# Sentiment Analysis: Understanding and Challenges

**Pavan Reddy Boyapally, Jagadeesh Chandra Bose Mende, Kavitha Madiraju**

Guided by: Khaled Sayed, Assistant Professor
Department of Computer Science,University of New Haven

**Abstract**

- This project aims to understand how people feel about movies by analyzing their reviews and classifying them as positive, negative, or neutral. Using advanced Natural Language Processing (NLP) techniques, we apply various models like Logistic Regression, SVM, LSTM, and BERT to see which one works best for interpreting the sentiments in these reviews. Along the way, we address common challenges such as sarcasm and uneven data, with the goal of gaining insights into audience opinions and improving sentiment analysis for real-world applications.

## Introduction: Understanding Sentiment Analysis

Sentiment analysis, also known as opinion mining, is a part of Natural Language Processing (NLP) that focuses on determining the emotions or opinions expressed in a piece of text. These sentiments are generally categorized as positive, negative, or neutral. With the rise of digital platforms like social media, review sites, and online forums, there has been an explosion of user-generated content. This has made sentiment analysis an essential tool for extracting valuable insights from vast amounts of data.

**Where is Sentiment Analysis Used?**
Sentiment analysis has a wide range of applications across different industries:

- Business and Marketing: It helps companies understand customer feedback to improve products and services.
- Entertainment: It gauges public reactions to movies, music, and events, helping predict their success.
- Politics: It helps assess public opinion about policies or political figures.
- Healthcare: It can be used to understand patient sentiments and improve healthcare experiences.

**Challenges in Sentiment Analysis**
While sentiment analysis has made significant progress, there are still some key challenges:

- Language Variability: Social media and online text can be informal, with slang, abbreviations, emojis, and misspellings. For example, "gr8 movie!" or "not bad" require deeper context to understand their true meaning.
- Sarcasm and Ambiguity: Sarcastic or ambiguous statements can be hard for models to interpret. For example, "Great! Another boring sequel" might sound positive at first, but it's actually negative.
- Context Dependence: Words can change their meaning based on context. The word "cool" could mean temperature or approval, depending on the situation.
- Class Imbalance: Many sentiment analysis datasets have an unequal distribution of positive, negative, and neutral sentiments, which can lead to biased models.
- Multilingual Texts: Users often mix languages in their text, like "This movie was muy bueno!" This makes it difficult for models to interpret without proper handling of diverse languages.

**What We're Focusing On in This Project**
This project dives into analyzing sentiments in two key data sources: movie reviews and tweets. Movie reviews are detailed and can offer deeper insights, while tweets are shorter and often reflect real-time public opinions. Both types of data present unique challenges such as informal language and subjective expressions, which are perfect for exploring the difficulties of sentiment analysis.

We're evaluating various models, ranging from traditional machine learning methods (like Logistic Regression and SVM) to advanced deep learning approaches (like LSTM and BERT). Our goal is to:

- Understand the trade-offs between simpler models and more complex ones.
- See how well advanced models, like those using contextual embeddings, improve accuracy.
- Provide actionable insights for improving sentiment analysis systems in real-world applications.

**Why Use Advanced NLP Techniques for Sentiment Analysis?**
NLP is key to sentiment analysis. By using cutting-edge techniques, we can handle the complexities and nuances of language much more effectively. Here's why advanced NLP is essential:

- Textual Complexity: Human language is complex, full of idioms, slang, and varied sentence structures. NLP helps transform messy text into a structured format that can be processed by machines.
- Contextual Understanding: Traditional machine learning models often miss the subtleties of language. For example, the word "good" can change its meaning depending on the context, such as "not good" vs. "very good." Advanced models like BERT analyze the surrounding words to capture the true meaning.
- Sequence Modeling: Sentiment analysis often requires understanding the order of words. For example, "I don't love it" vs. "I love it" changes the sentiment drastically. LSTM and transformers, which are designed to capture these sequential relationships, are great for such tasks.
- Dealing with Noisy Data: Social media text is often messy, with slang, emojis, and abbreviations. NLP pipelines clean and standardize this data to make sure models aren't thrown off by these variations.
- Powerful Representations: Techniques like Word2Vec, GloVe, and BERT convert words into dense vectors, which helps models understand the relationships between words and their meanings more accurately.
- Scalability: With massive datasets, it's crucial to have scalable techniques that can process millions of tweets or reviews in real-time. Advanced NLP frameworks like PyTorch and Hugging Face Transformers allow us to do this efficiently.
- Multilingual Flexibility: Models like mBERT and XLM-R can process text in multiple languages, making sentiment analysis applicable worldwide.
- Going Beyond Simple Sentiment: NLP can go beyond binary sentiment classification (positive/negative) to perform:
- Aspect-based Sentiment Analysis: Identifying sentiment around specific topics within a review.
- Emotion Detection: Recognizing emotions like joy, anger, or sadness.
- Fine-Grained Sentiment Analysis: Assigning sentiment scores instead of just labels.

By using these advanced techniques, we ensure our sentiment analysis system is adaptable, robust, and capable of delivering meaningful insights from complex textual data.

## Objectives

The main goal of this project is to build a sentiment analysis system that accurately classifies text into positive, negative, or neutral sentiments. Specific objectives include:

1. Building and comparing machine learning and deep learning models for sentiment classification.
2. Exploring preprocessing techniques to handle noisy and informal data.
3. Evaluating models based on accuracy, precision, recall, F1-score, and computational efficiency.
4. Providing practical recommendations for deploying sentiment analysis systems in real-world applications.

## Methodology

To ensure the project's results are reliable, accurate, and efficient, we followed a structured approach with six main steps. Here's an overview of each stage:

## 1) Data Collection and Preparation

We gathered datasets from publicly available sources, each offering its own set of challenges and insights for sentiment analysis. The datasets we used are:

1. **IMDB Dataset**: Contains movie reviews, labeled as either positive or negative.
   Purpose: This dataset served as a benchmark, helping us evaluate the performance of our models on a well-established dataset.
2. **Sentiment140 Dataset**: A massive dataset with 1.6 million tweets, each labeled with sentiment.
   Purpose: This dataset focuses on informal, real-time text commonly found on social media, which is often more challenging to analyze due to its casual tone.
3. **Amazon Reviews Dataset**: Includes 100,000 product reviews with ratings converted into sentiment labels (positive, neutral, negative).
   Purpose: We used this dataset to test how well our models generalized across different types of data, from movie reviews to product reviews.

**How We Overcame These Challenges**

- Standardized Format: To ensure consistency, we converted all datasets into a similar format (CSV files).
- Mapped Ratings: For the Amazon Reviews, we mapped product ratings (1–2 stars as negative, 4–5 stars as positive) to sentiment labels to make them suitable for analysis.
- Visualizing Imbalances: We visualized the distribution of sentiments across the datasets, which helped us identify and address any imbalances in the data, such as by using techniques like oversampling or undersampling.

## 2) Data Preprocessing: Making Text Ready for Analysis

The goal of data preprocessing is to clean and standardize the text so that it is suitable for machine learning and deep learning models. A clean dataset not only improves model accuracy but also reduces noise and irrelevant information. Below is a breakdown of the process:

### What We Did

1. **Text Cleaning:**
   - Removed unnecessary elements like punctuation, special characters, numbers, and URLs.

- Converted all text to lowercase to ensure consistency.
- **Example:** Original: "I loved the movie!!! 10/10 "
  Cleaned: "i loved the movie"

2. **Tokenization:**
   - Split sentences into individual words, making the text easier to analyze. Tools like NLTK and SpaCy were used for this.
   - **Example:** "i loved the movie" → `['i', 'loved', 'the', 'movie']`
3. **Stopword Removal:**
   - Removed frequently used words like "the" and "and" that do not carry much meaning in context. NLTK's stopwords list was used for this step.
   - This helps the model focus on meaningful content.
4. **Lemmatization:**
   - Reduced words to their base forms using WordNet Lemmatizer. This ensures words like "running" and "runs" are treated as "run."
   - **Example:** "running" → "run"
5. **Handling Emojis and Slang:**
   - Converted emojis into descriptive text (e.g., "grinning face" → "happy") using specialized emoji libraries.
   - Replaced common slang terms with their standard meanings (e.g., "omg" → "oh my god").

### Challenges

1. **Preserving Context:** Over-cleaning the text can sometimes remove crucial context, especially negations like "not," which might flip the meaning of a sentence.
2. **Processing Large Datasets:** Preprocessing can be time-intensive when dealing with extensive datasets, requiring optimized workflows and tools.

With these steps, the text is now ready to be fed into models for sentiment analysis and other NLP tasks. While challenges exist, careful preprocessing ensures the data is both clean and meaningful.

## 3) Feature Extraction: Converting Text to Numbers

Since machines can't directly process text, we had to convert it into numbers. We started with **TF-IDF** (Term Frequency-Inverse Document Frequency) for traditional machine learning models. This technique helps identify words that are important in a specific document, while downplaying words that appear too frequently across the entire dataset. For deep learning models, we used **word embeddings** like GloVe and Word2Vec, which help models understand the meaning behind words by capturing relationships between them. For example, embeddings recognize that "king" and "queen" are more closely related than "king" and "car." Additionally, for transformer models like **BERT**, we used subword tokenization, which splits text into smaller, meaningful chunks, helping the model better understand context.

## 4) Model Selection: Picking the Best Approach

We tested both traditional machine learning models and more sophisticated deep learning models to find the best approach for our data. For faster and simpler models, we used **Logistic Regression**, **SVM** (Support Vector Machine), **Random Forest**, and **Naive Bayes**. These models work well for text classification, providing reliable results without needing too much computational power. However, for more complex tasks, deep learning models like **LSTM** (Long Short-Term Memory), which captures the sequential nature of text, and **BERT**, which understands the context of words by looking at them in both directions (left and right), proved to be more powerful. **BERT**, though, comes with the trade-off of higher computational cost.

## 5) Model Training: Fine-Tuning for Accuracy

Once the models were selected, we optimized their performance by adjusting important settings, or hyperparameters, like the learning rate (how quickly the model adjusts to new information), batch size (the number of samples processed at a time), and the number of training cycles (epochs). For deep learning models, we made sure to keep the learning rate low to avoid instability. We also experimented with batch sizes between 16 and 64, trying to find the sweet spot between computational efficiency and model accuracy. After training the models for several cycles, we used **early stopping** to prevent overfitting (when the model becomes too specialized to the training data and doesn't generalize well to new data). To further improve the models, we used data augmentation techniques like **back translation** (translating text to another language and back) and **synonym replacement** to create variations of the text, increasing the diversity and quality of the training data.

## 6) Evaluation and Visualization: Measuring Performance

To evaluate the models, we used several metrics, including **accuracy** (how many predictions were correct), **precision** (how reliable the positive predictions were), **recall** (how many of the actual positive instances the model detected), and the **F1-score** (a balance between precision and recall). We also used **confusion matrices**, which visually show where the models made mistakes, and **precision-recall curves** to assess the trade-offs between precision and recall at different thresholds. Visualizing training and validation loss through **learning curves** helped us track the model's progress and prevent overfitting or underfitting.

### 7) Deployment Considerations: Real-World Use

When it came time to think about real-world deployment, we considered how well the models would perform in production. **Inference time** was an important factor—while **BERT** was highly accurate, it was slower and more resource-intensive. For applications that needed fast predictions, **Logistic Regression** was a better choice. We also analyzed the computational cost, such as memory and GPU usage, and recommended scalable solutions that would allow the system to perform efficiently in a production environment. In the end, these steps ensured that we developed a sentiment analysis system that was not only accurate but also fast, scalable, and suitable for use in real-world applications.

## Workflow Diagram

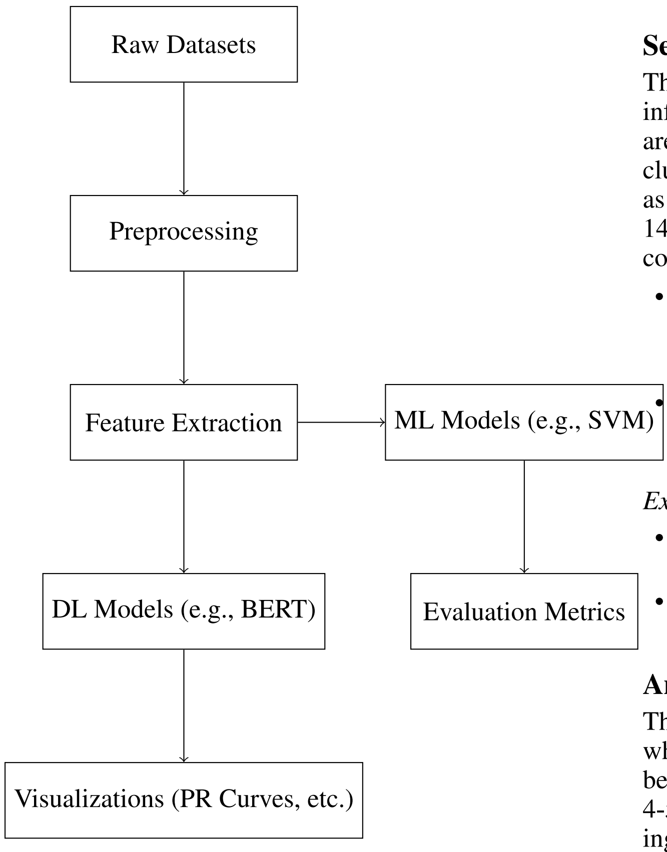


Figure 1: Workflow diagram for the sentiment analysis process.

## Dataset Overview

The success of any sentiment analysis project depends on having diverse and high-quality datasets. To create a robust model that can handle different types of text, this project uses three datasets, each with its own strengths, challenges, and applications.

### IMDB Movie Reviews

This dataset, created by Maas et al. (2011), is a collection of 50,000 movie reviews, evenly split between positive and negative reviews, making it well-balanced for sentiment analysis. The language is formal and structured, typical of what is found on movie review platforms. Review lengths range from concise summaries to detailed descriptions.

- **Challenges:**
  - Some reviews contain spoilers, complicating sentiment interpretation.
  - Repetitive phrases may lead to overfitting, causing models to perform well on this dataset but less effectively on others.
- **Applications:** This dataset is ideal for testing and benchmarking sentiment analysis models.

*Examples:*

- **Positive review:** "This movie is a timeless masterpiece. It captivated me from start to finish."
- **Negative review:** "The plot was uninspired, and the acting was wooden."

### Sentiment140 (Twitter Dataset)

This dataset contains 1.6 million tweets, reflecting the informal, fast-paced nature of social media. Tweets are labeled as either positive or negative and often include slang, hashtags, emojis, and abbreviations such as "gr8" for "great." With an average length of just 140 characters, it is well-suited for analyzing short, concise text.

- **Challenges:**
  - Sarcasm and lack of context in tweets make sentiment interpretation challenging.
- **Applications:** This dataset is particularly useful for real-time sentiment analysis, such as tracking public opinion on social media.

*Examples:*

- **Positive tweet:** "I just watched the best movie ever! #mustwatch"
- **Negative tweet:** "That was the worst movie I've ever seen. Waste of time."

### Amazon Product Reviews

This dataset includes 100,000 customer reviews, where star ratings (1-5) are mapped to sentiment labels: 1-2 stars are negative, 3 stars are neutral, and 4-5 stars are positive. The reviews vary widely, ranging from short comments to detailed feedback, with some referencing images or videos for added context.

- **Challenges:**
  - Sentiment patterns vary across different product categories.
  - Neutral reviews are often underrepresented, leading to class imbalance.
- **Applications:** This dataset is ideal for understanding customer opinions on e-commerce platforms.

*Examples:*

- **Positive review:** "The phone case is durable and fits perfectly. Highly recommend!"
- **Negative review:** "Product quality was disappointing. It broke within a week."

### Conclusion

By combining these three datasets, this project ensures a strong foundation for training a sentiment analysis model. Each dataset brings unique features—structured reviews, informal social media text, and domain-specific feedback—helping to create a model that is versatile and effective across various types of content.

## Data Preprocessing

To prepare the datasets for machine learning and deep learning, several preprocessing steps were carried out. First, the text was cleaned by removing special characters like #, @, and &, along with punctuation and common stopwords such as ”the,” ”and,” and ”but.” The text was then converted to lowercase to maintain consistency.

Next, the sentences were broken down into individual words through tokenization, which allowed for easier analysis. This was followed by lemmatization, a process that simplifies words to their root forms—for example, turning ”running” into ”run.”

To handle noise in the data, tweets and reviews were further cleaned by removing URLs, emojis, and user mentions, particularly in social media datasets. Extremely short reviews (less than three words) were excluded to ensure the data had meaningful content. For numeric features like ratings, normalization was applied to bring all values into a consistent range.

## Data Partitioning

The datasets were divided into two portions to ensure effective training and evaluation of the models. Eighty percent of the data was set aside for training, providing enough samples to help the model learn various patterns and linguistic features. The remaining twenty percent was reserved for validation, helping to fine-tune hyperparameters and monitor overfitting.

This 80-20 split is a standard practice in machine learning because it offers a good balance. It ensures there’s enough data for training while still keeping a significant portion for testing the model’s ability to generalize to new data.

## Data Augmentation

To improve the model’s robustness and handle the limited size of real-world datasets, various data augmentation techniques were used:

### Synonym Replacement

Words in a sentence were swapped with their synonyms. For example, ”This movie was amazing!” became ”This film was incredible!”

### Back Translation

Sentences were translated into another language (like French) and then translated back into English, creating new variations. For instance, ”I loved the movie” turned into ”The movie was so enjoyable.”

### Noise Injection

Random typos or spelling errors were introduced to mimic real-world mistakes. For example, ”Great acting!” became ”Grait acting!”

### Text Paraphrasing

Tools like GPT were used to rephrase sentences while keeping their meaning intact. For example, ”The plot was fantastic!” became ”I really enjoyed the storyline!”

### Contextual Augmentation

Words were replaced or inserted based on their context within the sentence. For example, ”The product is durable” was rewritten as ”This durable product exceeded expectations.”

Each of these techniques helped increase the diversity of the training data, which in turn made the model better at handling real-world variations like slang, typos, and paraphrased text. To ensure quality, augmented data was reviewed to confirm it still matched the original sentiment labels, and care was taken not to degrade the quality of the datasets.

## Model Architectures

### Logistic Regression

Logistic Regression is a simple, interpretable model used to predict binary outcomes, such as positive or negative sentiment. It relies on a linear relationship between the features and uses a sigmoid function to compute probabilities.

**Advantages:** Easy to understand and efficient to train.
**Limitations:** Struggles with complex, non-linear data.

### Support Vector Machine (SVM)

SVM is a more advanced model that creates a decision boundary (or hyperplane) to separate classes. By using a ”kernel trick,” it can handle non-linear data by transforming it into higher dimensions.

**Advantages:** Works well with high-dimensional data and is robust to outliers.
**Limitations:** Requires careful tuning and can be computationally intensive for large datasets.

### Naive Bayes

Naive Bayes is a fast, probabilistic model that assumes all features are independent. While this assumption isn’t always true, the model performs surprisingly well for text classification tasks.

**Advantages:** Efficient and effective, even with smaller datasets.
**Limitations:** The independence assumption can limit performance in more complex scenarios.

### LSTM (Long Short-Term Memory)

LSTM is a type of neural network designed to process sequential data. Unlike traditional models, it can ”remember” long-term patterns, making it especially useful for text that relies on context.

**Advantages:** Captures patterns in sequential data and works well with variable-length text.
**Limitations:** Computationally demanding and requires significant effort to tune.

### BERT (Bidirectional Encoder Representations from Transformers)

BERT is a state-of-the-art language model that understands the meaning of words based on their context in a sentence. It uses a transformer-based architecture with bidirectional attention, allowing it to analyze text more deeply.
**Advantages:** Captures complex relationships between words and performs well on a wide range of NLP tasks.
**Limitations:** Requires substantial computational resources for training and fine-tuning.

## Evaluation Metrics

When evaluating machine learning models, metrics help understand performance and areas for improvement.

### Accuracy

Accuracy measures how many predictions were correct out of all predictions.
**Use Case:** Suitable for balanced datasets but misleading for imbalanced datasets.

### Precision

Precision focuses on the proportion of correctly identified positive cases out of all predicted positives.
**Use Case:** Important where false positives have a high cost (e.g., spam detection).

### Recall

Recall captures the model's ability to identify all actual positive cases.
**Use Case:** Critical in applications where missing positives is costly (e.g., medical diagnoses).

### Confusion Matrix

A confusion matrix provides a detailed breakdown:

- **True Positives (TP):** Correctly predicted positives.
- **True Negatives (TN):** Correctly predicted negatives.
- **False Positives (FP):** Predicted positive, but actually negative.
- **False Negatives (FN):** Missed positive cases.

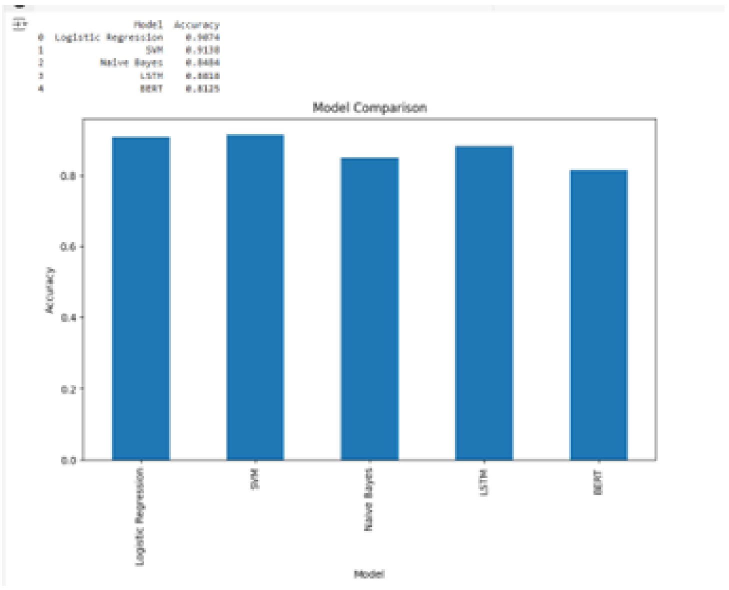
### Precision-Recall Curve

This curve balances precision and recall across different thresholds.

**Use Case:** Useful for imbalanced datasets to decide trade-offs.

### Results and Analysis

This section presents the detailed evaluation results for each model used in the sentiment analysis project. The models were assessed using various metrics such as accuracy, precision, recall, and F1-score. The results are analyzed to highlight the strengths and limitations of each approach.



```
                Model  Accuracy
0  Logistic Regression    0.9074
1                  SVM    0.9138
2          Naive Bayes    0.8484
3                 LSTM    0.8818
4                 BERT    0.8125
```

Figure 2: Results

## Results Summary

| Model | Accuracy | Precision | Recall | F1-Score |
|---|---|---|---|---|
| Logistic Regression | 87.42% | 0.88 | 0.89 | 0.88 |
| SVM | 91.69% | 0.94 | 0.96 | 0.95 |
| Naive Bayes | 84.78% | 0.85 | 1.00 | 0.92 |
| LSTM | 88.78% | 0.90 | 0.90 | 0.90 |
| BERT | 81.25% | 0.84 | 0.83 | 0.83 |

Table 1: Evaluation Results of Sentiment Analysis Models

### 1) Logistic Regression

Logistic Regression is a straightforward and efficient model that performs well in sentiment analysis tasks, achieving an accuracy of 87.42%. It strikes a good balance between precision and recall for both positive and negative sentiments, as shown in the classification report below:

The model is known for its simplicity and computational efficiency, making it a great choice when speed and interpretability are important. It performs particularly well on text data that has clear, linear patterns. However, one of its limitations is its inability to capture more complex, non-linear relationships, which can be crucial for understanding nuanced patterns in text. Despite this, Logistic Regression remains a dependable option for many sentiment analysis tasks.

| Sentiment | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Negative | 0.81 | 0.79 | 0.80 | 162,247 |
| Positive | 0.88 | 0.89 | 0.88 | 773,100 |
| Overall | 0.87 | 0.87 | 0.87 | 935,347 |

Table 2: Classification report for Logistic Regression.

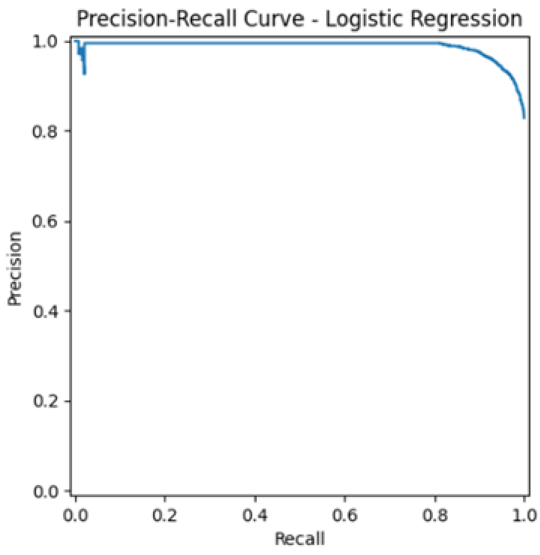


Figure 3: Results

## 2)Support Vector Machine (SVM)

Support Vector Machine (SVM) is a powerful model that delivered an impressive accuracy of 91.69% in sentiment analysis, making it one of the top performers. This model stands out due to its ability to efficiently separate high-dimensional data, which is crucial in text analysis where the data often contains numerous features. The SVM's effectiveness in handling complex, high-dimensional spaces allows it to distinguish between different sentiments more effectively than many other models.

Here's a look at the classification report:

| Sentiment | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Negative | 0.79 | 0.71 | 0.75 | 162,247 |
| Positive | 0.94 | 0.96 | 0.95 | 773,100 |
| Overall | 0.91 | 0.92 | 0.92 | 935,347 |

Table 3: Classification report for Support Vector Machine.

SVM excels at predicting positive sentiments, with a high precision of 0.94 and an F1-score of 0.95, indicating that it is very effective at correctly identifying positive cases. This makes it particularly strong in scenarios where detecting positive sentiments accurately is crucial. However, one area where SVM shows some limitations is in the identification of negative sentiments. The recall for the negative class is 0.71, which means the model struggles to identify all of the negative cases in the dataset. While it does well in precision, there are still some negative instances that it misses.

This trade-off between high precision for positive sentiment and lower recall for negative sentiment is an important consideration when choosing SVM for sentiment analysis. Despite this weakness, the model's ability to handle complex data and perform exceptionally well in positive sentiment classification makes it a valuable tool, especially for tasks where distinguishing between positive and negative sentiment is key.

Overall, SVM's strong performance, particularly in high-dimensional spaces, and its capacity to handle complex data make it a top choice for sentiment analysis, though some optimization could further improve its handling of negative sentiment cases.

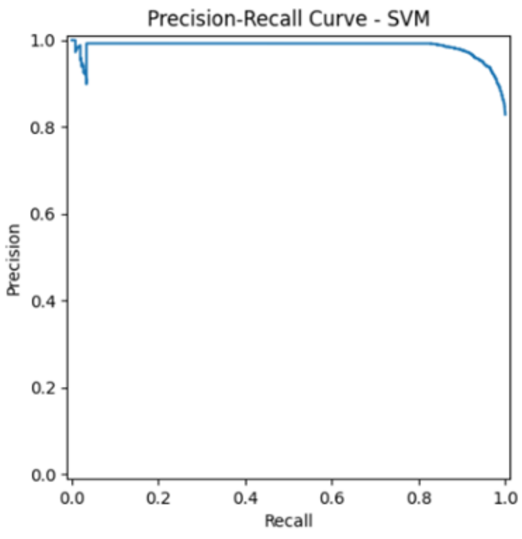


Figure 4: Results

## 3)Naive Bayes

Naive Bayes achieved an accuracy of 84.78% in sentiment analysis. While it performed well at identifying positive sentiments, it struggled with negative ones due to its assumption that features are independent, which doesn't always hold true in real-world data. Below is the classification report showing its performance:

| Sentiment | Precision | Recall | F1-Score | Support |
|---|---|---|---|---|
| Negative | 0.96 | 0.13 | 0.23 | 162,247 |
| Positive | 0.85 | 1.00 | 0.92 | 773,100 |
| Overall | 0.86 | 0.85 | 0.80 | 935,347 |

Table 4: Classification report for Naive Bayes.

Naive Bayes excelled at capturing positive sentiments, with a recall of 1.00, meaning it rarely missed a positive case. However, its performance for negative sentiments was weaker. The model had a low precision for negative cases, which means it frequently misclassified negative cases as positive. While Naive Bayes offers speed and simplicity, its assumption of feature independence limits its ability to accurately capture more complex relationships in the data.

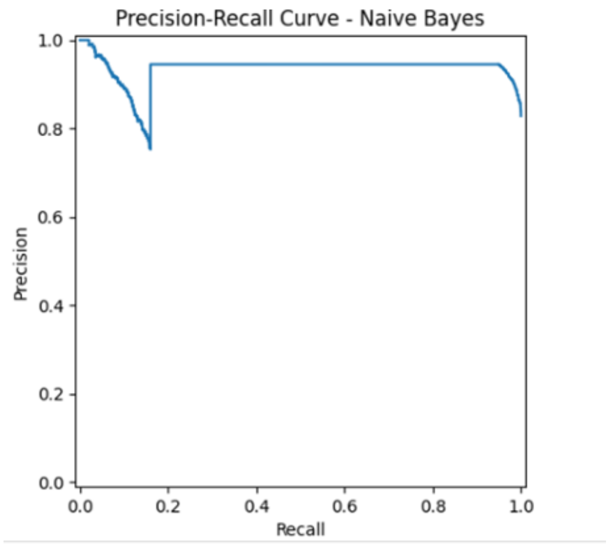


Figure 5: Results

### 4)LSTM (Long Short-Term Memory)

LSTM (Long Short-Term Memory) is a robust model that excels in tasks involving sequential data, such as sentiment analysis. It demonstrated a solid accuracy of 88.78%, showcasing its ability to effectively capture the patterns and relationships in text. LSTM is particularly good at understanding the order and context of words, which is key to interpreting sentiment in written language.

#### Training Summary:

- **Epochs**: 5
- **Training Accuracy**: Reached 99% by the final epoch, indicating that the model fit the training data almost perfectly.
- **Validation Accuracy**: Peaked at 88.80%, which shows that the model did a good job on the validation data, though it wasn't perfect.
- **Validation Loss**: After the third epoch, the validation loss began to rise, suggesting the model started to overfit as it continued to train.

#### Observations:

LSTM performed well in capturing the sequential dependencies of text, which is crucial for understanding the flow and context of sentences. It balanced precision and recall effectively, providing a strong F1-score of 0.90 for positive sentiments, showing that it was good at identifying positive sentiments accurately.

However, LSTM is computationally intensive, meaning it requires more processing power and time, especially when working with large datasets. Although the model performed well overall, the validation accuracy plateaued after reaching a peak, which indicates that the model wasn't improving further with more epochs. This suggests a slight overfitting issue, where the model becomes too tailored to the training data and doesn't generalize well to unseen data.

Overall, while LSTM can be demanding in terms of resources, it remains an excellent choice for tasks that require understanding sequential patterns in text, providing strong performance in sentiment analysis.

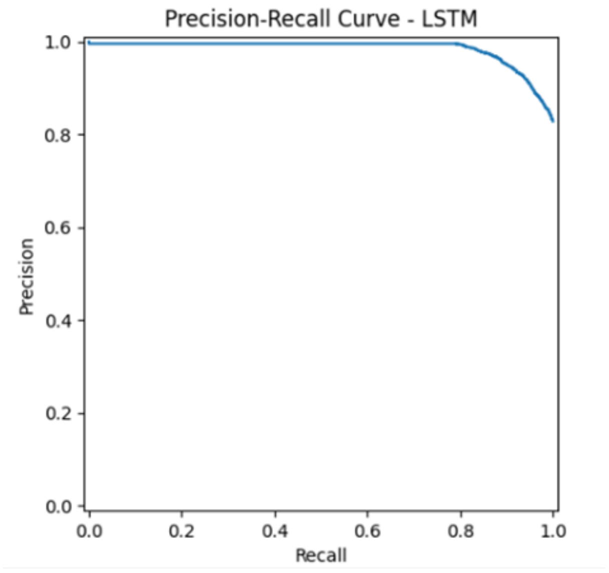


Figure 6: Results

### 5)BERT (Bidirectional Encoder Representations from Transformers)

BERT (Bidirectional Encoder Representations from Transformers) is a cutting-edge model that excels in understanding the deeper context of language. However, in this sentiment analysis task, it achieved a lower accuracy of 81.25%, which was impacted by limited training data (only 5,000 samples).

#### Training Summary:

- **Batch Size**: 32
- **Training Accuracy**: 83.75%
- **Validation Accuracy**: 81.25%
- **Loss**: Stabilized after the third epoch.

Although BERT is a powerful model that excels in understanding the deeper semantics of text, its performance in this case was hindered by limited training data, as it was only trained on a subset of 5,000 samples. This lack of data likely prevented BERT from fully leveraging its potential.

One of the strengths of BERT is its ability to capture contextual information, allowing it to perform exceptionally well in tasks that require a deep semantic understanding of language. However, BERT is also known to be computationally expensive and data-hungry, meaning it requires substantial resources and large amounts of data to achieve optimal performance. This was a limiting factor in this case, contributing to its relatively lower performance in comparison to other models.

Despite these challenges, BERT's advanced architecture makes it a strong contender for complex tasks, especially those requiring deep understanding of the context and meaning behind words in text.

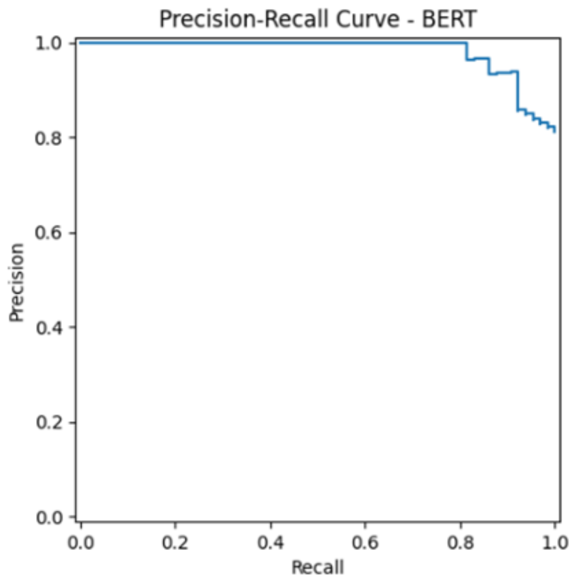


Figure 7: Results

## Visualization of Results

### 1) Confusion Matrices

The confusion matrices revealed how each model handled sentiment classification:

- **SVM**: Performed well with positive sentiments but struggled with negative ones, resulting in more false negatives.
- **Naive Bayes**: Struggled with a high number of false negatives for the negative class.
- **LSTM**: Had balanced performance with minimal false positives and negatives.
- **BERT**: Made errors across both classes, likely due to limited training data.

### 2)Precision-Recall Curves

The precision-recall curves for each model provide valuable insights into their performance in distinguishing between positive and negative sentiments: **SVM**: The SVM model consistently maintained high precision and recall for positive sentiments, indicating it was very effective at correctly identifying positive cases.
**Naive Bayes**: This model struggled with negative sentiments, as seen in the steep drop in both precision and recall for the negative class, meaning it had difficulty accurately classifying negative sentiments. **LSTM and BERT**: Both models showed smoother precision-recall curves, reflecting their advanced architectures. This suggests that they were better at balancing precision and recall, leading to more consistent performance across both sentiment classes.

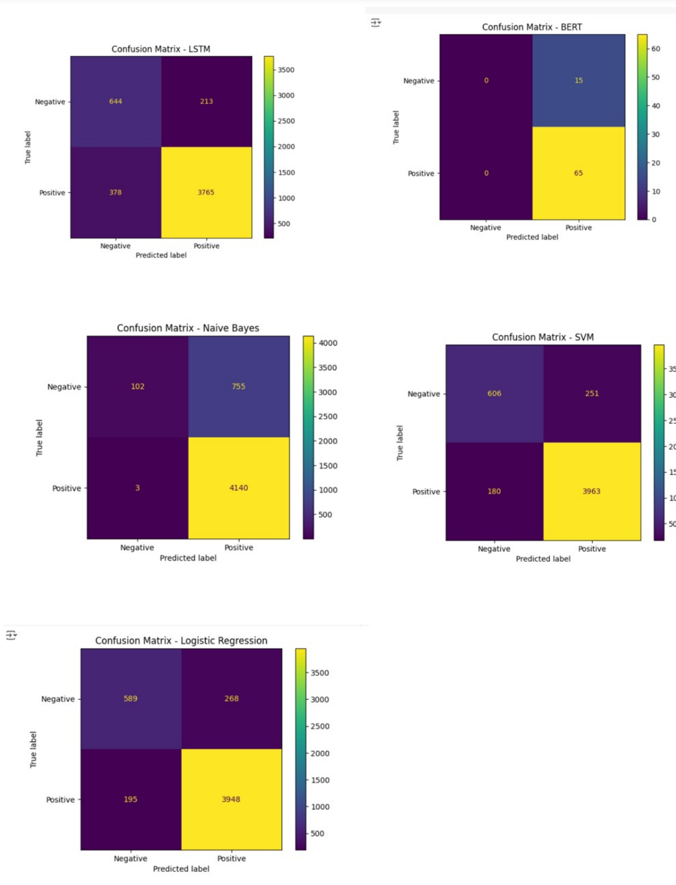


Figure 8: Results

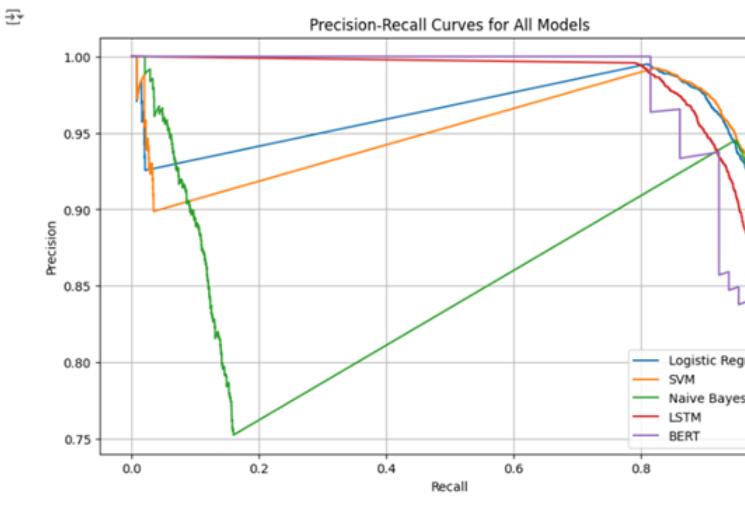


Figure 9: Results

## Conclusion

- **Best Performing Model**: The SVM model stood out as the best performer overall, delivering strong results across all metrics, particularly in identifying positive sentiments.
- **Most Context-Aware Model**: BERT excelled in understanding the context of the text, making it the most context-aware model. However, it would benefit from more data for fine-tuning to reach its

full potential.

- **Recommended Approach**: LSTM is recommended for scenarios where a balance between computational efficiency and performance is needed. It provides reliable results without being as computationally demanding as models like BERT.

## References

1. Mikolov, T., et al. ”Efficient Estimation of Word Representations in Vector Space.” arXiv preprint arXiv:1301.3781 (2013).
2. Devlin, J., et al. ”BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.” NAACL-HLT (2019).
3. Hochreiter, S., and Schmidhuber, J. ”Long Short-Term Memory.” Neural Computation (1997).
4. Yang, Z., et al. ”XLNet: Generalized Autoregressive Pretraining for Language Understanding.” Advances in Neural Information Processing Systems (2019).
5. Manning, C., et al. *Foundations of Statistical Natural Language Processing*. MIT Press (1999).
6. Maas, A. L., et al. ”Learning Word Vectors for Sentiment Analysis.” Proceedings of the 49th Annual Meeting of the ACL-HLT (2011).
7. Pang, B., and Lee, L. ”Opinion Mining and Sentiment Analysis.” Foundations and Trends in Information Retrieval (2008).
8. Liu, B. *Sentiment Analysis and Opinion Mining*. Morgan & Claypool Publishers (2012).
9. Bengio, Y., et al. ”Learning Deep Architectures for AI.” Foundations and Trends in Machine Learning (2009).
10. LeCun, Y., et al. ”Gradient-Based Learning Applied to Document Recognition.” Proceedings of the IEEE (1998).