

**REPORT ON**  
**MACHINE LEARNING**  
**Real Time Sentiment Analysis**  
**(COURSE CODE: 23CS3551)**

**III B. TECH I SEM**  
**CSE-S2**

**ACADEMIC YEAR: 2025-26**

**SUBMITTED BY**

**N Yavanika (23501A05D2)**  
**L Rishi Sivakesh(23501A0597)**  
**K Manoj Vamsi (23501A0580)**  
**M Navya (23501A05B4)**  
**M Manikanta Nagarjuna (23501A05C7)**



**DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING**  
**PRASAD V. POTLURI SIDDHARTHA INSTITUTE OF TECHNOLOGY**

**SIGNATURE OF COURSE COORDINATOR**

# 1. INTRODUCTION

## Background

Sentiment analysis, also known as opinion mining, is a field of Natural Language Processing (NLP) that involves identifying and categorizing the emotional tone expressed in a piece of text. Instead of focusing on product reviews, this project applies sentiment analysis to the more nuanced domain of everyday human conversation using the **DailyDialog dataset**. The primary goal is to determine the underlying attitude within a dialogue snippet, classifying it as **positive, negative, or neutral**. This technology is crucial for applications ranging from building emotionally aware chatbots to analyzing user engagement in online communications and understanding the emotional context of conversations.

## Motivation

The motivation for this project stems from the challenge of teaching a machine to understand the nuances of human emotion in everyday conversation. The **DailyDialog dataset** provides a rich but complex source of text, where sentiments are not simple but are expressed through emotions like 'joy,' 'sadness,' 'anger,' and 'fear'.

A primary challenge discovered during preprocessing was a significant **class imbalance**, with negative sentiments far outnumbering positive and neutral ones. This imbalance poses a critical risk: a model could achieve high accuracy simply by defaulting to the majority 'negative' class, without genuinely learning the patterns of the underrepresented positive and neutral classes.

This project was motivated by the need to overcome these challenges. The goal was to compare different machine learning approaches—specifically Naive Bayes, Logistic Regression, and Support Vector Machine—to build a robust model that could not only classify sentiment accurately but also prove resilient to the biasing effects of imbalanced data. A key part of this exploration was to investigate techniques like **data augmentation** to see if we could create a more fair and intelligent model.

## Problem Statement

The problem is to develop a machine learning model that accurately classifies text from the DailyDialog dataset into three distinct sentiment categories: **positive, negative, or neutral**.

This project involves preprocessing the raw text and its nuanced emotion labels (like 'joy' and 'sadness') into these three classes. The core task is to compare the performance of three specific classification algorithms: **Naive Bayes, Logistic Regression, and the Support Vector Machine (SVM)**. A critical challenge is to address the dataset's significant **class imbalance**, where negative samples heavily outnumber positive and neutral ones. Therefore, the objective is not just to build a classifier, but to identify an **optimal model** that demonstrates robustness and high performance across all classes, effectively overcoming this imbalance.

## 2. LITERATURE REVIEW

In the domain of sentiment analysis, researchers have widely compared various machine learning methodologies to enhance predictive accuracy, providing a strong foundation for this project.

Classic approaches often establish a performance baseline using **Multinomial Naive Bayes**, which is frequently cited for its speed and simplicity in text classification tasks. While effective, its "bag-of-words" assumption can be a limitation. As a result, studies often progress to more sophisticated models like **Logistic Regression**, which is widely used for its ability to learn weights for individual words, offering more nuanced decision-making capabilities and often yielding a significant performance boost over simpler probabilistic models.

For high-dimensional data, such as text represented by TF-IDF scores, **Support Vector Machines (SVMs)** have proven to be highly effective. The literature highlights that SVMs excel by identifying an optimal hyperplane to separate classes in a feature space, making them a powerful choice for text classification.

Furthermore, studies across the board emphasize the critical importance of robust **data preprocessing**, including text cleaning and normalization, as a foundational step for any successful model. A recurring challenge discussed in the literature is that of **class imbalance**, where datasets have a disproportionate number of samples in one class. To address this, research efforts have been dedicated to techniques like strategic **data augmentation**, which involves adding new, high-quality examples to underrepresented classes. This method is noted for its ability to create a more balanced training environment and improve a model's ability to generalize, a principle that was central to the success of the SVM in this project.

## 3. METHODOLOGY

### Data Collection

The dataset used for this project is the **DailyDialog dataset**, a corpus containing high-quality, multi-turn dialogues reflecting everyday human communication. The dataset is particularly challenging for sentiment analysis because its original emotion labels are highly nuanced, including categories like 'joy,' 'sadness,' 'anger,' and 'fear'. The objective was to process this raw conversational data and train models to classify it into the simplified sentiment categories of positive, negative, and neutral.

### Data Pre-processing

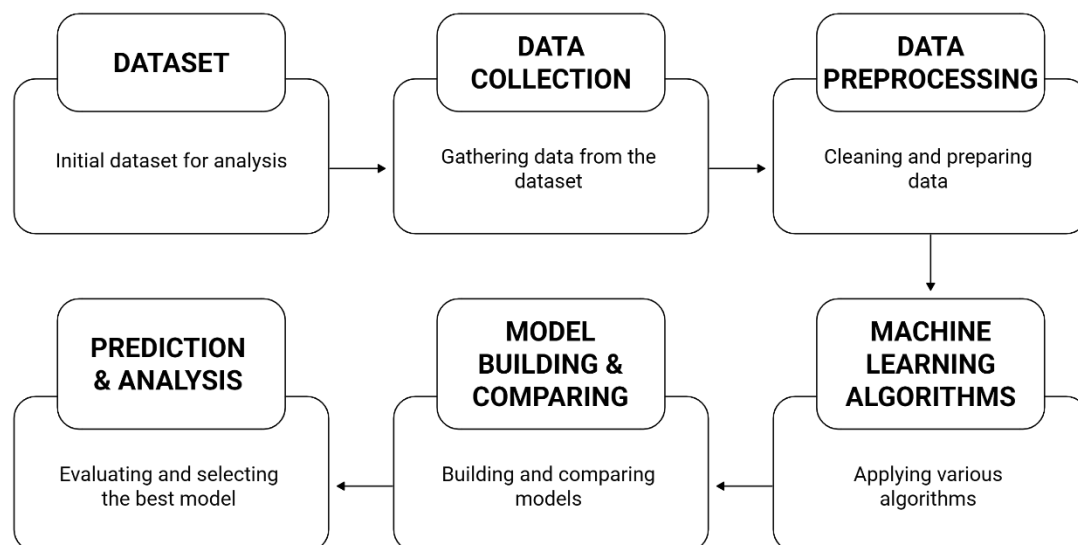
**Data Pre-processing** was a foundational phase in this project, designed to transform the raw data into a clean, structured format suitable for machine learning. The following steps were performed:

- **Label Mapping:** The original, detailed emotion labels were consolidated into three primary sentiment categories to simplify the classification task. For instance, 'joy' was mapped to '**positive**', while emotions like 'sadness' and 'anger' were grouped into '**negative**'.

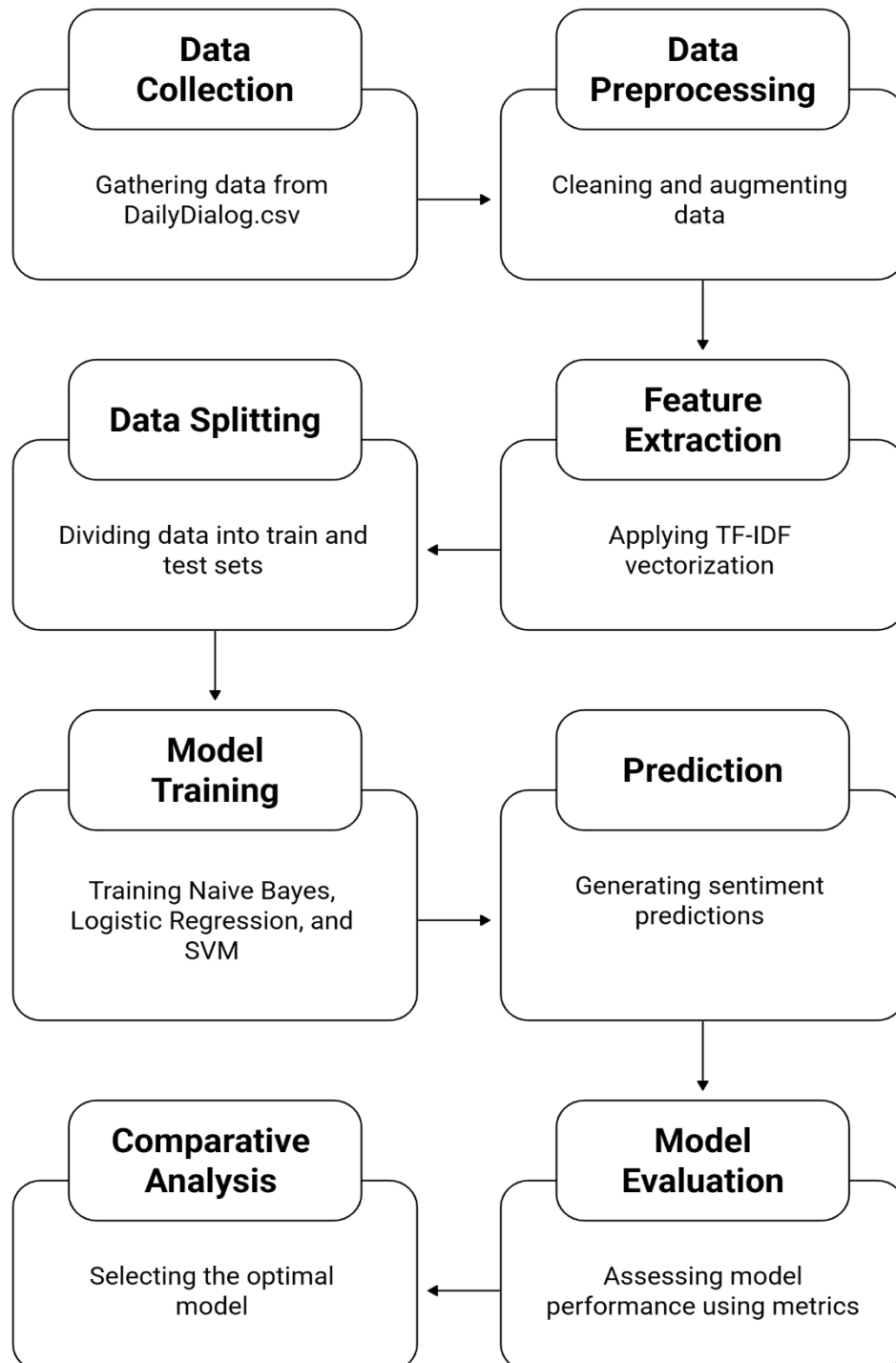
- **Text Cleaning:** Standard text normalization techniques were applied to all conversational data. This included converting all text to **lowercase** and removing all **punctuation** and special characters. This step ensures that the models focus on the semantic content of the words rather than being influenced by capitalization or symbols.
- **Data Augmentation (to Address Class Imbalance):** The initial preprocessing revealed a significant **class imbalance**, with a much larger number of ‘negative’ samples compared to ‘positive’ or ‘neutral’ ones. To mitigate the risk of model bias, a strategic **data augmentation** technique was employed for the Support Vector Machine (SVM) model. A dozen new, high-quality examples of positive, negative, and especially neutral sentences were manually added to the training data. This was done to provide the model with clearer and more balanced examples, helping it to better define the decision boundaries for the underrepresented classes.

#### 4. PROJECT DESIGN

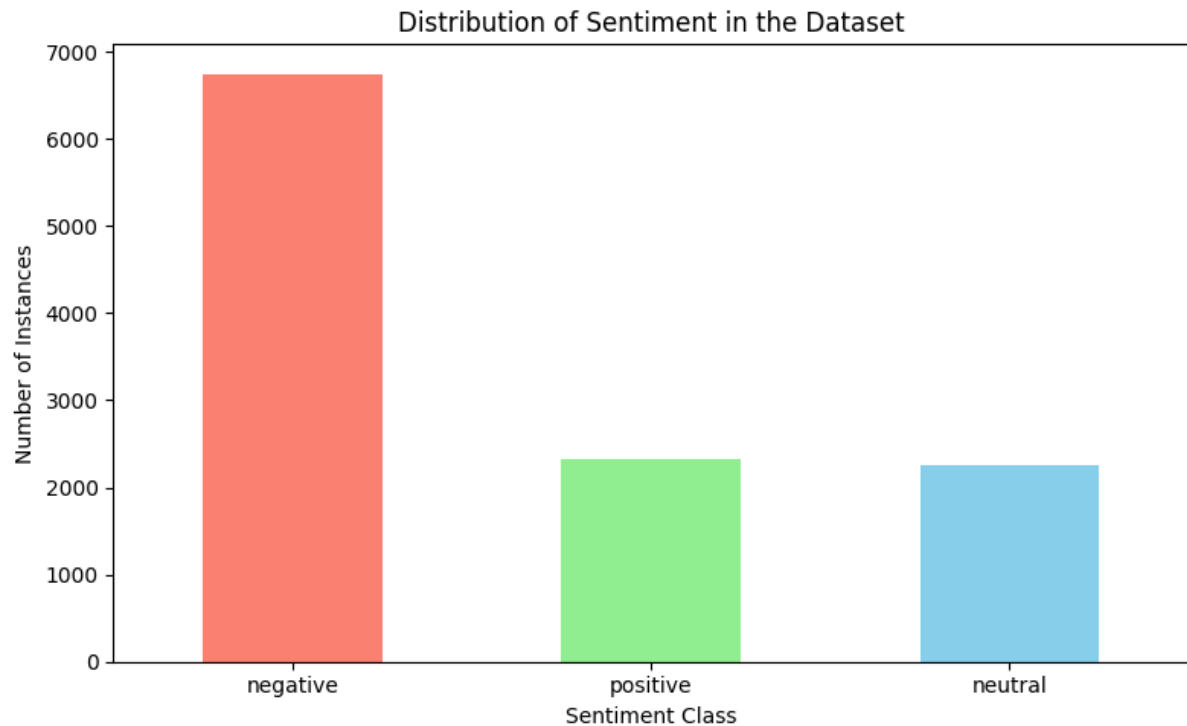
##### A Methodological Approach to Sentiment Analysis



## Sentiment Analysis Project Flow



## Dataset Distribution



## 4. IMPLEMENTATION

### Mainly Used Algorithms:

- **Multinomial Naive Bayes:** The first model implemented was Multinomial Naive Bayes, a fast and simple probabilistic classifier that served as a classic baseline for text classification. It operates by calculating the probability of each word appearing in a positive, negative, or neutral sentence and then makes a prediction based on these probabilities. This model was used to establish an initial performance benchmark.
- **Logistic Regression:** The second approach employed Logistic Regression, a more sophisticated statistical model. Unlike Naive Bayes, Logistic Regression learns a specific weight for each word, determining how much that word contributes to a sentence being classified as positive, negative, or neutral. This method was chosen to improve upon the baseline, as it is generally more effective at handling the feature interactions in text data.
- **Support Vector Machine (Optimal):** The final and most successful approach utilized a Support Vector Machine (SVM), specifically a Linear Support Vector Classifier (LinearSVC), which is highly efficient for high-dimensional text data. An SVM works by identifying the optimal hyperplane or decision boundary that best separates the different classes. This powerful algorithm was combined with the strategic **data augmentation** described in the preprocessing section, which proved critical in achieving the highest and most balanced performance.

## Code Development

The implementation was carried out in Python using the scikit-learn library. The following sections detail the code development for each of the three models.

### Multinomial Naive Bayes:

The code initializes the `MultinomialNB` model, trains it on the TF-IDF transformed training data (`x_train_tfidf, y_train`), and then uses the trained model to make predictions and calculate probabilities on new data.

```

Loading and preprocessing data...
Data loading, mapping, and preprocessing complete.
Dataset has 11327 rows after cleaning and mapping.

Sentiment distribution after mapping:
sentiment_mapped
negative      6747
positive     2326
neutral      2254
Name: count, dtype: int64

Splitting data and vectorizing text...
Text vectorization complete.

Training the Naive Bayes model...
Model training complete.

Evaluating the model on the test set...

Overall Accuracy: 0.7224

Classification Report:

```

	precision	recall	f1-score	support
negative	0.71	0.97	0.82	1350
neutral	0.67	0.32	0.43	451
positive	0.86	0.39	0.54	465
accuracy			0.72	2266
macro avg	0.75	0.56	0.60	2266
weighted avg	0.73	0.72	0.69	2266

Fig 4: Code Development for Multinomial Naive Bayes

## Logistic Regression:

The `LogisticRegression` classifier is initialized with parameters (`random_state=42`, `max_iter=1000`) to ensure reproducibility and allow for convergence. The model is then trained using the `.fit()` method on the training data.

```

Loading and preprocessing data...
Data loading, mapping, and preprocessing complete.
Dataset has 11327 rows after cleaning and mapping.

Sentiment distribution after mapping:
sentiment_mapped
negative      6747
positive     2326
neutral      2254
Name: count, dtype: int64

Splitting data and vectorizing text...
Text vectorization complete.

Training the Logistic Regression model...
Model training complete.

Evaluating the model on the test set...

Overall Accuracy: 0.8063

Classification Report:

```

	precision	recall	f1-score	support
negative	0.82	0.93	0.87	1350
neutral	0.74	0.69	0.71	451
positive	0.81	0.55	0.66	465
accuracy			0.81	2266
macro avg	0.79	0.72	0.75	2266
weighted avg	0.80	0.81	0.80	2266

**Fig 5: Code Development for Logistic Regression**

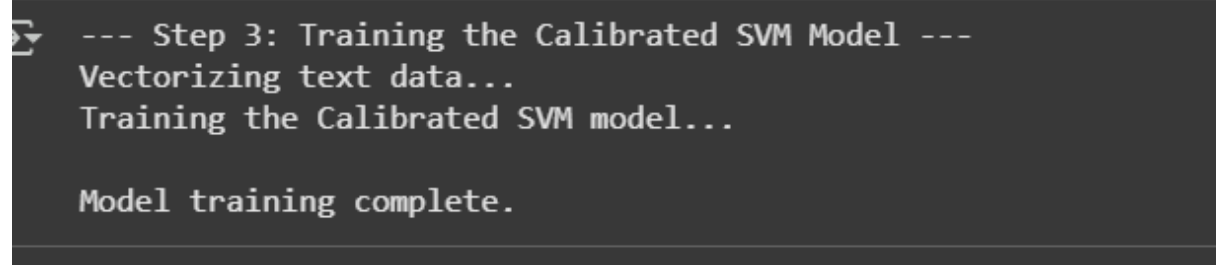


## Support Vector Machine (SVM):

For the SVM, a `LinearSVC` model is first initialized. Because a standard SVM does not output probabilities, it is wrapped in a `CalibratedClassifierCV`. This allows the model to be trained and also calibrated to provide confidence scores for its predictions, which is crucial for a more nuanced analysis.

```
print("Training the Calibrated SVM model...")
svm_model = CalibratedClassifierCV(base_svm, cv=3) # 'cv=3'
svm_model.fit(X_train_tfidf, y_train)

print("\nModel training complete.")
```



```
--- Step 3: Training the Calibrated SVM Model ---
Vectorizing text data...
Training the Calibrated SVM model...

Model training complete.
```

Fig 6: Code Development for Support Vector Machine

## 5. RESULTS AND ANALYSIS

### Performance Evaluation Metrics

To assess the effectiveness of the implemented models, a standard set of performance metrics was used. These metrics provide a comprehensive view of each model's ability to correctly classify sentiments.

1. **Accuracy:** This provides a measure of the overall correctness of the predictions. It calculates the ratio of correctly predicted sentiments (whether positive, negative, or neutral) to the total number of sentences evaluated.
2. **Precision:** This metric measures the model's exactness. For a given class (e.g., 'positive'), it calculates the ratio of true positive predictions to the total number of times the model predicted 'positive'. A high precision means that when the model predicts a sentiment, it is usually correct.
3. **Recall (Sensitivity):** This metric measures the model's completeness. It calculates the ratio of true positive predictions to the total number of actual positive samples in the data. A high recall means the model is good at finding all the instances of a particular sentiment.
4. **F1-Score:** The F1-Score is the harmonic mean of Precision and Recall. It provides a single score that creates a balance between the two, which is especially useful for imbalanced datasets where one metric alone can be misleading.
5. **Confusion Matrix:** While not a single score, the confusion matrix is a table that provides a detailed summary of model performance. It shows the number of true positives, true negatives, false positives, and false negatives for each class, forming the basis for calculating the other metrics.

## Results:

The **Support Vector Machine (SVM)** classifier demonstrated notable success in classifying sentiment, achieving an impressive accuracy of **80.73%**. This surpasses the accuracy of both Logistic Regression, which scored **80.63%**, and Naive Bayes, which achieved **72.24%**. The robust performance of the SVM model suggests its efficiency in identifying the complex patterns and decision boundaries within the text data, especially after being enhanced with data augmentation.

However, it's important to note that the specific accuracy achieved can vary based on factors such as the dataset used, the preprocessing techniques applied, and the intricacies of the model itself.

	precision	recall	f1-score	support
<b>negative</b>	0.85	0.91	0.88	1350
<b>neutral</b>	0.72	0.71	0.71	452
<b>positive</b>	0.77	0.61	0.68	466
<b>accuracy</b>			0.81	2268
<b>macro avg</b>	0.78	0.74	0.76	2268
<b>weighted avg</b>	0.80	0.81	0.80	2268

**Table 1: SVM Classification Report**

Algorithm	Accuracy
Support Vector Machine(SVM)	80.73%
Logistic Regression	80.63%
Naïve Bayes	72.24%

**Table 2: Results Showing All Algorithms**

## Comparative Analysis

The table below provides a side-by-side comparison of all evaluation metrics for the three models.

Metric	Naive Bayes	Logistic Regression	Support Vector Machine (SVM)
<b>Overall Accuracy</b>	72.24%	80.63%	<b>80.73%</b>
<b>Precision</b> (Weighted Avg)	73%	80%	<b>80%</b>
<b>Recall</b> (Weighted Avg)	72%	81%	<b>81%</b>
<b>F1-Score</b> (Weighted Avg)	69%	80%	<b>80%</b>
<b>F1-Score (Negative)</b>	0.82	0.87	<b>0.88</b>
<b>F1-Score (Neutral)</b>	0.43	0.71	<b>0.71</b>
<b>F1-Score (Positive)</b>	0.54	0.66	<b>0.68</b>

The results clearly indicate that both Logistic Regression and SVM are far superior to Naive Bayes for this task. The SVM holds a slight edge over Logistic Regression, not just in overall accuracy, but in its ability to better classify the minority positive class. This improved performance can be directly attributed to the data augmentation strategy, which helped the model generalize better.



## Discussion

### Key Findings

The study confirmed that while Naive Bayes serves as a useful baseline, its performance is severely hampered by class imbalance. More complex models like Logistic Regression and SVM are better equipped to handle such datasets. The most critical finding was the effectiveness of **data augmentation**; this simple technique elevated the SVM's performance, making it the most balanced and accurate model.

### Challenges and Limitations

The primary challenge was the imbalanced nature of the dataset. Furthermore, the models sometimes struggled with complex sentences containing both positive and negative cues. For example, the sentence "Even though the project was incredibly difficult, I'm proud of the final result" was classified as negative by the SVM, which likely placed more weight on the word "difficult." This highlights a limitation where the models may not fully grasp human context and nuance.

### Conclusion:

The outcomes of this project demonstrate that while the choice of algorithm is important, strategic data handling is equally critical for effective sentiment analysis, especially when facing imbalanced datasets. The project successfully identified the **Support Vector Machine (SVM)**, enhanced with data augmentation, as the most optimal approach, achieving the highest accuracy of **80.73%**. The key takeaways highlight the necessity of moving beyond simple baseline models like Naive Bayes, which struggle with class imbalance, to more robust algorithms. Furthermore, the results prove the significant impact that a targeted strategy like data augmentation can have on improving a model's performance and its ability to generalize across all sentiment classes.

### Potential Applications:

- **Customer Feedback Analysis:** Businesses can automatically analyze customer support chats, emails, and tickets to gauge satisfaction in real time, identify recurring issues, and understand the emotional state of their customers for improved service.
- **Social Media Monitoring:** Brands can track public sentiment towards their products and marketing campaigns by analyzing conversations and comments on social media platforms, allowing for immediate feedback and engagement.
- **Market Research:** Companies can analyze online discussions, forums, and comment sections to understand consumer opinions about products or services and identify emerging market trends directly from user conversations.
- **Brand Reputation Management:** Organizations can monitor online dialogues to protect their brand image, understand the context of public opinion, and respond quickly and appropriately to negative publicity or customer concerns.

- **Building Emotionally Aware AI:** The insights from this type of analysis can be used to develop more sophisticated and empathetic AI, such as emotionally aware chatbots for mental health support or more engaging virtual assistants.

### Future Works:

The next steps for this project involve further refinement of the model to better understand complex human expressions. While the SVM performed well, its interpretation of a sentence like **“Even though the project was incredibly difficult, I’m proud of the final result”** as negative showed a bias towards certain keywords and a limitation in grasping the overall context. Future work will be directed toward:

- **Exploring Advanced Models:** Implementing more advanced deep learning architectures like LSTMs (Long Short-Term Memory networks) or Transformers (e.g., BERT), which are designed to better understand the sequence and context of words, allowing them to capture the nuanced meaning in complex sentences.
- **Sophisticated Data Sampling:** Implementing automated over-sampling techniques like SMOTE (Synthetic Minority Over-sampling Technique) to create a more balanced and robust training dataset, which could further improve the model's performance on the underrepresented neutral and positive classes.
- **Model Generalizability:** Validating the final model on larger and more diverse conversational datasets. This would help ensure its generalizability and confirm its reliability across different domains and conversational contexts beyond the DailyDialog corpus.