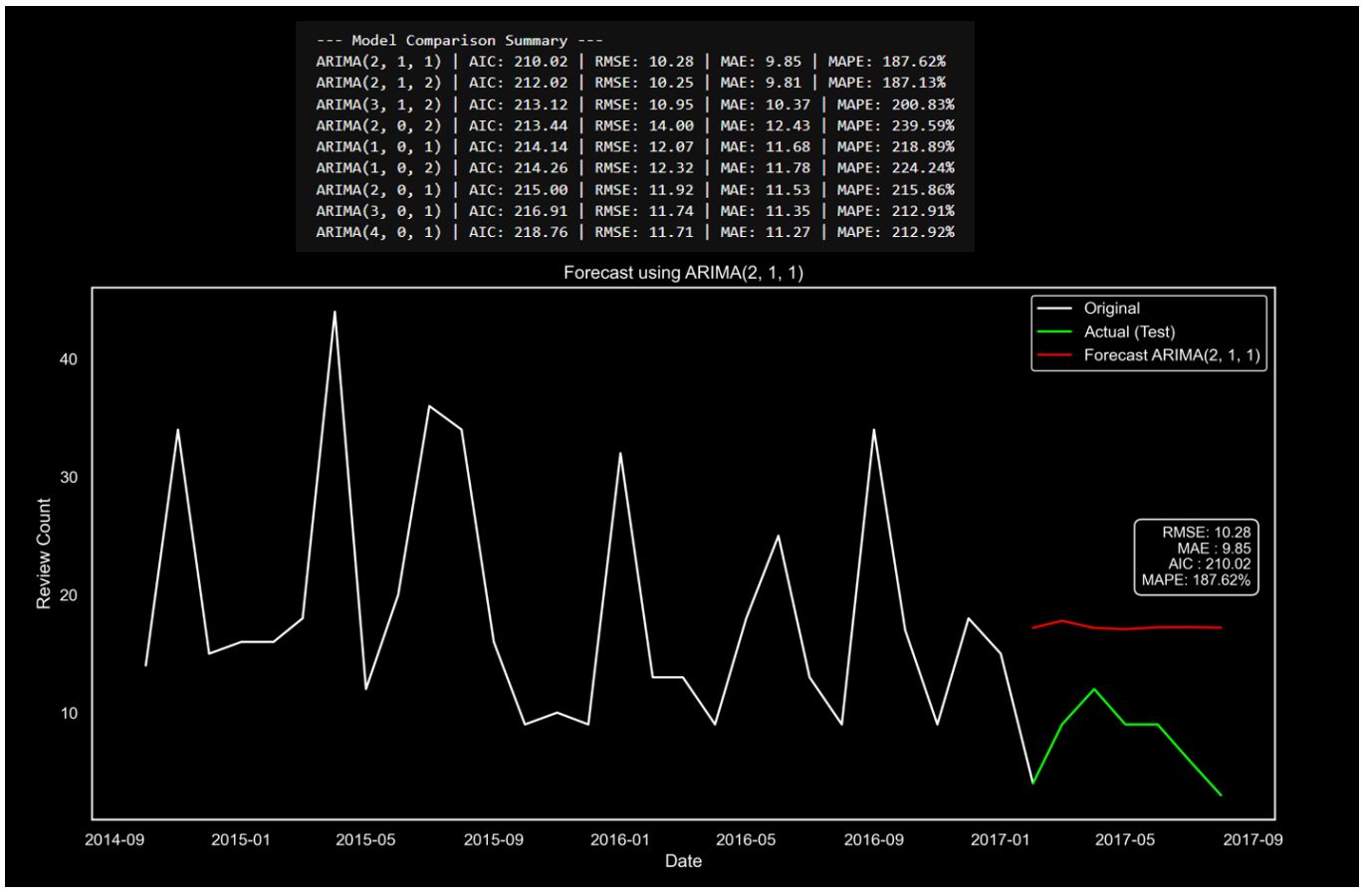

Figure 18: Performance of ARIMA-based forecasting Model

ability is necessarily restricted in situations where demand is subject to factors beyond simple historical trends. For example, changes in product sentiment, customer satisfaction, campaigns, or rates of recommendations could have a major effect on review/product volumes in the future.

6.6 Experiment 5 — SARIMAX with Recommendation Rate for Demand Forecasting

The second experiment extended the baseline ARIMA approach by introducing an exogenous variable — the recommendation rate — into a SARIMAX model. The idea was that while ARIMA takes only the lags and trend of the time series itself, SARIMAX allows us to introduce external predictors that could be of explanatory use in explaining demand variation.

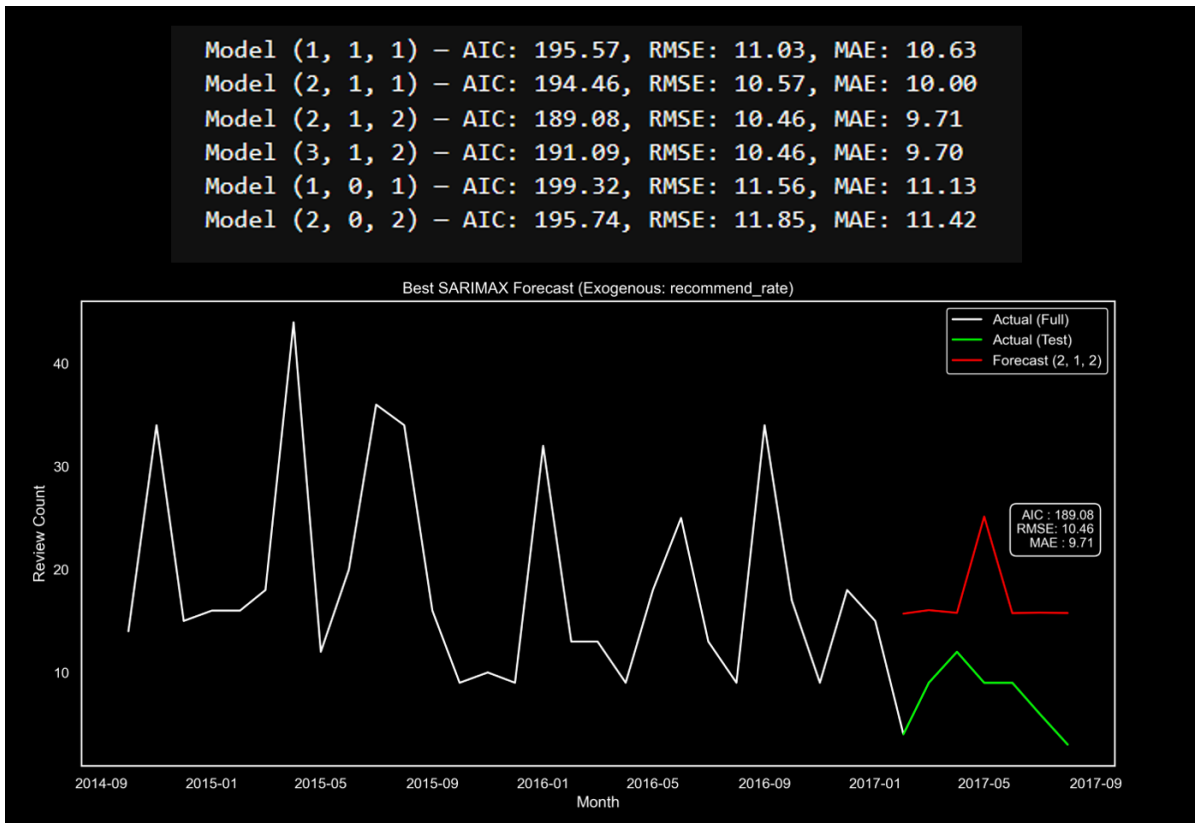

Figure 19: Performance of SARIMAX Model with Recommendation rate as Exogenous variable

In this case, the average recommendation rate for each month was included as an exogenous variable, the idea being that months where the recommendation rate was higher might be correlated with higher future demand.

To find the best-fit model, various SARIMAX orders were attempted systematically, as evident from the comparison output. The best model, SARIMAX(2, 1, 2), had an AIC of approximately 189.08 with an RMSE of 10.46 and an MAE of 9.71. This represents a marginal improvement in the AIC and error metrics over the basic ARIMA model, suggesting that by including the recommendation rate, there is moderate predictive capability.

The SARIMAX model forecast (red dashed line) in the forecast plot is more responsive to changes in the test interval than the ARIMA output, which indicates that the exogenous variable allowed the model to better follow demand signals in the guise of customer recommendations. Nevertheless, one can still observe significant discrepancies between forecast and actual values, which indicates that the recommendation rate by itself does not explain all the variation in review volume. This experiment offered a crucial point of learning for the research objective: introducing product-level recommendation behavior as an exogenous variable does have the effect of increasing demand forecasts, but more context — such as sentiment trends — might offer more explanatory value. This result had a direct impact on the next iteration of modelling using recommendation and sentiment as drivers for demand, making the layered approach to iterating on successively richer inputs for more realistic use.

6.7 Experiment 6 — SARIMAX with Recommendation & Sentiment as Exogenous Variables

Finally, in the final step of demand forecasting, the SARIMAX model was extended to include both the recommendation rate and the sentiment score as exogenous variables. This was to try to capture not only the autoregressive and seasonal trends seen in the review count but also customer attitude and probability of recommendation influencing future demand. Several model configurations were tried, and the SARIMAX(4, 1, 4) proved to be the most appropriate based on the lowest AIC (181.43) with error metrics under control (RMSE: 11.69, MAE: 10.83).

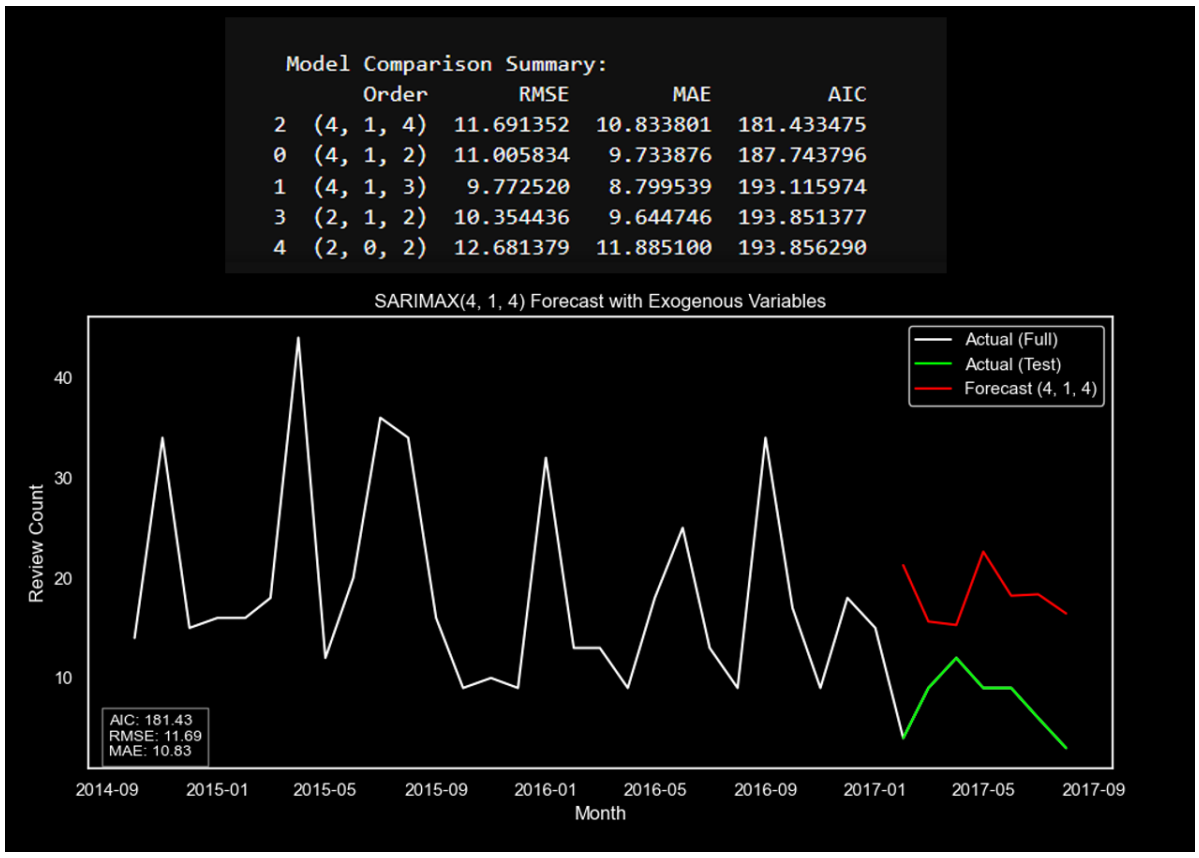

Figure 20: Performance of SARIMAX Model with Recommendation & Sentiment as Exogenous Variables

Compared to previous ARIMA and SARIMAX tests, this combined exogenous model provided the optimal fit, as the addition of sentiment in combination with recommendation rate resulted in a deeper understanding of demand behavior.

The plot of the forecast shows that even though the estimated review quantities don't quite match the actuals, including sentiment allows the model to better adjust for small directional shifts in demand. This demonstrates the value of combining unstructured review signals with standard numeric metrics.

Overall, this experiment shows that sentiment analysis can be of value to time series forecasting in e-commerce environments, allowing for more intelligent inventory control and more adaptive marketing response. By including a range of drivers of consumer behavior, the model is directly contributing to the project objective to illustrate how advanced text mining and time series techniques can be employed to enhance demand prediction accuracy for practitioners.

6.8 Result & Discussion

The overall outcomes from this work demonstrate the applicability and limitations of practice in applying sentiment analysis with demand forecasting on e-commerce reviews. The ultimate model employing BERT-GRU with an unfrozen BERT layer performed much better than the simpler baselines by being capable of learning deeper contextual word relationships within the review text. Not only did this enhancement show up as its higher per-class accuracy, precision, and F1-scores, but its power when classifying ambiguous or contradictory reviews also improved. In comparison with the frozen counterpart and the past logistic regression baseline, the fine-tuned BERT-GRU performed better in traversing the nuanced differences between negative, positive, and neutral sentiment — an essential stepping stone towards facilitating more meaningful downstream analysis. The word clouds also provide qualitative insight into how

Figure 21: Word Cloud for Positive & Negative Sentiment

this sentiment analysis supplements the text data. For example, in positive feedbacks, typical words like "love", "easy", "use", "great", and "enjoy" come up again and again in similar context, highlighting customer satisfaction with ease of use and value of product, while negative feedbacks feature prominently words like "stopped", "working", "battery", and "poor quality", which directly indicate similar issues. These trends indicate actionable information for product development and marketing teams in proving the practical-world value of accurate sentiment tagging over simple numerical values.

When the sentiment information was added to SARIMAX forecasting models, the results also reflected its promise. Although the best SARIMAX model that uses recommendation rate and sentiment score never exactly aligned with actual review counts, the forecasted trends would support the direction of actual movement in demand. That is, if exact numerical forecasts are challenging due to inherent

volatility, the inclusion of exogenous sentiment data enables the model to react more realistically to patterns in customer attitudes. To practitioners, this can translate into better inventory decisions, tailored promotions, or earlier detection of issues in products.

However, the experiments also make certain limitations and future avenues apparent. One was that rule-based sentiment annotations based on numerical ratings were used; these do not necessarily capture the true sentiment expressed in the text. A more fine-grained, manually created dataset would probably improve training quality and provide a better benchmark for subsequent work. Additionally, other input-related exogenous variables such as price fluctuation, seasonality, or external events influencing demand can be added to enrich the forecasting models.

Overall, since the ultimate design successfully demonstrated that the combination of cutting-edge NLP with time series forecasting has the potential to generate more informative business data, subsequent advancements can extend the scope even wider by including other more diverse sources of data, exploring alternative architectures (such as LSTMs or transformers for time series), and applying the approach to more granular sales measures. This critical evaluation supports the contention that a context-aware, sentiment-driven pipeline can occupy the gap between business action and unstructured review text — a contention well served by the previous research encountered within the literature review.

7 Conclusion & Future Work

This project had the goal of answering the research question: "To what extent can verified customer sentiment be effectively integrated into demand forecasting models to enhance prediction accuracy and support more informed inventory decisions in e-commerce?" To address this, the main objectives were to develop a consistent sentiment analysis model on actual customer reviews, provide clear insights into how sentiment drives product demand, and show how these insights could integrate into time-series forecasting to improve decision-making.

With these objectives in mind, the research progressed from simple baseline models, such as logistic regression over rule-based tags, to a high-level BERT-GRU model and fine-tuning the BERT to use the full context of text reviews. The final unfrozen BERT-GRU model had strong performance, with high precision, recall, and F1-scores across all sentiment classes, which suggested that it was picking up on subtle sentiment beyond the limits of what ratings could indicate. Word cloud analysis also supported the findings by uncovering significant positive and negative themes that influenced customer experiences — with actionable recommendations for product and service improvement.

For the demand forecasting component, including sentiment scores and recommendation rates as exogenous variables within the SARIMAX model was extremely effective. While the numerical forecasts were not always close to actual volumes — due to limitations such as short time horizons and absence of broader external factors — the forecast trends closely followed the direction of actual demand. This proves that verified sentiment signals can usefully augment standard forecasting, enabling retailers to more accurately predict peaks and troughs and make smarter inventory and marketing decisions.

This study has thus effectively demonstrated that the integration of verified customer sentiment with demand forecasting models can make the interpretation of predictions more meaningful and deliver richer, context-rich insights for e-commerce inventory management.

But there are important limitations to mention. One is that review volume was used as a proxy for established sales volume in the process of demand forecasting, which may not necessarily reflect actual purchasing habits. Adding authenticated sales data can further enhance later accuracy and operations forecasts. Further, the experiments only considered one product category, therefore limiting the applicability of the results to other products or sectors. Additionally, the time-series models did not include other external factors like promotions, competitor actions, or macroeconomic events, all of which have significant impacts on demand in real scenarios.

Future studies may address these gaps by building a larger manually tagged dataset to train and test the sentiment model more thoroughly, making it even more accurate. Including more advanced external variables — say, pricing changes, advertising spend, seasonality, or even broader patterns of customer behavior — may be able to add further to the forecasting element. Exploring more robust architectures like transformer-based time series models could potentially deal with higher-level dependencies more effectively as well. On an operational level, a useful next step would be to pilot this integrated approach across multiple products and product categories to validate its scalability and value to actual businesses. These innovations give rise to persuasive industry applications, including dynamic pricing mechanisms that react to immediate signals of demand, real-time inventory control to minimize stockouts or overstock, and sentiment-based marketing interventions targeted towards personal tastes.

In short, this project demonstrates that properly validated sentiment analysis can indeed be integrated into demand forecasting methodologies and offer useful tools to e-commerce players to better align inventory and marketing programs with changing customer sentiments. With further development and field-testing, this approach might be able to make more responsive, data-driven decision-making a reality in a competitive retail market.

References

Ahmed, M., Chen, Q., & Li, Z. (2020). Constructing domain-dependent sentiment dictionary for sentiment analysis. *Neural Computing and Applications*, *32*, 14719–14732. https://doi.org/10.1007/s00521-020-04824-8

Akhtar, M. S., Ekbal, A., Bhattacharyya, A., et al. (2017). A multilayer perceptron based ensemble technique for fine-grained financial sentiment analysis. *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, 540–546. https://doi.org/10.18653/v1/D17-1057

Alamdari, P. M., Navimipour, N. J., Hosseinzadeh, M., Safaei, A. A., & Darwesh, A. A. (2020). Systematic study on the recommender systems in the e-commerce. *IEEE Access*, *8*, 115694–115716. https://doi.org/10.1109/ACCESS.2020.3002803

AlBadani, B., Shi, R., & Dong, J. (2022). A novel machine learning approach for sentiment analysis on twitter incorporating the universal language model fine-tuning and svm. *Applied System Innovation*, *5*(1), 13. https://www.mdpi.com/2571-5577/5/1/13

Alzahrani, M. E., Aldhyani, T. H., Alsubari, S. N., Althobaiti, M. M., & Fahad, A. (2022). Developing an intelligent system with deep learning algorithms for sentiment analysis of e-commerce product reviews. *Computational Intelligence and Neuroscience*, 1–10. https://doi.org/10.1155/2022/3840071

Antony, C., et al. (2025). A systematic review of sentiment analysis approaches and techniques. *2025 6th International Conference on Mobile Computing and Sustainable Informatics (ICMCSI)*. https://doi.org/10.1109/ICMCSI64620.2025.10883219

Assiri, A., & Gumaei, A. (2024). Deberta-gru: Sentiment analysis for large language model. *Computers, Materials & Continua*. https://doi.org/10.32604/cmc.2024.050781

Bansal, B., & Srivastava, S. (2019). Hybrid attribute based sentiment classification of online reviews for consumer intelligence. *Applied Intelligence*, *49*(1), 137–149. https://link.springer.com/article/10.1007/s10489-018-1299-7

Birjali, M., Kasri, M., & Beni-Hssane, A. (2021). A comprehensive survey on sentiment analysis: Approaches, challenges and trends. *Knowledge-Based Systems*, *226*, 107134. https://doi.org/10.1016/j.knosys.2021.107134

Cambria, E., Das, D., Bandyopadhyay, S., & Feraco, A. (2017). Affective computing and sentiment analysis. In *A practical guide to sentiment analysis: Socio-affective computing* (pp. 1–10). Springer. https://doi.org/10.1007/978-3-319-55394-8_1

Chen, Y. (2024). Sentiment analysis based on multiscale convolutional neural network and bidirectional long short-term memory. *Membrane Technology*. https://doi.org/10.52710/mt.97

Chintalapudi, N., Battineni, G., Canio, M. D., Sagaro, G., & Amenta, F. (2021). Text mining with sentiment analysis on seafarers' medical documents. *International Journal of Information Management Data Insights*, *1*(1), 100005. https://doi.org/10.1016/j.jjimei.2020.100005

Darraz, N. (2023). Using sentiment analysis to spot trending products. *2023 Sixth International Conference on Vocational Education and Electrical Engineering (ICVEE)*. https://doi.org/10.1109/ICVEE59738.2023.10348252

Geetha, M. P., & Renuka, D. K. (2021). Improving the performance of aspect based sentiment analysis using fine-tuned bert base uncased model. *International Journal of Intelligent Networks*, *2*, 64–69. https://doi.org/10.1016/j.ijin.2021.06.005

Hussein, D. M. (2018). A survey on sentiment analysis challenges. *Journal of King Saud University - Engineering Sciences*, *30*(4), 330–338. https://doi.org/10.1016/j.jksues.2016.04.002

Iddrisu, A.-M., Mensah, S., Boafo, F., Yeluripati, G. R., & Kudjo, P. A. (2023). A sentiment analysis framework to classify instances of sarcastic sentiments within the aviation sector. *International Journal of Information Management Data Insights*, *3*(2), 100180. https://doi.org/10.1016/j.jjimei.2023.100180

Jayakody, J., et al. (2021). Sentiment analysis on product reviews on twitter using machine learning approaches. *2021 International Conference on Decision Aid Sciences and Application (DASA)*. https://ieeexplore.ieee.org/document/9682291

Nimmi, K., et al. (2022). Pre-trained ensemble model for identification of emotion during covid-19 based on emergency response support system dataset. *Applied Soft Computing*, *122*, 108842. https://pubmed.ncbi.nlm.nih.gov/35465357/

Obiedat, R., Al-Darras, D., Alzaghoul, E., & Harfoushi, O. (2021). Arabic aspect-based sentiment analysis: A systematic literature review. *IEEE Access*, *9*, 152628–152645. https://doi.org/10.1109/ACCESS.2021.3127140

Pang, B., Lee, L., & Vaithyanathan, S. (2002). Thumbs up? sentiment classification using machine learning techniques. https://aclanthology.org/W02-1011/

Sanyal, S., & Wamique, M. (2019). Factors affecting customer satisfaction with e-commerce websites – an omani perspective. *2019 International Conference on Digitization (ICD)*. https://doi.org/10.1109/ICD47981.2019.9105780

Savci, P., & Das, B. (2023). Prediction of the customers' interests using sentiment analysis in e-commerce data for comparison of arabic, english, and turkish languages. *Journal of King Saud University - Computer and Information Sciences*, *35*(3), 227–237. https://doi.org/10.1016/j.jksuci.2023.02.017

Singh, R., et al. (2024). Ai-driven multi-modal demand forecasting: Combining social media sentiment with economic indicators and market trends. *Journal of Informatics Education and Research*, *4*(3). https://www.researchgate.net/publication/387075739

Sirisha, U., & Bolem, S. C. (2022). Aspect-based sentiment & emotion analysis with roberta, lstm. *International Journal of Advanced Computer Science and Applications (IJACSA)*. https://doi.org/10.14569/IJACSA.2022.0131189

Tubishat, M., Idris, N., & Abushariah, M. (2021). Explicit aspects extraction in sentiment analysis using optimal rules combination. *Future Generation Computer Systems*, *114*, 448–480. https://doi.org/10.1016/j.future.2020.08.019

Zhang, J., Zhang, A., Liu, D., & Bian, Y. (2021). Customer preferences extraction for air purifiers based on fine-grained sentiment analysis of online reviews. *Knowledge-Based Systems*, *228*, 107259. https://doi.org/10.1016/j.knosys.2021.107259

Zhang, Y., et al. (2024). A comparative study of sentiment analysis on customer reviews using machine learning and deep learning. *Computers*, *13*(12), 340. https://doi.org/10.3390/computers13120340