PROJECT REPORT ON

**Sentiment Analysis on Social Media**



**TEAM**:

1. PAVAN REDDY BOYAPALLY
2. JAGADEESH CHANDRA BOSE MENDE
3. KAVITHA MADIRAJU

**GUIDED BY:**

KHALED SAYED, ASSISTANT PROFESSOR

**I. INTRODUCTION**

**Sentiment Analysis: An Overview**

Sentiment analysis, also referred to as opinion mining, is a field within Natural Language Processing (NLP) that focuses on analyzing textual data to determine the sentiment, opinion, or emotion expressed within the text. This sentiment is typically categorized as positive, negative, or neutral. The rapid growth of digital platforms, such as social media, review sites, and online forums, has resulted in a massive volume of user-generated data, making sentiment analysis a crucial tool for deriving insights.

The application of sentiment analysis spans various domains:

* Business and Marketing: Understanding customer feedback to improve products and services.
* Entertainment: Analyzing public reactions to movies, music, or events.
* Politics: Gauging public opinion on policies or political figures.
* Healthcare: Identifying patient sentiments to enhance healthcare experiences.

Importance of Sentiment Analysis

In today’s digital age, people actively express their opinions online, whether it is through tweets, product reviews, or blog posts. Sentiment analysis helps in transforming these opinions into actionable insights. For instance:

1. Businesses can use it to monitor brand perception and manage customer relationships.
2. Filmmakers and studios can assess public sentiment toward movie trailers or releases, predicting potential success.
3. Social platforms can detect and mitigate negative behaviours, such as hate speech or bullying.

Given the abundance of textual data, manually analyzing sentiments is impractical. Automated sentiment analysis systems leverage machine learning and deep learning models to provide accurate and scalable solutions.

Challenges in Sentiment Analysis

While sentiment analysis has shown remarkable progress, it also presents unique challenges:

1. **Linguistic Variations:**

* Textual data often includes informal language, abbreviations, emojis, and misspellings. For instance, tweets may contain expressions like “gr8 movie!” or “not bad,” which require contextual interpretation.

1. **Sarcasm and Ambiguity:**

* Sarcasm, irony, and ambiguous statements can mislead models. For example, "Great! Another boring sequel" expresses a negative sentiment despite the word "Great."

1. **Contextual Understanding:**

* Words may have different meanings depending on the context. For example, the word "cool" can refer to temperature or approval.

1. **Class Imbalance:**

* In many datasets, positive sentiments may dominate, leading to biased models.

1. **Multilingual Texts:**

* With users expressing opinions in different languages or mixed-language formats (e.g., “This movie was muy bueno!”), models need to handle diverse linguistic inputs.

**Relevance to the Project**

This project focuses on analyzing sentiments in movie reviews and tweets, which are crucial data sources in the entertainment industry. Movie reviews offer detailed critiques of films, while tweets provide real-time, concise opinions. Both types of data exhibit characteristics like informal language, abbreviations, and subjective expressions, making them ideal for exploring sentiment analysis challenges.

The project investigates the performance of various models, from traditional machine learning approaches (e.g., Logistic Regression, SVM) to advanced deep learning architectures (e.g., LSTM, BERT). By evaluating these models on diverse datasets, the project aims to:

1. Understand the trade-offs between model simplicity and accuracy.
2. Explore the effectiveness of contextual embeddings in deep learning.
3. Provide actionable insights for future applications.

**Why Choose Advanced Natural Language Processing Techniques for Sentiment Analysis?**

Natural Language Processing (NLP) is at the core of sentiment analysis, providing the tools and methodologies to understand, process, and extract meaningful information from text. In this project, leveraging state-of-the-art NLP techniques ensures that the sentiment analysis system can handle the complexities and nuances of textual data effectively. Here’s why advanced NLP is essential for sentiment analysis:

1. **Textual Complexity and Linguistic Variability:**

* Human language is inherently complex, with varied sentence structures, idioms, and colloquial expressions.
* NLP techniques are equipped to manage these challenges by preprocessing and transforming raw text into structured formats that are analyse able by algorithms.

1. **Contextual Understanding:**

* Traditional machine learning models often struggle to capture the context in which words are used. For instance, the word “good” can have varying sentiment depending on its surrounding text (“not good” vs. “very good”).
* Advanced NLP models like BERT leverage contextual embeddings to analyze words in the context of their neighbours, improving sentiment prediction accuracy.

1. **Sequence Modelling:**

* Sentiment analysis often requires understanding the sequence of words in a sentence, as word order can drastically change the meaning (e.g., "I don't love it" vs. "I love it").
* Techniques like LSTMs and transformers are designed to handle sequential dependencies, making them ideal for sentiment tasks.

1. **Adaptability to Noisy and Informal Data:**

* Social media data (e.g., tweets) often contains abbreviations, emojis, and slang. NLP pipelines incorporate tokenization, lemmatization, and other preprocessing steps to standardize such data, ensuring robustness in noisy environments.

1. **Rich Representations with Embeddings:**

* Word embeddings (e.g., Word2Vec, GloVe) and contextual embeddings (e.g., BERT) transform raw text into dense vector representations, capturing semantic and syntactic relationships.
* These representations are crucial for models to understand the underlying sentiment accurately.

1. **Scalability for Big Data:**

* Advanced NLP techniques are scalable, allowing for real-time processing of large datasets such as millions of tweets or reviews.
* Optimized algorithms and frameworks (e.g., PyTorch, TensorFlow, Hugging Face Transformers) enable high-performance sentiment analysis even on massive datasets.

1. **Language-Agnostic Capabilities:**

* With pre-trained multilingual models like mBERT and XLM-R, NLP can handle texts in various languages, making it a versatile solution for global sentiment analysis applications.

1. **Insights Beyond Sentiment:**

* NLP techniques can go beyond binary sentiment classification (positive/negative) to perform:
  + - Aspect-based sentiment analysis (e.g., identifying sentiment for specific topics within a review).
    - Emotion detection (e.g., joy, anger, sadness).
    - Fine-grained sentiment analysis (e.g., assigning sentiment scores instead of categories).

By employing cutting-edge NLP techniques, this project ensures that the sentiment analysis system is robust, adaptable, and capable of delivering meaningful insights from complex textual data.

**Objective of the Project**

The primary objective of this project is to develop a sentiment analysis system that accurately classifies textual data into positive or negative sentiments. The specific goals include:

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

**II. METHODOLOGY**

The methodology for this project was developed with a systematic approach to ensure robustness, accuracy, and efficiency. It includes six primary stages, as outlined below:

**1. Data Collection and Preparation**

**Overview**

The datasets for this project were sourced from publicly available repositories, each providing unique challenges and opportunities for sentiment analysis. The datasets include:

1. **IMDB Dataset:**
   * Contains balanced movie reviews labelled as positive or negative.
   * Purpose: Benchmarking performance on a well-known dataset.
2. **Sentiment140 Dataset:**
   * A large dataset of 1.6 million tweets with sentiment labels.
   * Purpose: Analyzing informal text typical of social media.
3. **Amazon Reviews Dataset:**
   * Includes 100,000 product reviews with ratings converted into sentiment categories.
   * Purpose: Testing model generalization across different domains.

**Challenges**

* Imbalanced Data: Some datasets had uneven distributions of sentiment classes, necessitating techniques like oversampling or under sampling.
* Noisy Data: Presence of emojis, abbreviations, and misspellings, particularly in tweets.
* Domain Variance: Sentiment expressions in movie reviews differ significantly from tweets and product reviews.

**Solutions**

* Normalized all datasets into a consistent format (CSV files).
* Mapped ratings to sentiment classes (e.g., Amazon reviews: 1–2 → negative, 4–5 → positive).
* Visualized dataset distributions to identify and address imbalances.

**2. Data Preprocessing**

**Objective**

To clean and standardize the text data, ensuring that it is suitable for machine learning and deep learning models. Proper preprocessing enhances model accuracy and reduces noise.

**Steps**

1. **Text Cleaning:**

* Removed punctuation, special characters, numbers, and URLs.
* Converted all text to lowercase for uniformity.
* Example: *"I loved the movie!!! 10/10 👍"* → *"i loved the movie"*

1. **Tokenization:**

* Split sentences into individual words using tools like NLTK and SpaCy.
* Example: *"i loved the movie"* → ['i', 'loved', 'the', 'movie']

1. **Stopword Removal:**

* Removed common words (e.g., "the", "and") that do not contribute to sentiment.
* Tools used: NLTK's stopwords list.

1. **Lemmatization:**

* Reduced words to their base forms using WordNet Lemmatizer.
* Example: *"running"* → *"run"*

1. **Handling Emojis and Slang:**

* Converted emojis to text using emoji libraries (e.g., 😀 → "happy").
* Replaced common slang terms with their standard equivalents (e.g., "omg" → "oh my god").

**Challenges**

* Maintaining Semantic Meaning: Aggressive cleaning can remove important context (e.g., negations like "not").
* Processing Time: Preprocessing large datasets is computationally intensive.

**3. Feature Extraction**

**Objective**

Convert text data into numerical representations that machine learning and deep learning models can process.

**Techniques**

1. **TF-IDF (Term Frequency-Inverse Document Frequency):**

* Used for traditional machine learning models.
* Captures the importance of words relative to the entire dataset.

1. **Word Embeddings:**

* Used for deep learning models like LSTM and BERT.
* Pre-trained embeddings like GloVe and Word2Vec provide semantic meaning.
* Example:
  + - "king" - "man" + "woman" ≈ "queen"

1. **Tokenization for Transformers:**

* Applied subword tokenization (e.g., BERT tokenizer).
* Splits text into meaningful subwords for context understanding.

**4. Model Selection**

Traditional Machine Learning Models

1. **Logistic Regression:**

* Linear classifier for baseline comparison.

1. **Support Vector Machine (SVM):**

* Effective for sparse data like text.

1. **Random Forest:**

* Ensemble method to handle non-linearities.

1. **Naive Bayes:**

* Probabilistic model for fast text classification.

Deep Learning Models

1. **LSTM (Long Short-Term Memory):**

* Captures sequential dependencies in text.
* Suitable for longer reviews with complex structures.

1. **BERT (Bidirectional Encoder Representations from Transformers):**

* Pre-trained on large corpora for contextual understanding.
* Fine-tuned for sentiment analysis tasks.

**5. Model Training**

Hyperparameters

* **Learning Rate:**
* Optimized using grid search.
* Lower rates for deep learning models to ensure stability.
* **Batch Size**:
* Experimented with sizes ranging from 16 to 64.
* Balanced computational efficiency and generalization.
* **Epochs:**
  + Trained for 10 epochs for LSTM and BERT.
  + Early stopping applied to prevent overfitting.

Augmentation

* Enhanced datasets using back translation and synonym replacement.
* Increased training diversity for better generalization.

**6. Evaluation and Visualization**

**Metrics**

1. Accuracy: Measures overall correctness.
2. Precision: Indicates reliability of positive predictions.
3. Recall: Measures ability to detect all positive instances.
4. F1-Score: Balances precision and recall.
5. Confusion Matrix:

* Visualizes model performance across all classes.

1. Precision-Recall Curve:

* Highlights trade-offs between precision and recall at various thresholds.

**Visualization**

* Plotted learning curves for training and validation losses.
* Generated heatmaps of confusion matrices to identify errors.

**7. Deployment Considerations**

**Objective**

Optimize the system for real-world applications, ensuring scalability and efficiency.

**Key Considerations**

1. **Inference Time:**

* Evaluated prediction time for each model.
* BERT was slower but highly accurate; Logistic Regression offered faster predictions.

1. **Computational Cost:**

* Analyzed memory and GPU usage.
* Recommended scalable solutions for production environments.

## **Workflow Diagram:**

+-----------------------+ +--------------------+ +---------------------+  
| Raw Datasets | --> | Preprocessing | --> | Feature Extraction |  
+-----------------------+ +--------------------+ +---------------------+  
 |  
 v  
 +--------------------+ +--------------------+  
 | Machine Learning |--> | Evaluation Metrics |  
 | Models (e.g., SVM) | +--------------------+  
 +--------------------+  
 |  
 v  
 +--------------------+ +--------------------+  
 | Deep Learning |--> | Visualizations |  
 | Models (e.g., BERT)| | (PR Curves, etc.) |  
 +--------------------+ +--------------------+

**III. DATASET**

The success of a sentiment analysis project is largely dependent on the quality and diversity of the datasets used. This project utilizes three datasets to ensure robustness, variability, and generalization across different types of text.

**Dataset 1: IMDB Movie Reviews**

* Source: The Large Movie Review Dataset by Maas et al. (2011) [6].
* Size: 50,000 reviews.
* Format: Balanced classes, with 25,000 positive and 25,000 negative reviews.
* Characteristics:
  + Formal and structured language, typical of review platforms.
  + Review lengths vary from a few sentences to detailed paragraphs.
* Challenges:
  + Some reviews include spoilers, which may affect sentiment interpretation.
  + Repetitive patterns in text could lead to overfitting for simpler models.
* Applications:
  + Ideal for benchmarking sentiment analysis algorithms due to its structured nature.
* Example Entries:
* Review: "This movie is a timeless masterpiece. It captivated me from start to finish."
* Sentiment: Positive
* Review: "The plot was uninspired and the acting was wooden."
* Sentiment: Negative

**Dataset 2: Sentiment140 (Twitter Dataset)**

* Source: Stanford University's Sentiment140 Dataset [9].
* Size: 1.6 million tweets.
* Format:
  + Labels: 0 (negative), 4 (positive).
  + Each record includes tweet text, user information, and metadata.
* Characteristics:
  + Informal and noisy text, reflective of social media language.
  + Includes hashtags, emojis, URLs, and user mentions.
  + Average tweet length: 140 characters.
* Challenges:
  + Noisy data with abbreviations ("gr8" for "great"), slang, and emojis.
  + Ambiguity due to sarcasm and contextual dependence.
  + Limited context: Tweets often lack sufficient background information.
* Applications:
* Real-time analysis of public opinion on social media.
* Example Entries:
* Tweet: "I just watched the best movie ever! #mustwatch"
* Sentiment: Positive
* Tweet: "That was the worst movie I've ever seen. Waste of time."
* Sentiment: Negative

**Dataset 3: Amazon Product Reviews**

* Source: Amazon customer review data [10]
* Size: 100,000 reviews.
* Format:
  + Ratings: 1-5 stars, mapped to sentiments:
    - Ratings 1-2: Negative.
    - Rating 3: Neutral.
    - Ratings 4-5: Positive.
* Each record includes product ID, user review, and timestamp.
* Characteristics:
  + Text diversity: Reviews range from short comments to detailed opinions.
  + Multimodal content: Some reviews include references to images or videos.
* Challenges:
  + Domain specificity: Sentiment patterns vary across product categories.
  + Imbalance: Neutral reviews are often underrepresented.
* Applications:
  + Evaluating sentiment in e-commerce platforms.
* Example Entries:
* Review: "The phone case is durable and fits perfectly. Highly recommend!"
* Sentiment: Positive
* Review: "Product quality was disappointing. It broke within a week."
* Sentiment: Negative

**Data Preprocessing**

To ensure the datasets were suitable for machine learning and deep learning models, the following preprocessing steps were applied:

1. Text Cleaning:
   * Removed special characters (e.g., #, @, &) and punctuation.
   * Standardized text to lowercase.
   * Removed stopwords (e.g., "the", "and", "but").
2. Tokenization:
   * Split sentences into words for analysis.
3. Lemmatization:
   * Reduced words to their root forms (e.g., "running" → "run").
4. Noise Handling:
   * Removed URLs, emojis, and user mentions in tweets.
   * Filtered out short reviews with less than three words.
5. Normalization:

* Scaled numeric features (e.g., ratings) to a consistent range.

**Data Partitioning**

The datasets were partitioned to balance training, validation, and testing:

1. Training Set (80%):
   * Used for model training.
   * Contains diverse samples to capture a wide range of linguistic features.
2. Validation Set (20%):
   * Used for hyperparameter tuning and monitoring overfitting during training.

**Rationale for the Split**

The 80-20 split is a widely used standard in machine learning as it:

* Provides sufficient data for training without compromising validation and testing.
* Balances the trade-off between model performance and generalization.

**Data Augmentation**

To increase the diversity and robustness of the datasets, the following augmentation techniques were applied:

1. **Synonym Replacement:**
   * Replaced words with their synonyms using NLP libraries (e.g., NLTK, SpaCy).
   * Example:
     + Original: "This movie was amazing!"
     + Augmented: "This film was incredible!"
2. **Back Translation:**

* Translated text into another language (e.g., French) and back to English to create variations.
* Example:
  + - Original: "I loved the movie."
    - Augmented: "The movie was so enjoyable."

1. **Noise Injection:**
   * Introduced random spelling errors and character swaps to simulate real-world typos.
   * Example:
     + Original: "Great acting!"
     + Augmented: "Grait acting!"
2. **Text Paraphrasing:**
   * Used GPT-based tools to rephrase sentences while retaining meaning.
   * Example:
     + Original: "The plot was fantastic!"
     + Augmented: "I really enjoyed the storyline!"
3. **Contextual Augmentation:**

* Inserted or replaced words based on contextual embeddings.
* Example:
  + - Original: "The product is durable."
    - Augmented: "This durable product exceeded expectations."

**Data Augmentation Techniques**

Data augmentation is a vital step in Natural Language Processing (NLP) projects, especially for sentiment analysis, to improve model generalization and robustness. By generating additional samples from existing data, augmentation compensates for limited dataset sizes, mitigates overfitting, and introduces diversity in the training data.

**Why Data Augmentation?**

1. Handling Limited Data: Real-world datasets often lack sufficient size to train robust machine learning and deep learning models. Augmentation creates synthetic variations, effectively increasing the dataset.
2. Improving Generalization: Models trained on augmented data are better equipped to generalize to unseen data, reducing overfitting.
3. Simulating Real-world Scenarios: Techniques like noise injection and paraphrasing mimic variations observed in user-generated text (e.g., typos, slang).

**Augmentation Process in the Project**

1. **Pre-Augmentation Analysis:**
   * Dataset inspection to identify under-represented classes and text variations.
   * Selection of augmentation techniques suitable for informal and formal text (e.g., tweets and reviews).
2. **Implementation:**
   * Synonym replacement, back translation, and paraphrasing were prioritized for their semantic fidelity.
   * Noise injection and token swapping were applied sparingly to avoid degradation in data quality.
3. **Post-Augmentation Validation:**
   * Ensured that augmented data preserved original sentiment labels.
   * Sampled augmented texts were reviewed manually to confirm relevance and coherence.

**IV. MODEL ARCHITECTURES**

**1. Logistic Regression**

* Description: Logistic Regression is a linear model that predicts the probability of a binary outcome using a logistic function.
* Architecture:
  + Input Layer: Represents feature vectors (e.g., TF-IDF scores).
    - sigmoid function: P(y=1|X) = 1 / (1 + e^-(w^T X + b))
  + Output Layer: Produces probabilities for binary classification (positive/negative).
* Advantages:
  + Simple and interpretable.
  + Computationally efficient.
* Limitations:
  + Performs poorly on non-linear data.
  + Assumes linear decision boundaries.

**2. Support Vector Machine (SVM)**

* Description: SVM constructs a hyperplane in a high-dimensional space to separate data into classes.
* Architecture:
  + Kernel Trick: Transforms data into a higher dimension to find the optimal hyperplane.
  + Margin Maximization: Ensures maximum separation between classes.
* Advantages:
  + Effective for high-dimensional data.
  + Robust to outliers.
* Limitations:
  + Requires careful parameter tuning (e.g., kernel type, regularization).
  + Computationally intensive for large datasets.

**3. Naive Bayes**

* Description: A probabilistic classifier based on Bayes' theorem, assuming feature independence.
* Architecture:
  + Probability Calculation: Computes conditional probabilities: P(c∣x)∝P(x∣c)P(c)P(c|x) \propto P(x|c)P(c)P(c∣x)∝P(x∣c)P(c)
  + Maximum Likelihood Estimation: Assigns class labels based on highest posterior probability.
* Advantages:
  + Fast and effective for text classification.
  + Works well with small datasets.
* Limitations:
  + Assumes feature independence, which may not hold in real-world scenarios.

**4. LSTM (Long Short-Term Memory)**

* Description: A type of recurrent neural network (RNN) that addresses the vanishing gradient problem by using memory cells to capture long-term dependencies.
* Architecture:
  + Input Embedding Layer: Converts words into dense vector representations.
  + LSTM Units: Use gating mechanisms to manage information flow:
    - Forget Gate: Decides which information to discard.
    - Input Gate: Determines which new information to add.
    - Output Gate: Controls the final output.
* Fully Connected Layer: Maps LSTM outputs to sentiment classes.
* Advantages:
  + Captures sequential patterns effectively.
  + Handles variable-length text inputs.
* Limitations:
  + Computationally intensive.
  + Requires extensive tuning of hyperparameters.

**5. BERT (Bidirectional Encoder Representations from Transformers)**

* Description: A transformer-based model that pre-trains deep bidirectional representations by jointly conditioning on left and right contexts.
* Architecture:
  + Input Tokenizer: Splits text into tokens and adds special tokens ([CLS], [SEP]).
  + Transformer Encoder Layers:
    - Multi-Head Attention: Captures relationships between words in all directions.
    - Feedforward Network: Processes information at each token position.
  + Classification Head: Fine-tunes the pre-trained embeddings for sentiment classification.
* Advantages:
  + Captures complex linguistic features.
  + Pre-trained on large corpora, reducing training time.
* Limitations:
  + Requires significant computational resources.

# Comparison Table

|  |  |  |
| --- | --- | --- |
| Model | Strengths | Limitations |
| Logistic Regression | Simple, interpretable | Limited for non-linear data |
| SVM | Robust for high-dimensional data | Expensive for large datasets |
| Naive Bayes | Fast and efficient for small datasets | Assumes feature independence |
| LSTM | Captures sequential patterns | Computationally intensive |
| BERT | State-of-the-art contextual understanding | Requires extensive computational power |

**V. EVALUATION METRICS**

Evaluation metrics assess the performance of each model, providing insights into their strengths and weaknesses. Below are detailed descriptions and practical implications of each metric.

**1. Accuracy**

* Definition: Measures the proportion of correctly classified samples out of the total samples.
* Formula: Accuracy = (True Positives + True Negatives) / Total Instances
* Use Case:
  + Best for balanced datasets.
  + May not provide meaningful insights for imbalanced datasets.

**2. Precision**

* Definition: Indicates the proportion of true positive predictions among all positive predictions.
* Formula: Precision = True Positives / (True Positives + False Positives)
* Use Case:
  + Important when false positives have a high cost (e.g., spam detection).

**3. Recall**

* Definition: Measures the model’s ability to correctly identify all positive samples.
* Formula: Recall = True Positives / (True Positives + False Negatives)
* Use Case:
  + Crucial when missing positive cases is costly (e.g., medical diagnosis).

**4. Confusion Matrix**

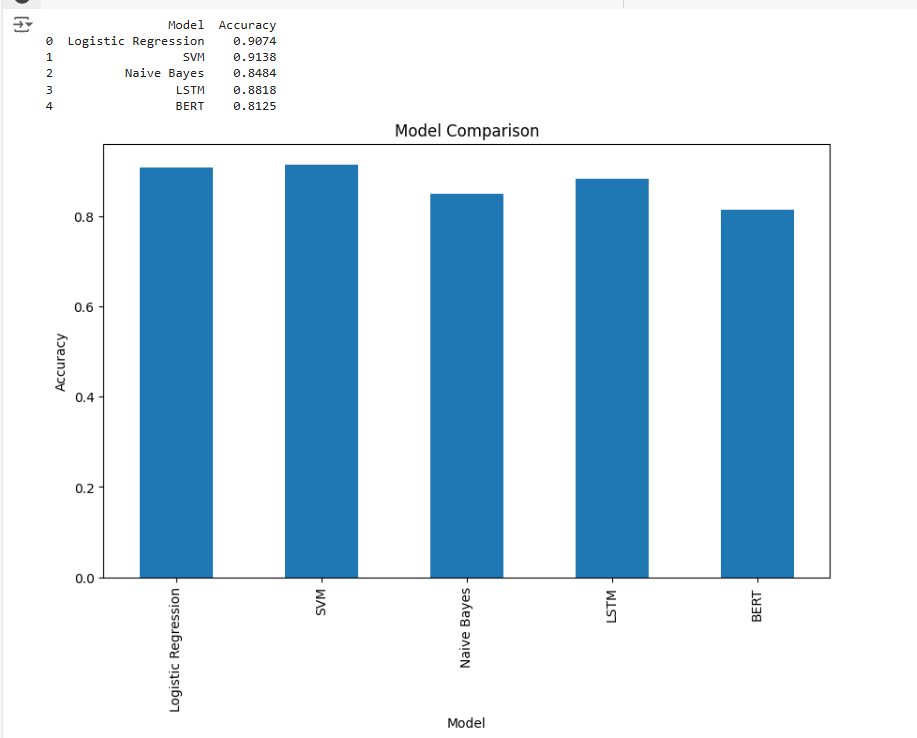
* Definition: A tabular representation of model predictions against true labels.
* Structure:
  + True Positives (TP): Correctly predicted positive samples.
  + True Negatives (TN): Correctly predicted negative samples.
  + False Positives (FP): Incorrectly predicted positive samples.
  + False Negatives (FN): Missed positive samples.
* Visualization:
  + Highlights model strengths and areas of confusion.

**5. Precision-Recall Curve**

* Definition: Plots precision against recall at various probability thresholds.
* Use Case:
* Evaluates performance for imbalanced datasets.

**VI. 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.



**Results Summary**

| **Model** | **Accuracy** | **Precision (Positive)** | **Recall (Positive)** | **F1-Score (Positive)** |
| --- | --- | --- | --- | --- |
| 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 |

* + 1. **Logistic Regression**

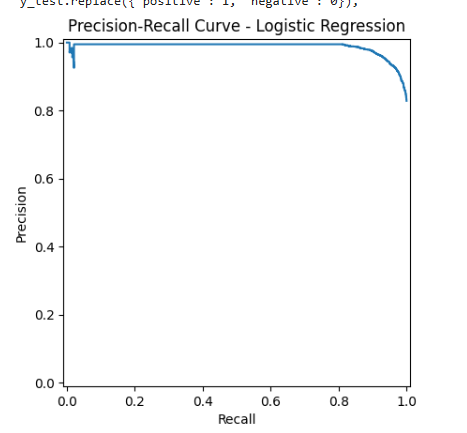
Accuracy: 87.42%

Classification Report:

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

Observations:

* Logistic Regression provides reliable performance with balanced precision and recall for both positive and negative sentiments.
* Strengths:
  + Simplicity and computational efficiency.
  + Performs well on text data with linear separability.
* Weaknesses:
  + Struggles with non-linear relationships, limiting its ability to capture nuanced patterns in text.

****

* + 1. **Support Vector Machine (SVM)**

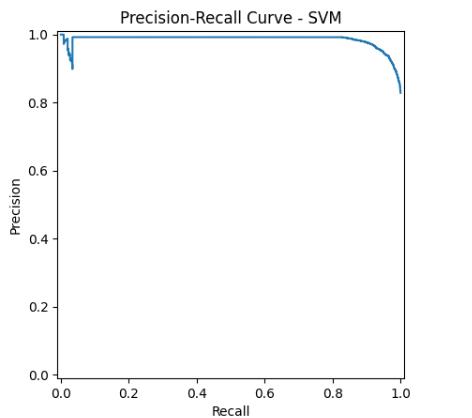
Accuracy: 91.69%

Classification Report:

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

**Observations:**

* SVM achieved high accuracy due to its ability to effectively separate high-dimensional data.
* Strengths: Strong performance in predicting positive sentiments with a high precision of 0.94.
* Weaknesses: Lower recall for negative sentiments indicates difficulty in identifying all negative cases.

****

* + 1. **Naive Bayes**

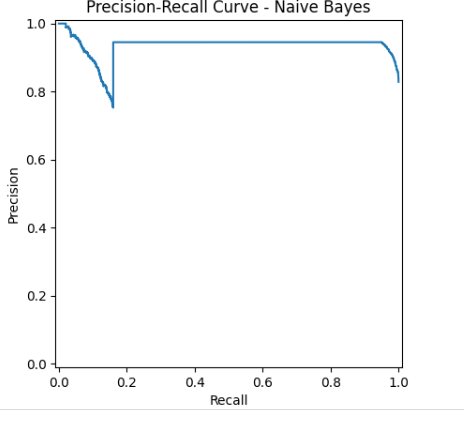
Accuracy: 84.78%

Classification Report:

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

**Observations:**

* Naive Bayes excelled in identifying positive sentiments but struggled with negative ones due to its simplistic assumption of feature independence.
* Strengths: High recall for positive sentiments ensures minimal missed positive predictions.
* Weaknesses: Precision for negative sentiments is low, indicating frequent false positives for this class.

****

* + 1. **LSTM (Long Short-Term Memory)**

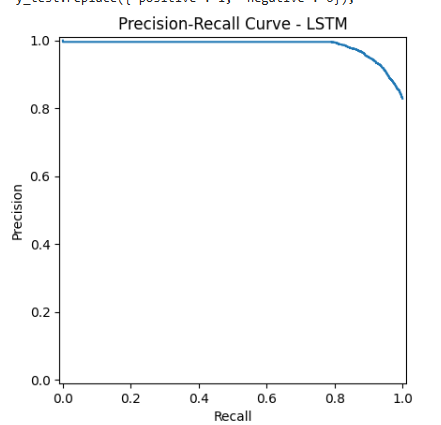
Accuracy: 88.78%

Training Summary:

* Epochs: 5
* Training Accuracy: Reached 99% at the final epoch.
* Validation Accuracy: Peaked at 88.80%.
* Validation Loss: Increased after epoch 3, indicating slight overfitting.

Observations:

* LSTM demonstrated robust performance in capturing sequential dependencies in text.
* Strengths: Balanced precision and recall, providing a reliable F1-score of 0.90 for positive sentiments.
* Weaknesses: Computationally intensive; validation accuracy plateaued at higher epochs.

****

* + 1. **BERT (Bidirectional Encoder Representations from Transformers)**

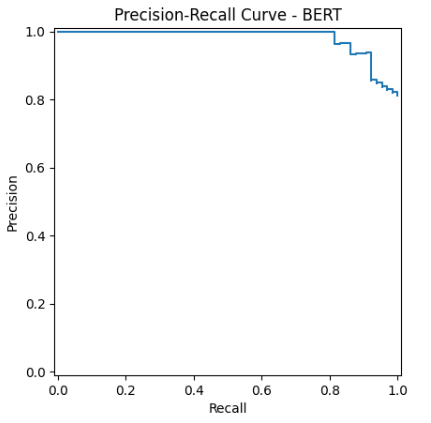
Accuracy: 81.25%

Training Summary:

* Batch Size: 32
* Training Accuracy: 83.75%
* Validation Accuracy: 81.25%
* Loss: Stable after epoch 3.

Observations:

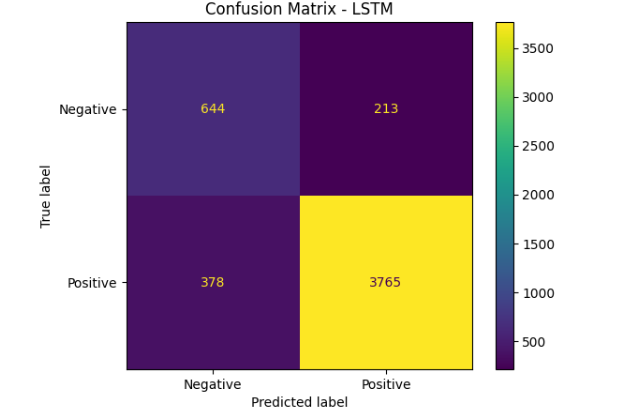
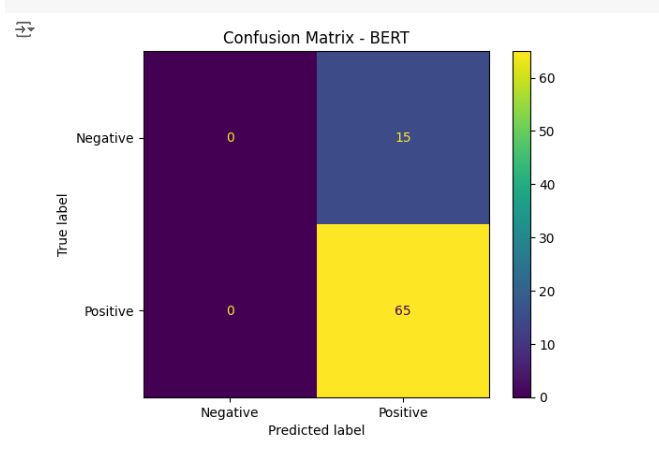
* Despite its state-of-the-art architecture, BERT underperformed due to limited training data (subset of 5,000 samples).
* Strengths: Captures contextual nuances effectively, excelling in scenarios requiring semantic understanding.
* Weaknesses: Computationally expensive and data-hungry.

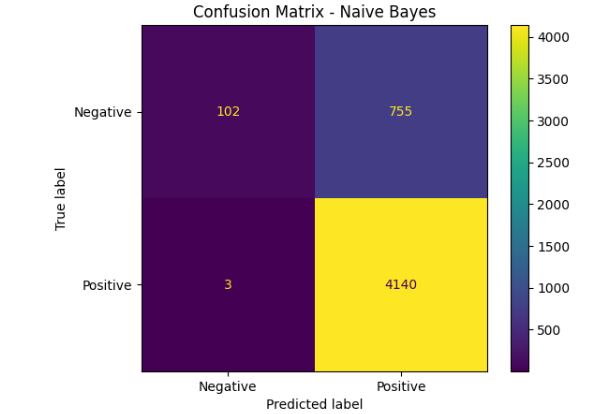
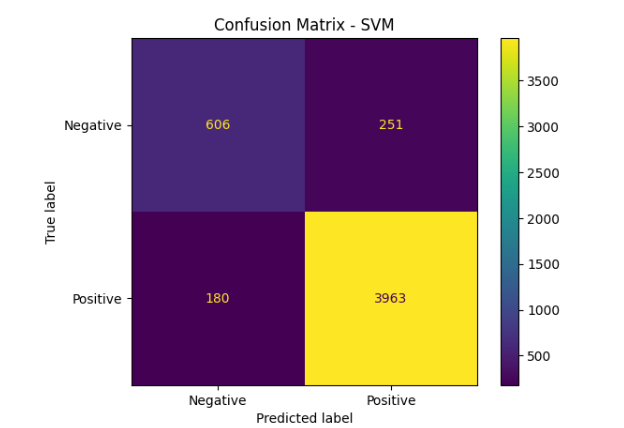
****

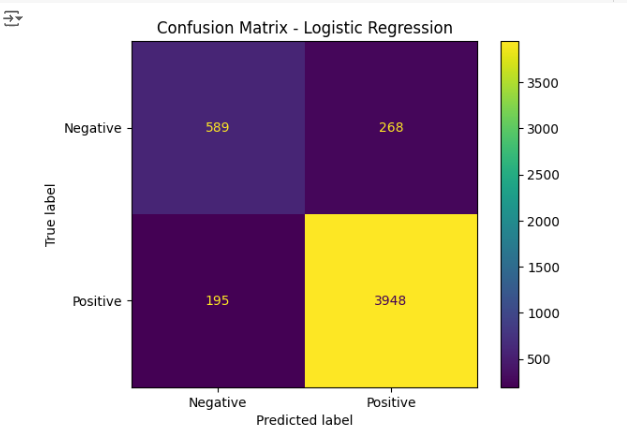
**Visualization of Results**

**1. Confusion Matrices**

* **SVM**: Showed high true positives for positive sentiments but struggled with negatives.
* **Naive Bayes**: Overwhelming false negatives for the negative class.
* **LSTM**: Balanced performance with minimal false positives and negatives.
* **BERT**: Errors evenly distributed across both classes due to limited training data.

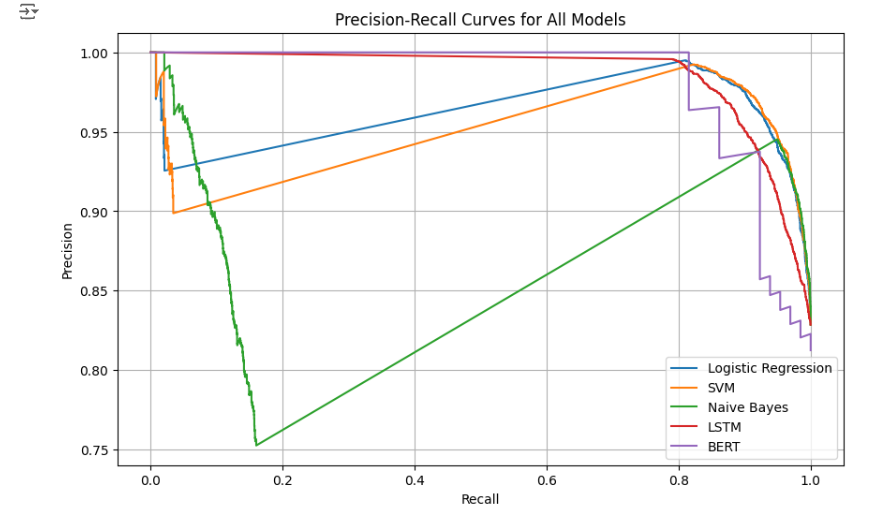
 



**2. Precision-Recall Curves**

* SVM maintained consistently high precision and recall for positive sentiments.
* Naive Bayes exhibited steep precision-recall drops for negative sentiments.
* LSTM and BERT achieved smoother curves due to their advanced architectures.



**VII. Conclusion**

* **Best Performing Model**: SVM, with its robust performance across metrics.
* **Most Context-Aware Model**: BERT, despite requiring more data for fine-tuning.
* **Recommended Approach**: LSTM for scenarios needing balance between computational efficiency and performance.

**VII. 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).