Final Project

RSM 8512 Leveraging AI and Deep Learning Tools in Marketing

Group 8

Abstract

This study presents a systematic evaluation of four sentiment analysis models—VADER, SVM, LSTM, and BERT—leveraging a large-scale Amazon review dataset to assess their classification performance. Through rigorous modeling and fine-tuning, the study provides a comparative analysis to justify the effectiveness of each model in sentiment classification tasks.

1 Introduction

In the digital age, customer reviews have become a cornerstone of consumer decision-making and business strategy. Remarkably, **70% of shoppers trust online reviews as much as personal recommendations**, underscoring their pivotal role in shaping purchasing behaviour and brand perception. Sentiment analysis—the automated extraction of subjective opinions from text—has thus emerged as an essential tool for businesses seeking to interpret this vast, unstructured feedback at scale.

However, the inherent complexity of human language, including nuances, sarcasm, and context-dependent meanings, presents significant challenges for accurate sentiment classification.

This project addresses a critical question: **Which sentiment analysis model best balances accuracy, computational efficiency, and contextual understanding for real-world applications?**

To answer this, we conduct a systematic comparison of four distinct approaches:

- VADER (Valence Aware Dictionary and sEntiment Reasoner): a lexicon-based model optimized for social media text.
- SVM (Support Vector Machine): a traditional machine learning model leveraging TF-IDF features.
- LSTM (Long Short-Term Memory): a recurrent neural network designed to capture sequential dependencies.
- BERT (Bidirectional Encoder Representations from Transformers): a cutting-edge transformer model excelling at contextual understanding.

By evaluating these models on a 3.6 million-sample Amazon review dataset, this study offers actionable insights for businesses automating sentiment analysis and provides practical guidelines for machine learning practitioners selecting models. Beyond comparing accuracy and speed, we also bridge theoretical concepts—such as attention mechanisms and gradient optimization—with their real-world implementation.

2 Data Overview and Pre-processing

Our project utilizes the Kaggle Amazon Reviews dataset for sentiment analysis, comprising approximately 3.6 million reviews in the training set and 400,000 in the test set. For computational efficiency, we worked with a stratified random sample of 50,000 training and 10,000 testing reviews while preserving the dataset's core characteristics. This comprehensive collection contains reviews paired with their corresponding star ratings, where 1-2 stars are labeled as negative sentiment

(__label1__) and 4-5 stars as positive sentiment (__label2__).

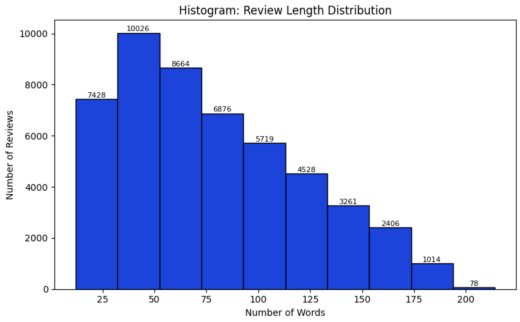


Figure 1: Review Length Distribution

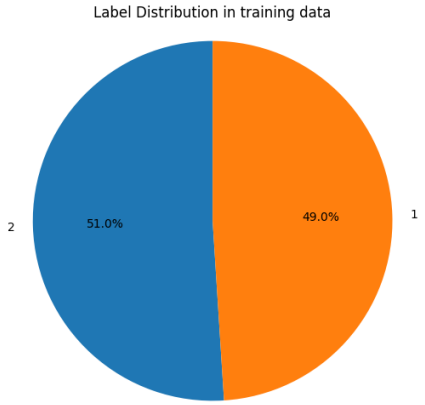


Figure 2: Pie Chart of Label Distribution in Training Data

The dataset exhibits an ideal balance with approximately 50% positive reviews, preventing sentiment bias in our models. Reviews showcase considerable diversity in topics and emotional expression—ranging from enthusiastic praise for products like the Chrono Cross soundtrack to pointed criticism of product flaws. Review length varies significantly, typically between 25-100 words with an average of 82 words. The distribution of word counts for reviews displays a right-skewed distribution with a long tail toward higher token counts.This variation is meaningful as shorter reviews tend to be direct while longer ones incorporate multiple opinions or emotions.

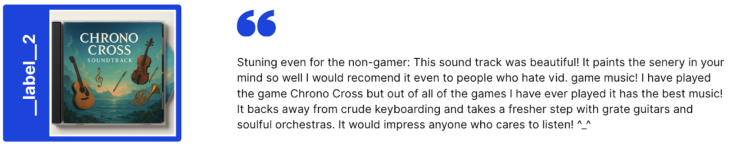


Figure 3: Real Amazon review (label 2) – richly positive and emotionally expressive.

Our data preparation pipeline included converting text to lowercase, removing punctuation, emojis, and non-alphabetic characters, reducing extra spaces, and converting text into word sequences. The dataset was originally provided in raw text format (.ft.txt) following the pattern "__label__1 Batteries died within a year..." and required Python-based processing to transform it into a structured DataFrame suitable for analysis and modeling.

The dataset was then systematically partitioned in an 8:2 ratio to create robust training and validation sets for model development. For the Support Vector Machine implementation, we applied additional text cleaning procedures, employing regular expressions to eliminate invalid symbols and redundant punctuation. Text normalization was also performed, which included converting all tokens to lowercase to ensure consistency throughout the corpus.

3 Methodology

3.1 VADER

3.1.1 VADER Overview

VADER (Valence Aware Dictionary for Sentiment Reasoning) is a rule-based sentiment analysis tool designed for social media and short text. Unlike machine learning methods that require training, VADER uses a predefined lexicon and grammatical heuristics to calculate sentiment polarity. Its

key output is the **compound score**, which is a normalized metric ranging from –1 (most negative) to +1 (most positive).

3.1.2 VADER Result

We evaluated two rule-based sentiment classification strategies using VADER: the compound score method and the positive-vs-negative (Pos/Neg) score comparison. The compound score method predicts sentiment as positive if the compound score is greater than or equal to zero, and negative otherwise. This approach is simple and efficient but can struggle with neutral or mixed-sentiment reviews. In contrast, the Pos/Neg method compares the raw positive and negative sentiment scores output by VADER and classifies a review as positive only if the positive score exceeds the negative score. This strategy is more sensitive to clearly polarized sentiment but still has difficulty with emotionally complex inputs. On a dataset of 50,000 training samples and 10,000 test samples, the Pos/Neg method slightly outperformed the compound score method in terms of accuracy and F1 score. Specifically, the compound method achieved 0.7086 test accuracy and 0.7648 F1 score, while the Pos/Neg method achieved 0.7127 and 0.7663 respectively. Despite the minor improvement, both methods fall short of deep learning models in handling nuanced language, highlighting the limitations of lexicon-based approaches for real-world sentiment analysis.

3.2 SVM

3.2.1 SVM Overview

SVM (Support Vector Machine) is a supervised machine learning algorithm used for classification and regression tasks. Unlike rule-based methods like VADER, SVM learns patterns from labeled training data to identify decision boundaries. Its core objective is to find an optimal hyperplane that maximally separates data points of different classes in a high-dimensional space. A key strength of SVM is its use of kernel functions, which enable handling non-linear relationships by implicitly mapping data into higher dimensions. The algorithm's performance is often evaluated by its generalization ability and margin maximization, which reduces overfitting. SVM is widely applied in text classification, image recognition, and bioinformatics due to its robustness in high-dimensional spaces and versatility with diverse datasets.

3.2.2 SVM pre-processing

To enable sentiment classification, the raw Amazon review texts were transformed into structured features through a series of preprocessing steps. **Text normalization** was first applied by converting all characters to lowercase, ensuring uniformity (e.g., “Excellent” became “excellent”). Special characters such as punctuation and symbols were removed using regular expressions, and extra spaces and hyphens were normalized to single spaces.

Next, **tokenization and stemming** prepared the text for feature extraction. A custom tokenizer split the reviews into individual tokens (words), preserving meaningful apostrophes (e.g., ”don't” became [”don”, ”t”]). Common English stopwords (e.g., ”the,” ”and”) were removed to reduce noise, and Porter stemming reduced words to their root forms (e.g., ”running” $\rightarrow$ ”run”), consolidating semantically similar terms.

To convert the preprocessed text into numerical features, **TF-IDF vectorization** was employed. Both unigrams (single words) and bigrams (two-word phrases) were extracted to capture local context, such as the phrase “not bad.” To control dimensionality, only the top 5,000 terms by frequency were retained, excluding extremely rare terms (minimum document frequency of 5) and overly common terms (maximum document frequency of 80%).

3.2.3 SVM Architecture

The LinearSVC algorithm was selected for its speed and suitability for high-dimensional, sparse data produced by TF-IDF. A linear kernel was chosen to maximize computational efficiency and align with the linear structure of the feature space. The regularization parameter was set to the default C = 1.0, balancing margin maximization with classification error minimization. To ensure reproducibility, a random_state value of 42 was used.

Unlike the SVC class from scikit-learn, which supports various kernels (linear, polynomial, radial basis function, etc.) and solves the optimization problem using libsvm, LinearSVC uses liblinear, which is specifically optimized for linear problems. This makes LinearSVC significantly faster and more scalable when dealing with large, sparse datasets—such as TF-IDF matrices typical in text classification. While SVC can provide probability estimates (with probability=True), it becomes computationally expensive and impractical for datasets of this size. In contrast, LinearSVC is designed to handle millions of samples efficiently, albeit without native probability outputs.

The SVM was chosen for its balance between computational efficiency and competitive performance. Training the model on 50,000 samples required under two minutes—a significant advantage compared to the hours needed for deep learning models like BERT. The linear kernel also enabled fast inference, making the approach suitable for real-time sentiment scoring tasks. Additionally, SVMs offer interpretability, with feature coefficients highlighting the most discriminative terms (see Table 1). Lastly, the SVM served as a baseline to benchmark the performance of more advanced models such as LSTM and BERT.

3.3 LSTM

3.3.1 LSTM Overview

Long Short-Term Memory is a type of recurrent neural network (RNN) designed to capture long-range dependencies in sequential data by using memory cells and gating mechanisms to control the flow of information. It addresses the vanishing gradient problem of traditional RNNs, making it effective for tasks like sentiment analysis, where the order and context of words matter. LSTMs are especially good at learning patterns over time, such as understanding emotional tone in text

3.3.2 LSTM pre-processing

The raw Amazon review texts were transformed through a multi-stage pipeline to prepare them for sequence-based deep learning. Each line in the dataset was first parsed into a label and review body, with labels cleaned to extract binary sentiment values (1 for negative, 2 for positive). Text preprocessing followed, where reviews were converted to lowercase, and all punctuation, numbers, emojis, and non-alphabetic characters were removed using regular expressions. This ensured that only meaningful words remained.

After cleaning, tokenization was performed using Keras' Tokenizer, retaining the 1,000 most frequent words in the training data to control vocabulary size and minimize overfitting. All sequences were padded to a fixed length of 100 words, balancing the need to capture review detail while maintaining computational efficiency. The target labels were extracted and stratified, and a validation split (80/20) was applied to the training set to allow early stopping during model training.

3.3.3 LSTM Architecture and Hyperparameters

The model architecture was based on a sequential deep learning pipeline using Long Short-Term Memory (LSTM) layers for their ability to model sequential dependencies. It began with an Embedding layer that projected input tokens into 128-dimensional vectors. This was followed by two stacked Bidirectional LSTM layers—first with 128 units, then 64—each paired with a 0.2 dropout rate to mitigate overfitting. A dense layer with 64 ReLU-activated units added non-linearity, and the final output layer used a sigmoid activation function for binary sentiment prediction.

The model was compiled with the Adam optimizer (default learning rate), binary cross entropy loss, and accuracy as the evaluation metric. Training was performed over 10 epochs with a batch size of 256, using early stopping based on validation loss (patience = 2) to avoid overfitting and restore the best model weights. While the model architecture and training parameters were manually selected, this LSTM was not fine-tuned using pretrained embeddings or transfer learning techniques; it serves as a baseline model with basic manual hyperparameter tuning.

3.3.4 LSTM Performance and Interpretation

The LSTM model was trained on a randomly sampled subset of 50,000 reviews and evaluated on a holdout set of 10,000 reviews. Instead of relying on a fixed decision threshold of 0.5, we searched for

the optimal threshold that maximized the F1 score. The best-performing threshold was found to be 0.37, at which the model achieved a test F1 score of 0.8935. At this threshold, precision and recall were well balanced across both classes, with precision of 0.91 and recall of 0.86 for negative reviews (label 0), and precision of 0.87 and recall of 0.92 for positive reviews (label 1).

```
Optimal threshold: 0.37
Best F1 score: 0.8935
              precision    recall  f1-score   support

           0       0.91      0.86      0.88      4903
           1       0.87      0.92      0.89      5097

    accuracy                           0.89     10000
   macro avg       0.89      0.89      0.89     10000
weighted avg       0.89      0.89      0.89     10000
```

Figure 4: Metrics at optimal threshold (0.37)

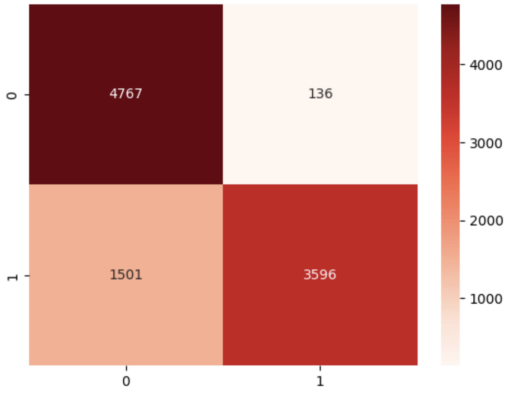


Figure 5: Confusion matrix at threshold 0.37

The overall test accuracy was 89%, with both macro and weighted F1 scores aligning at 0.89. Additionally, the ROC AUC score was 0.9576, indicating strong discriminative ability between positive and negative classes. The confusion matrix confirmed this performance, showing 4,767 true negatives and 3,596 true positives, with moderate misclassifications primarily in the form of false negatives (1,501) rather than false positives (136). These results highlight the model's robustness and its capacity to generalize effectively across varying sentiment expressions in real-world reviews.

3.4 DistilBERT

3.4.1 DistilBERT Overview

DistilBERT is a compact version of BERT developed by HuggingFace through knowledge distillation. While significantly smaller and faster than the original model, it preserves approximately 95% of BERT's performance. Despite having fewer layers and parameters, DistilBERT maintains a compatible architecture that works with the same tokenizers and can handle identical tasks.

Knowledge distillation is an efficient training technique where a smaller "student" model (DistilBERT) learns from a larger "teacher" model (BERT). Rather than training exclusively on hard labels, the student model is trained to mimic the teacher's probability distributions and decision patterns. This approach allows the distilBERT to capture the nuanced reasoning of the BERT while achieving substantial improvements in computational efficiency and model size with minimal performance loss.

3.4.2 DistilBERT Preprocessing

Raw text requires specific preprocessing steps before it can be fed into DistilBERT. The process begins with WordPiece tokenization, which divides text into subword units—for example, breaking "playing" into "play" and "##ing"—allowing the model to handle words not explicitly included in its vocabulary.

The system then inserts special tokens: a [CLS] token at the start of the sequence (used for classification tasks) and [SEP] tokens to mark boundaries between sentence pairs. Each token is subsequently converted to a numerical ID according to the model's predefined vocabulary.

To manage varying text lengths, the preprocessing generates attention masks that help the model distinguish between actual content and padding tokens. Finally, all sequences are standardized to a consistent length by either adding padding tokens or truncating overly long sequences.

3.4.3 DistilBERT Fine-tuning & Architecture

For sentiment classification, a linear layer is positioned above the Transformer architecture to convert the final embedding of the [CLS] token into class-specific logits. During fine-tuning, all parameters—both in the Transformer encoder and classification layer—are jointly optimized using labeled sentiment data.

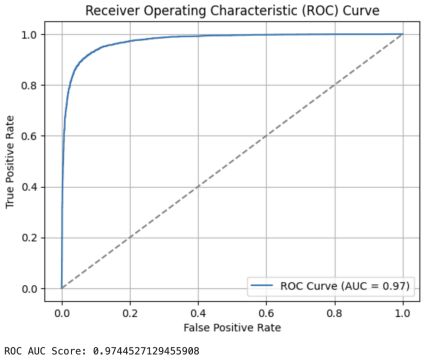
Training proceeds for up to 8 epochs with early stopping (patience = 1) to prevent overfitting by halting when validation loss stops improving. The system uses a batch size of 16 for both training and evaluation phases, automatically restoring the model checkpoint that achieved the lowest validation loss. Evaluation metrics are recorded every 10 steps, with the training explicitly configured to minimize validation loss (greater_is_better=False).

Our performance testing revealed that batch sizes exceeding 16 caused memory-related system failures. With a batch size of 8, the model achieved approximately 90% training accuracy, while increasing to batch size 16 yielded an additional 2% improvement. Given DistilBERT's inherent strength in analyzing short, informal customer reviews, we observed that extending training beyond a single epoch introduced overfitting—validation loss increased significantly while training loss remained artificially low at around 0.01%. Consequently, our optimal configuration (epoch = 1, batch_size = 16) delivered 92% testing accuracy, 97.4% AUC, and a 92% F1 score.

	precision	recall	f1-score	support
0	0.92	0.91	0.92	4875
1	0.92	0.92	0.92	5125
accuracy			0.92	10000
macro avg	0.92	0.92	0.92	10000
weighted avg	0.92	0.92	0.92	10000

Accuracy: 0.9197

Figure 6: Confusion matrix for BERT's best performance



ROC AUC Score: 0.9744527129455908

Figure 7: ROC curve of BERT

4 Model Analysis & Tradeoff

Sentiment analysis techniques fall into three main paradigms, each with unique strengths and tradeoffs.

4.1 Rule-Based Models (VADER)

- **Mechanism**: Applies predefined lexicons (e.g., words scored for positivity/negativity) and grammatical rules (e.g., punctuation, capitalization) to compute sentiment polarity.
- **Strengths**: Requires no training; offers fast inference.
- **Limitations**: Struggles with sarcasm, long texts, and contextual nuance.
- **Example**: The phrase "This product is not bad" might be misclassified as positive because VADER may not adequately account for negation.

4.2 Traditional Machine Learning (SVM)

- **Mechanism**: Converts text into numerical features (TF-IDF vectors) and identifies a hyperplane that best separates sentiment classes.
- **Strengths**: Efficient for smaller datasets; results are interpretable.
- **Limitations**: Ignores word order and context.

- **Example**: The sentences "I love the product but hate the service" and "I hate the product but love the service" would yield identical TF-IDF vectors, potentially leading to misclassification.

4.3 LSTM

- **Mechanism**: Processes text sequentially, using memory cells to retain long-term dependencies.
- **Strengths**: Handles variable-length sequences; mitigates vanishing gradients.
- **Limitations**: Computationally intensive; struggles with very long texts.

4.4 BERT

- **Mechanism**:Employs a transformer architecture to analyze bidirectional context.
- **Strengths**: Delivers state-of-the-art accuracy; comprehends complex syntax and semantics.
- **Limitations**: Demands significant computational resources; slower inference.

5 Experiments & Results

Model	Test Accuracy(%)	F1 Score	ROC AUC
VADER	0.71	0.77	0.70
SVM	0.87	0.87	0.94
LSTM	0.908	0.91	0.97
BERT	0.92	0.92	0.974

Figure 8: Model Performance Comparison.

All four models were trained and evaluated on the 3.6 M / 0.4 M train-test split of the Kaggle Amazon Reviews corpus (binary labels: 1-2= negative, 4-5= positive). We fine-tuned DistilBERT for up to 8 epochs (early-stopping patience = 1, batch-size 16) with AdamW and a linear warm-up schedule; LSTM used a 300-dim embedding layer, a 128-unit LSTM, and a 64-unit ReLU dense layer (batch-size 128, sequence-length 200, vocab = 5 000, optimiser = Adam 1 × 10-3); SVM employed a TF-IDF bag-of-words (max 50 000 n-grams, C = 1, RBF kernel); VADER required no training. All training ran on a single NVIDIA-T4 GPU with mixed-precision enabled, and we tracked Accuracy, F1, and ROC-AUC on the held-out test set.

Among the four models, BERT achieved the best overall performance (92% accuracy, 0.92 F1 score, 0.974 ROC AUC). This highlights its strength in capturing contextual and nuanced sentiment, especially for our Amazon Review Dataset. LSTM also performed well (90.8% accuracy, 0.91 F1 score, 0.97 ROC AUC), particularly in modelling sequential patterns. SVM balanced speed and accuracy but lacked the ability to interpret deeper contextual meanings. As a result, it achieved moderate performance among the four models (87% accuracy, 0.87 F1 score, 0.94 ROC AUC). In contrast, VADER, while fast and requiring no training, underperformed on complex reviews due to its rule-based limitations. It has the lowest performance scores (71% accuracy, 0.77 F1 score, 0.70 ROC AUC).

Despite high performance scores, our failure analysis revealed notable challenges. In BERT, around

44.5% of misclassifications were false negatives, where satisfied reviews were predicted as unsatisfied, and 55.5% were false positives. A primary source of error was the presence of mixed sentiment language. For example, a review praising product quality but criticizing delivery can confuse the models; or some users gave high ratings while expressing strong dissatisfaction in text. This discrepancy between review text and rating points to a key limitation: human sentiment is multifaceted and not always consistently expressed across text and numerical ratings. In some examples, positive reviews with sarcastic or complex language were misclassified as negative, while critical reviews that began with praise misled the classifier into predicting a positive sentiment. These insights emphasize the need for future improvements in handling nuanced expressions and sarcasm, as well as better alignment between textual content and assigned sentiment labels.

6 Conclusion

In this project, we conducted a comprehensive comparison of four sentiment analysis models—VADER, SVM, LSTM, and DistilBERT—on a large-scale Amazon Reviews dataset. Our goal was to evaluate their tradeoffs in terms of accuracy, contextual understanding, and practical deployment.

The results demonstrate that while VADER offers fast, rule-based performance with no training overhead, it falls short on longer and nuanced reviews. SVM strikes a balance between speed and accuracy but cannot capture sequential context. LSTM performs better by leveraging word order and memory but comes with increased training complexity. DistilBERT outperformed all other models in accuracy and F1 score, though it risks overfitting in cases of mixed sentiment and shorter texts.

Through hands-on implementation and evaluation, we gained practical insights into each model's strengths and limitations. Our findings reinforce the importance of aligning model choice with specific business goals—such as real-time analysis, contextual depth, or deployment scalability.

7 Future Discussion

Future work can explore more advanced architectures such as RoBERTa—which improves upon BERT by using dynamic masking rather than static masking—along with hybrid modeling approaches. Additionally, efforts can expand beyond binary classification into multi-class or regression-based sentiment tasks, enhance the handling of multilingual and informal review formats, and improve computational efficiency by implementing models using optimized frameworks like PyTorch or TensorFlow.