**Capstone Project Report**

***On***

**“E-Commerce Sentiment Analysis”**

***Submitted to***

**Talent Sprint**

***And***

**Indian Institute of Technology, Madras**

***For the evaluation against the course***

**PG Level Advanced Certificate on Applied Data Science and Machine Learning (ADSML)**

**Cohort 6**

***Submitted by***

**Group 8**

**(Akshay, Udith, Shanmugavalli, Srinivas, Manisha, Swapnil, Basker)**

**Mentor: Sriram**

**Capstone Project: E-commerce Sentiment Analysis—3rd May 2025**

**Purpose**

This document outlines the end-to-end approach taken by our team to build a suite of machine learning (ML) and deep learning (DL) models to perform sentiment analysis on Amazon product reviews. The primary goal is to classify customer reviews into one of three categories—Positive, Negative, or Neutral—using advanced natural language processing (NLP) techniques. The document details the technical methodology followed, from transforming raw review text into numerical embeddings, to applying various ML and DL models for sentiment classification.

**Background**

In the digital economy, customer satisfaction is critical to a business’s success. E-commerce platforms, in particular, generate a vast amount of customer feedback in the form of product reviews. These reviews offer valuable insights into customer perceptions, product shortcomings, and areas for improvement. However, since these reviews are unstructured textual data, extracting actionable insights at scale poses a significant challenge.

To address this, businesses need a robust mechanism to convert unstructured text into structured sentiment indicators that can be used by product analysts and decision-makers. Sentiment analysis provides a scalable and automated way to interpret customer feedback, enabling data-driven product enhancements and improved customer experiences.

**Solution**

The proposed solution involves designing a sentiment analysis pipeline that can automatically label customer reviews as Positive, Negative, or Neutral. The overall approach includes two major steps:

**Step 1: Text Vectorization (Conversion to Embeddings):**

Before feeding textual data into any ML or DL model, it needs to be converted into numerical format—embeddings. We explored two broad categories of text vectorization techniques:

1. *Frequency-Based Vectorization*

*Count Vectorizer:* Converts text into a matrix of token counts. Each unique word in the corpus is treated as a feature, and its frequency in a document determines the value.

*TF-IDF (Term Frequency–Inverse Document Frequency):* Enhances the basic count vectorizer by assigning weights to words based on how important a word is to a document relative to the entire corpus. Commonly used words like “the” or “is” are down-weighted, while more unique and informative words are given higher weights.

1. *Semantic Embedding Models*

*Word2Vec:* Learns word embeddings using shallow neural networks. It captures semantic relationships by placing similar words close together in vector space.

*Sentence-BERT (SBERT):* A powerful transformer-based model that generates sentence-level embeddings, retaining semantic context better than Word2Vec or traditional vectorizers. SBERT is particularly effective for tasks that require understanding of sentence-level nuances, such as sentiment classification.

**Step 2: Sentiment Classification:**

Once the reviews were vectorized, various classification models were trained to predict the sentiment labels. We implemented and compared two categories of classifiers:

1. *Traditional Machine Learning Models:*

*Logistic Regression:* A baseline linear model for classification.

*Support Vector Machines (SVM):* Effective in high-dimensional spaces with strong performance on text classification.

*Random Forests:* An ensemble learning method that combines multiple decision trees for more robust predictions.

1. *Deep Learning Models:*

*Multilayer Perceptron (MLP):* A feedforward neural network used for basic non-linear classification.

*LSTM (Long Short-Term Memory):* A type of recurrent neural network (RNN) that can capture long-range dependencies in sequential data.

*GRU (Gated Recurrent Units):* A simpler alternative to LSTM, often with comparable performance.

*Transformer-based Models:* State-of-the-art architectures such as BERT, which use self-attention mechanisms to model complex relationships in text. These models are pre-trained on large corpora and fine-tuned on our sentiment dataset for classification.

The project included comparative experiments combining different embedding strategies with a range of classifiers. The final analysis highlights the trade-offs between performance, accuracy, and computational complexity for each model pairing, aiding in the identification of the optimal solution for scalable sentiment analysis in an e-commerce setting.

***Neural Network Approach:***  
*NN #1: Sentence BERT with SMOTE + Simple Neural Network*

This model combines powerful sentence embeddings with a custom neural classifier to effectively handle sentiment classification. It begins with the all-MiniLM-L6-v2 variant of Sentence-BERT, which transforms raw text reviews into dense 384-dimensional vector representations that capture semantic meaning. To address the challenge of class imbalance, SMOTE (Synthetic Minority Over-sampling Technique) is applied on the SBERT embeddings, generating synthetic examples for underrepresented classes to ensure a more balanced training set. The processed embeddings are then passed through a fully connected feed-forward neural network constructed using the Keras Sequential API, which comprises five dense layers with ReLU activations and a final Softmax layer for multiclass classification (Negative, Neutral, Positive). Regularization techniques such as L2 penalties (λ = 0.001), Batch Normalization, and Dropout (rates ranging from 0.3 to 0.5) are incorporated throughout the model to enhance generalization and mitigate overfitting. This combination of advanced sentence encoding, synthetic balancing, and robust deep learning design makes NN #1 a strong baseline for sentiment prediction.

*NN #2: Sentence BERT with SMOTE Tomek + Simple Neural Network*

This model enhances the baseline design by integrating a more refined data balancing technique and robust training strategies. Starting with 384-dimensional sentence embeddings from SBERT (all-MiniLM-L6-v2), this model applies SMOTE-Tomek, a hybrid sampling method that not only oversamples the minority class using synthetic examples but also removes Tomek links—ambiguous borderline samples—resulting in a cleaner and better-separated dataset. The classification engine is a fully connected neural network similar to NN #1, but with improved training dynamics. It uses the Adam optimizer (learning rate = 0.001) and is trained using the Sparse Categorical Crossentropy loss function, ideal for multi-class problems with label-encoded targets. The model features L2 regularization (λ = 0.001), Batch Normalization, and Dropout layers (with rates of 0.5 to 0.3) after each dense layer (except output) to prevent overfitting and stabilize training. To further enhance generalization and convergence, callbacks such as EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint are employed. This architecture carefully balances data quality, model complexity, and training efficiency, making it a more refined and resilient variant of the original neural network approach.

*NN #3: Sentence BERT with SMOTE Tomek with LSTM*

This model advances the architecture by integrating temporal sequence modeling through LSTM layers, making it well-suited for capturing sequential dependencies in the SBERT embeddings. Each input sample is a 384-dimensional sentence embedding, balanced using SMOTE-Tomek to address class imbalance by both oversampling minority classes and removing noisy borderline examples. The core of the model is a Long Short-Term Memory (LSTM) network, which, unlike feedforward networks, can retain contextual signals across dimensions in the input embedding. The model uses the Adam optimizer (learning rate = 0.001) and is trained using Sparse Categorical Crossentropy, suitable for multi-class classification with integer-encoded targets. To combat overfitting and ensure stable training, it includes L2 regularization (λ = 0.001) on all LSTM and Dense layers (except the output), along with Batch Normalization and Dropout layers (rates: 0.5 for LSTM, 0.5 and 0.3 for Dense). Training is further optimized using EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint callbacks. By incorporating temporal modeling with rigorous regularization and data balancing, this architecture provides a strong foundation for nuanced sentiment classification in sentence embeddings

*NN #4: Sentence BERT with SMOTE Tomek with GRU*

This model introduces a more efficient recurrent architecture using Gated Recurrent Units (GRUs), ideal for capturing contextual patterns in sentence embeddings with reduced computational overhead compared to LSTMs. Each input is a 384-dimensional SBERT embedding, balanced using SMOTE-Tomek to mitigate class imbalance by synthesizing minority samples and removing ambiguous ones. The model starts with a Bidirectional GRU layer (128 units), enabling the network to learn dependencies in both forward and backward directions, improving context sensitivity. This is followed by a Dense layer (64 units, ReLU) to further transform the learned features. Regularization is enforced through L2 penalties (λ = 0.01) on the GRU and Dense layers, along with Batch Normalization and Dropout (rates: 0.5 and 0.4, respectively) to reduce overfitting and enhance generalization. It uses the Adam optimizer (learning rate = 0.001) and Sparse Categorical Crossentropy loss for efficient multi-class classification. With EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint callbacks, the model ensures optimal performance and robustness. This architecture strikes a balance between sequential understanding and computational efficiency, making it a strong candidate for sentence-level sentiment classification

*NN #5: Sentence BERT with SMOTE Tomek with Transformers*

This model leverages the power of Transformer Encoder blocks to enhance representational depth and capture intricate relationships within SBERT embeddings. Input to the model consists of 384-dimensional SBERT vectors, preprocessed through SMOTE-Tomek to correct class imbalances by both oversampling and cleaning boundary samples. At its core, the model features four stacked Transformer Encoder blocks, each incorporating Multi-Head Attention with dropout (0.4) and Feed-Forward Networks (FFN) with dropout (0.3). LayerNormalization is applied within each block to stabilize training, and L2 regularization is implicitly handled through dense layer weights inside these blocks.

Following the Transformer encoders, the output is passed through a Multi-Layer Perceptron (MLP) consisting of two Dense layers (128 and 64 units, ReLU activation) with dropout (0.3) to aid generalization. Finally, a Softmax output layer predicts one of the three sentiment classes. The model uses a low learning rate (0.0001) with the Adam optimizer to accommodate the complexity of the Transformer layers, along with Sparse Categorical Crossentropy as the loss function. EarlyStopping, ReduceLROnPlateau, and ModelCheckpoint callbacks ensure training efficiency and safeguard against overfitting. This architecture excels in extracting nuanced patterns and relationships from sentence embeddings, making it highly suitable for sentiment tasks involving subtle contextual cues.

*NN #6: Sentence BERT + SMOTE Tomek + Transformers + Weighted Loss Function*

This model is designed for sentiment classification of text reviews, utilizing a combination of advanced techniques to improve accuracy and robustness. The model begins with Sentence-BERT embeddings, which encode the input text into 384-dimensional vectors. It then passes through four transformer encoder blocks, applying attention mechanisms and regularization techniques like L2 and dropout to capture semantic relationships in the text. Afterward, the model uses a multi-layer perceptron (MLP) structure with two dense layers (128 and 64 units) to further process the information. The output layer consists of three units with a softmax activation function, suitable for multi-class classification. To address class imbalance, the model incorporates SMOTE and Tomek Links for data preprocessing and uses a weighted loss function. Regularization techniques, including dropout and layer normalization, help prevent overfitting, ensuring the model generalizes well for sentiment classification tasks.

*NN #7: SBERT + SMOTE Tomek + Transformers + Weighted Loss + Pipeline of Models*

This model is a two-stage transformer-based sentiment classification model that first classifies text reviews into “Positive” vs “Rest” (Negative/Neutral), and then refines the classification by distinguishing between “Negative” and “Neutral” sentiments for the samples predicted as “Rest.” Both stages use SBERT embeddings, with four transformer encoder blocks, global average pooling, and dense layers. The model incorporates SMOTE and Tomek Links for handling class imbalance and applies a weighted loss function to ensure balanced predictions. This pipeline structure improves classification precision by breaking down the task into simpler binary classification problems.

*NN #8: SBERT + SMOTE Tomek + Transformers + Weighted Loss + Pipeline of Model + Autoencoders*

This model is an enhanced two-stage sentiment classification pipeline that introduces autoencoders to compress SBERT embeddings before classification. The model first uses an autoencoder to reduce the 384-dimensional SBERT embeddings to a compact 128-dimensional representation, helping eliminate noise and improve generalization. This compressed embedding is then fed into two sequential transformer-based binary classifiers: the first distinguishes Positive from Rest (Negative/Neutral), and the second further classifies the “Rest” into Negative vs Neutral. Each classifier consists of four transformer encoder blocks with dropout (0.4 in attention, 0.3 in FFN), followed by global average pooling and dense layers (64 units with ReLU and a final softmax layer with 2 output units). The model continues to use SMOTE + Tomek Links to address class imbalance and applies a weighted loss function for better performance on underrepresented classes. This architecture combines the strengths of transformer encoders and dimensionality reduction to build a more robust and efficient sentiment classification system.

*NN #9:* *SBERT + Embedding Scaling + Auto encoder + Simple NN + Pipeline of Models*

This model is a lightweight yet structured sentiment classification pipeline that combines SBERT embeddings, autoencoder-based dimensionality reduction, and simple feedforward neural networks for classification. Starting with 384-dimensional SBERT embeddings, the model applies SMOTE-Tomek for class balancing and StandardScaler for input normalization. Each stage of the pipeline begins with an autoencoder, which compresses the input to a 64-dimensional latent space through successive linear layers (384 → 256 → 128 → 64) with ReLU activations. In Stage 1, this compressed representation is fed into a binary classifier that distinguishes Positive from Rest (Negative/Neutral), while Stage 2 uses a separate autoencoder and classifier to differentiate Negative from Neutral within the “Rest” predictions. Both classifiers consist of two ReLU-activated dense layers (128 and 64 units) with 0.3 dropout, followed by a final softmax-compatible linear output layer with 2 units. The model uses MSELoss for training autoencoders and CrossEntropyLoss for classification, optimized via Adam/AdamW with a learning rate of 0.001. Training is done via custom loops, enabling finer control and transparency over model progress

*NN #10: SBERT + Embedding Scaling + Auto encoder + Simple NN + Pipeline of Models + SMOTE*

This model is a robust two-stage sentiment classification pipeline that combines SBERT embeddings, autoencoder-based dimensionality reduction, and simple neural networks, with a focus on class balance and modular design. It uses the all-MiniLM-L6-v2 SBERT model to generate 384-dimensional embeddings, which are standardized using StandardScaler and balanced via SMOTE. Each stage of the pipeline—Stage 1: Positive vs Rest, and Stage 2: Neutral vs Negative—has its own dedicated autoencoder and binary classifier. The autoencoders, trained separately for the Positive and Neutral classes using MSELoss, compress the embeddings down to a 64-dimensional latent space without dropout. These encoded vectors are then passed to binary classifiers comprising two ReLU-activated dense layers with 0.3 dropout, followed by a final linear layer that outputs logits for two classes. The classifiers are trained using CrossEntropyLoss and the AdamW optimizer. This architecture enhances generalization by isolating sentiment distinctions into simpler binary tasks, leveraging class-specific compression and dropout-regularized classifiers for more accurate and stable sentiment predictions.

*#11 Ensemble:*

This is an ensemble architecture that integrates the strengths of the previous ten models to deliver a more robust and generalized sentiment classification system. By combining diverse approaches—ranging from transformer-based architectures (Models 6–8), autoencoder-driven dimensionality reduction (Models 8–10), to staged binary classification pipelines and data balancing techniques like SMOTE-Tomek—the ensemble leverages complementary decision boundaries and mitigates individual model biases. Each constituent model contributes predictions which are then aggregated, typically through majority voting or weighted averaging, to produce the final sentiment label. This ensemble strategy enhances prediction stability, improves classification accuracy across all sentiment classes (Positive, Negative, Neutral), and ensures better handling of edge cases and class imbalance.

*RoBERTa #1 : Pre-trained RoBERTa Sentiment Classifier*

This is a default pre-trained RoBERTa-based sentiment classifier that leverages the rich contextual understanding of the RoBERTa language model to predict sentiment directly from raw text inputs. Without additional fine-tuning or architectural modifications, it outputs sentiment probabilities—typically across Positive, Negative, and Neutral classes—using a classification head built on top of RoBERTa’s final hidden states. This model benefits from large-scale pretraining on diverse corpora, enabling strong baseline performance in general sentiment analysis tasks with minimal setup. While it may lack task-specific optimization compared to custom models, it offers rapid deployment, solid accuracy, and is ideal for benchmarking or use cases requiring quick sentiment labeling.

*RoBERTa #2 : Fine-tuned RoBERTa Sentiment Classifier*

This is a fine-tuned RoBERTa sentiment classifier tailored specifically for the given text review dataset. Building on the powerful language understanding capabilities of the pre-trained RoBERTa model, it is further trained on labeled sentiment data—allowing the model to learn task-specific nuances and domain-specific vocabulary. Fine-tuning adjusts RoBERTa’s weights along with the classification head, typically a softmax layer over three classes (Positive, Neutral, Negative), resulting in improved accuracy and generalization compared to the default model. This approach combines the strength of transfer learning with dataset-specific adaptability, making it highly effective for precision sentiment analysis.

*Gen AI : Anthropic Claude Sonnet 3 with Few Shot Prompting*

This utilizes Anthropic’s Claude Sonnet 3, a powerful large language model, to perform sentiment classification through few-shot prompting. Instead of traditional training or fine-tuning, this approach involves providing the model with a handful of labeled examples alongside new input text to infer the sentiment—leveraging its strong in-context learning capabilities. Claude Sonnet 3 processes these prompts and generalizes from the examples to accurately classify sentiment as Positive, Negative, or Neutral. This method offers high flexibility, requires no model training, and is especially useful in zero-to-low resource scenarios or rapid prototyping, with the added advantage of explainable, human-readable outputs.

***Conventional Approach:***

*Conv #1: Using Random Forest Classifier and SBERT Embeddings*

This employs a Random Forest Classifier using SBERT embeddings as input features to perform sentiment classification. In this approach, each text review is first converted into a fixed-size, 384-dimensional embedding using the SBERT model, capturing semantic relationships in a dense vector form. These embeddings are then fed into a Random Forest—a robust ensemble of decision trees—that learns to classify the sentiment based on patterns in the feature space. Known for its resistance to overfitting and ease of interpretability, the Random Forest provides a strong baseline for sentiment tasks, especially when computational efficiency and model transparency are priorities.

*Conv #2: Using SVM and SBERT Embeddings*

This combines Support Vector Machine (SVM) with SBERT embeddings for sentiment classification. Each text review is transformed into a 384-dimensional SBERT embedding, capturing its contextual and semantic information. These dense vector representations are then used as input to a linear or kernel-based SVM, which finds the optimal hyperplane to separate the sentiment classes (Positive, Neutral, Negative) with maximum margin. SVM is particularly effective in high-dimensional spaces and works well with small to medium-sized datasets, offering strong generalization capabilities and solid baseline performance for text classification tasks.

*Conv #3: Using Neural Network and CountVectoriser*

This leverages a Count Vectorizer combined with a simple neural network for sentiment classification. In this approach, text reviews are first transformed into sparse, high-dimensional vectors using the Count Vectorizer, which captures the frequency of each word in the corpus without considering semantic context. These vectors are then fed into a neural network consisting of one or more dense layers with non-linear activations, enabling the model to learn patterns and relationships within the raw word counts. While this method is less context-aware than models using embeddings, it serves as an interpretable and computationally efficient baseline, particularly effective on smaller or more structured datasets.

*Conv #4: Using Neural Network and Word2Vec*

This combines a Neural Network with Word2Vec embeddings for sentiment classification. In this approach, each text review is first converted into dense vector representations using Word2Vec, which captures semantic relationships between words based on their context in the training corpus. These pre-trained word embeddings are then passed into a neural network, which learns to identify sentiment patterns by processing the embeddings through layers of neurons. The neural network can capture more complex, non-linear relationships between the word embeddings and sentiment labels, offering an improvement over simpler models like count-based methods. This method balances the richness of semantic understanding from Word2Vec with the flexibility and power of neural networks for sentiment classification tasks

*Conv #5: Using Neural Network and TF-IDF*

This combines a Neural Network with TF-IDF (Term Frequency-Inverse Document Frequency) features for sentiment classification. In this approach, text reviews are first transformed into numerical representations using the TF-IDF technique, which weights words based on their frequency in a given document relative to their frequency across all documents, highlighting important terms while reducing the impact of common ones. These TF-IDF vectors are then fed into a neural network, which processes the features through one or more dense layers to learn patterns related to sentiment classification. This method balances the importance of individual words in context with the neural network’s ability to capture complex, non-linear relationships between features and sentiment, making it effective for a wide range of text classification tasks.

***Comparing Various Models:***

The chart compares F1 scores across various neural network models, including standard neural networks (NN #1-10), ensemble models, RoBERTa variants, Claude Sonnet 3, and convolutional networks (Conv #1-5). Consistently across all models, positive F1 scores maintain the highest performance at around 0.95-1.0, while negative F1 scores generally range between 0.4-0.7, and neutral F1 scores show the lowest performance at 0.2-0.4. There's a notable dip in performance for both negative and neutral scores in the later convolutional models (Conv #3-4), though Conv #5 shows some recovery in performance. The ensemble model and RoBERTa variants demonstrate relatively stable performance compared to the standard neural networks.

The chart compares memory usage and inference time across ten neural network models (NN #1-10). For the first six models (NN #1-6), both memory usage and inference time remain consistently low and stable. However, there's a notable spike in inference time starting at NN #7, reaching nearly 2 seconds per sample and maintaining this elevated level through NN #8, before dropping back to baseline levels for NN #9-10. Memory usage shows a gradual increasing trend from NN #6 onwards, reaching approximately 1.2 MB by NN #10, contrasting with the more volatile pattern of inference times.

**Measures of Success:**

The team aims to evaluate the performance of the models through the following key metrics:

1. Comparison of Model Performance: A comprehensive analysis of metrics such as F1-score, inference time, and computational cost across different models to assess their effectiveness, efficiency, and overall suitability for sentiment classification tasks.
2. Scalability and Adaptability: Evaluating the solution’s ability to scale and adapt to diverse text analysis use cases, ensuring it can handle varying data sizes, types, and complexities while maintaining high performance and flexibility.

**Conclusion:**

1. Transformer-based Models: The two-stage sequential classification pipeline using transformer-based models outperformed simpler architectures, including large language models (LLMs), especially in detecting the “Neutral” class.
2. Ensemble Technique: The use of ensemble methods significantly improved performance, enhancing the robustness and generalization of the models.
3. Class Imbalance Solutions: Techniques such as SMOTE, SMOTE-Tomek, and a Weighted Sparse Categorical Cross-Entropy Loss Function were effectively implemented to address the issue of class imbalance.
4. Precision and Recall Trade-offs: To improve Precision and Recall for minority classes, deeper and more complex neural networks were developed, which increased the risk of overfitting. This risk was mitigated through advanced methods such as Batch/Layer Normalization, L2 Regularization, Dropout, Model Checkpoints, Early Stopping, and ReduceLROnPlateau.
5. Interpretability Challenges: As model complexity grew, interpretability decreased. Tools like SHAP struggled to effectively explain the predictions made by the more complex neural networks.
6. Traditional ML Models: While traditional machine learning models provided better interpretability, they fell short in Precision and Recall, particularly for accurately classifying the minority classes, unlike the more advanced neural ensemble methods.
7. Impact of Class Imbalance and Labeling Errors: Despite consistently high F1-scores for the majority (Positive) class (>0.95), the performance on minority classes suffered due to significant class imbalance, which was exacerbated by data labeling errors.
8. Efficiency of Simpler Models: Simpler models, such as NN#1 and NN#2, were notably more resource-efficient and quicker to train compared to the more complex architectures.
9. Challenges with Ambiguous Sentiment: Models faced difficulties with reviews containing ambiguous or mixed sentiments, which impacted classification consistency and accuracy.

**Next steps and Enhancements Planned**

1. Real-time Inference Integration: We plan to enhance the solution by integrating real-time inference capabilities, enabling the model to process and classify text inputs instantly as they arrive.
2. Exploring Class Imbalance Solutions: We will experiment with additional class imbalance techniques from the imblearn package to further refine model performance, particularly for minority classes.
3. Improving Explainability: We aim to improve model transparency and interpretability by incorporating SHAP for better understanding and explaining the predictions made by our models.
4. Building an Application: We will develop a user-friendly application that utilizes this model, making sentiment classification accessible and actionable for end users.