Group12

SENTIMENT ANALYSIS PROJECT

Text Sentiment Analysis with Machine Learning

Ezgi SUBAŞI 20050111016
Tayyibe PEHLIVANLI 20050111018
Şevval ÇOLAK 20050111082

2024

Problem Definition

Analyzing the emotions expressed in user-generated content on social media is a significant problem. Sentiment analysis is a machine learning task that aims to understand and classify emotions expressed by individuals through written text.

This analysis is widely used in areas such as social media, customer reviews, and user feedback. Identifying the emotional tone of texts plays a crucial role in enhancing customer satisfaction, monitoring mental health, and understanding user behavior.

The goal of this project is to develop a machine learning model that classifies given texts as positive or negative focusing on social media-based sentiment analysis. Additionally, it aims to improve the accuracy of the model and ensure effective performance on large datasets.

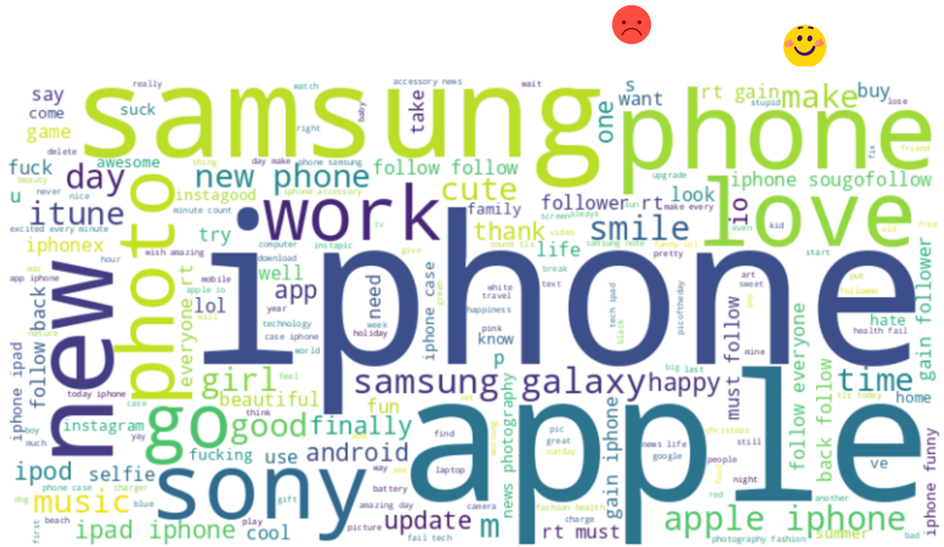
Data Source

The dataset used in this project consists of texts from social media platforms, labeled as 0 for positive sentiment, 1 for negative sentiment.

Data features:

- **Texts:** Content expressing users' sentiments.
- **Labels:** 0, 1.
- **Size:** Examples for each class, highlighting class imbalance.

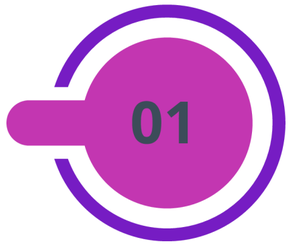
This dataset serves as a foundation for model development and evaluation.



Data Preprocessing

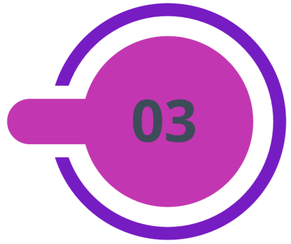
Data preprocessing is essential for improving model accuracy.
The steps include:

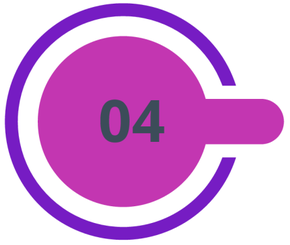
Cleaning Missing Values: Remove missing values if exist.(not exist in our dataset)



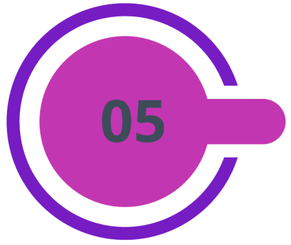
Removing Special Characters: Irrelevant special characters (punctuation, numbers, etc.) are removed.

Lowercasing: All letters in the text are converted to lowercase.





Removing Stopwords: Words with no meaning (e.g., "and", "the") are removed.



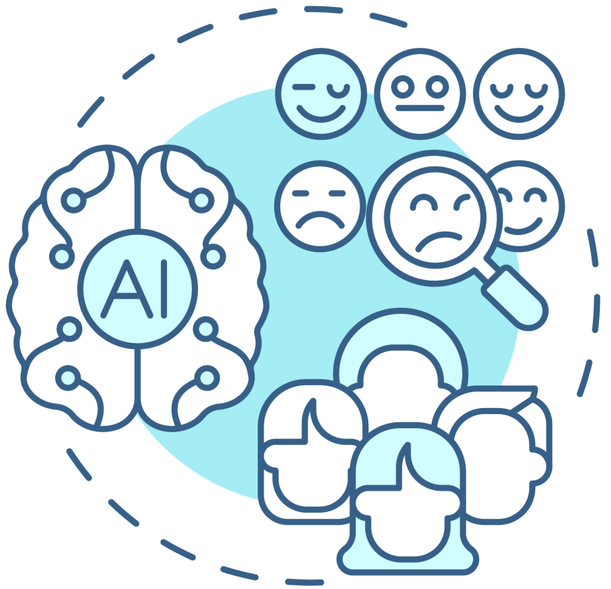
Lemmatization: Words are reduced to their root form by using lemmatization.

Text Processing

Text processing is crucial for sentiment analysis. During this step, unnecessary information is removed, and meaningful data is extracted:

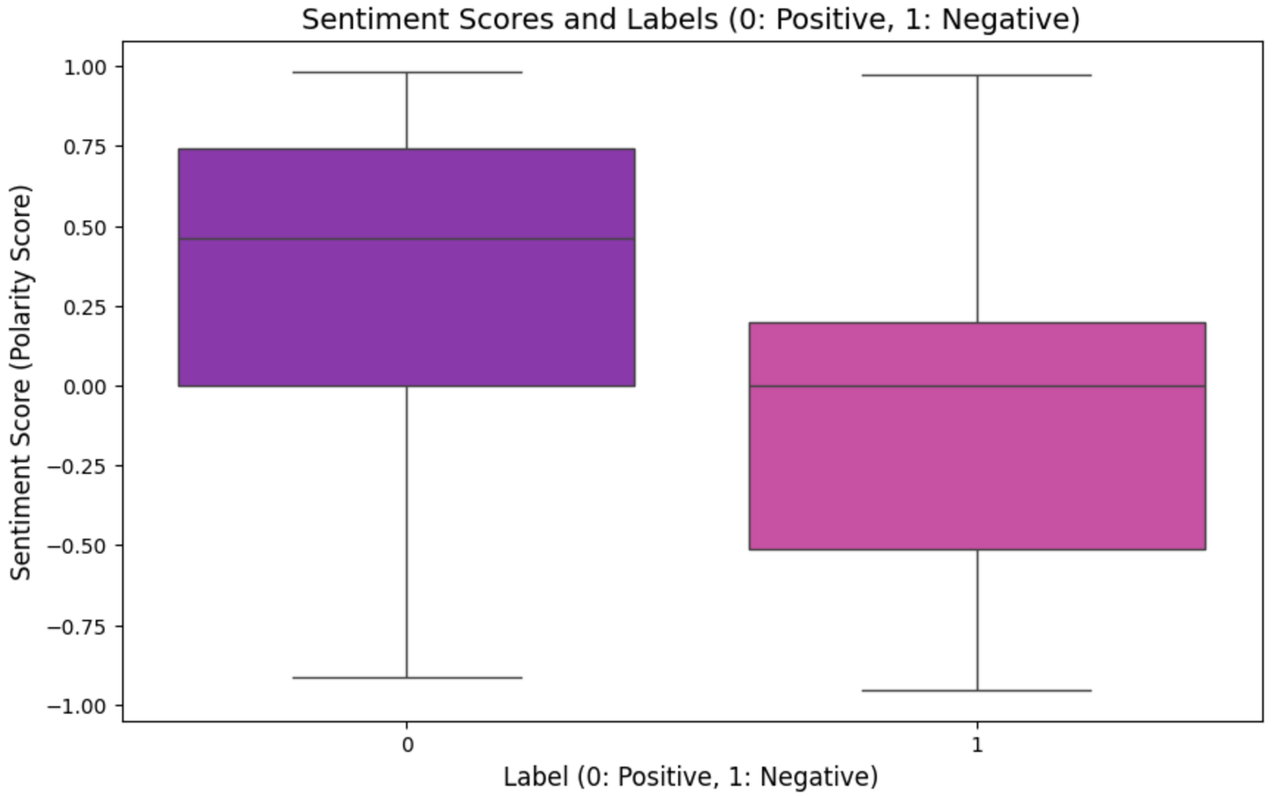
1. **Removing Stopwords:** Common, insignificant words (e.g., "and", "the", "this") are removed.
2. **Lemmatization:** Words are reduced to their corrected to their base forms ("running" → "run").
3. **Converting to Numerical Data:** Texts are converted into numerical data using methods like TF-IDF or word2vec.

These steps preserve the meaning of the text and improve model accuracy.



Evaluating Sentiment Labels: NLTK Polarity Scores

- NLTK is a Python library for processing textual data.
- NLTK calculates the polarity score for each tweet, which ranges from -1 (negative) to 1 (positive). This score helps determine the overall sentiment of the text.
- The goal of calculating the polarity scores is to evaluate whether the sentiment labels (negative and positive) in our dataset are consistent.



a box plot visualizing the distribution of sentiment scores

Vocabulary and Vectorization

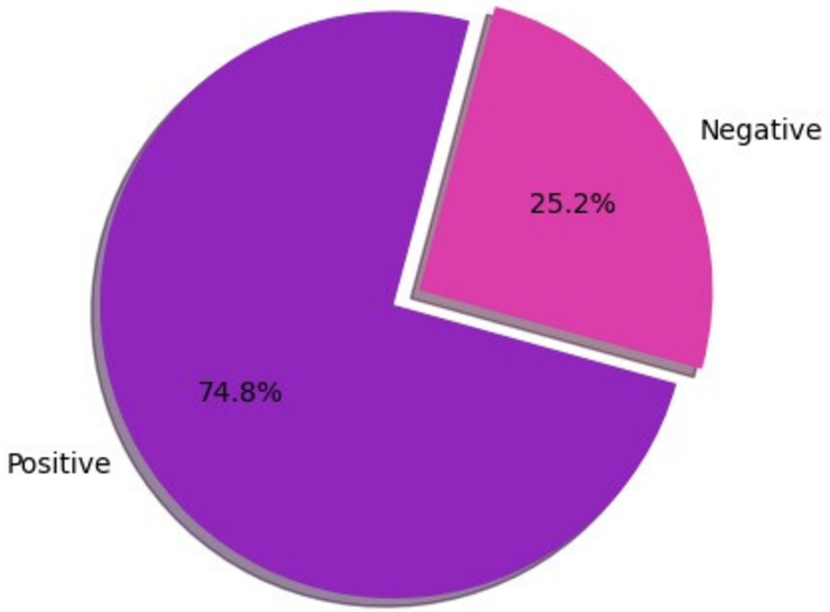
- Models cannot work directly with text data because computers cannot understand text. Therefore, text must be converted into numerical data.

- **TF-IDF (Term Frequency-Inverse Document Frequency)** helps with this conversion. After applying TF-IDF, each tweet is transformed into a numerical vector, representing the importance of each word in a sparse matrix format. This enables the model to highlight important words and make more accurate predictions.

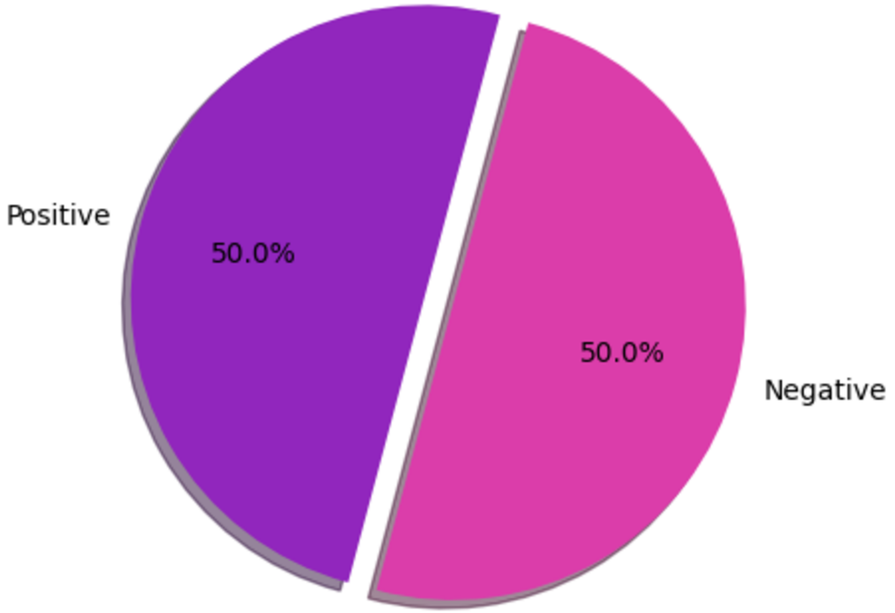
- In addition to TF-IDF, we also experimented with **Word2Vec**. Word2Vec captures the semantic meaning and relationships between words by converting them into vectors in a continuous space. This helps the model understand context better, leading to more meaningful and accurate predictions. But, our dataset is limited, Word2Vec give better performance in large and complicated dataset.

- Using both TF-IDF and Word2Vec leverages their strengths to improve the robustness and accuracy of our sentiment analysis.

Handling Imbalanced Dataset



BEFORE APPLICATION OF SMOTE



AFTER APPLICATION OF SMOTE

The dataset had an imbalance, with fewer samples in the negative class. To address this, SMOTE (Synthetic Minority Over-sampling Technique) was applied to generate synthetic samples for the minority class, helping the model learn all classes equally and improving accuracy.

Algorithms Used

We applied five different machine learning algorithms in our sentiment analysis to improve accuracy and reliability. This approach allows us to validate the consistency of sentiment labels in our dataset and select the most effective model.

Logistic Regression: A simple and effective method suitable for our binary classification problem; it provides fast and reliable results with a linear model for sentiment analysis

Random Forest: Strong in learning complex relationships in text data; it uses multiple decision trees to reduce overfitting and improve accuracy.

Decision Tree: Directly creates rules from our data; its easily interpretable model makes it suitable for sentiment analysis.

Support Vector Machine (SVM): Maximizes class separation and works well with high-dimensional text data, enhancing the accuracy of our project.

Naive Bayes: Offers fast and effective classification; performs well in sentiment analysis even with small datasets.

Model Training

Model training focuses on using data effectively to make accurate predictions. In this project, the dataset was divided into training and validation sets:

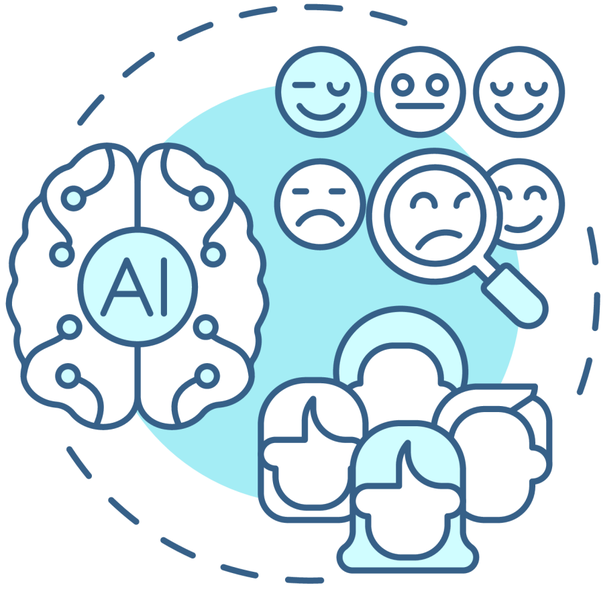
- Training Set: **80%**
- Validation Set: **20%**

The training set was used for model learning, while the validation set was used to evaluate the model's accuracy. The model was trained with default parameters, and the performance of different algorithms was compared.

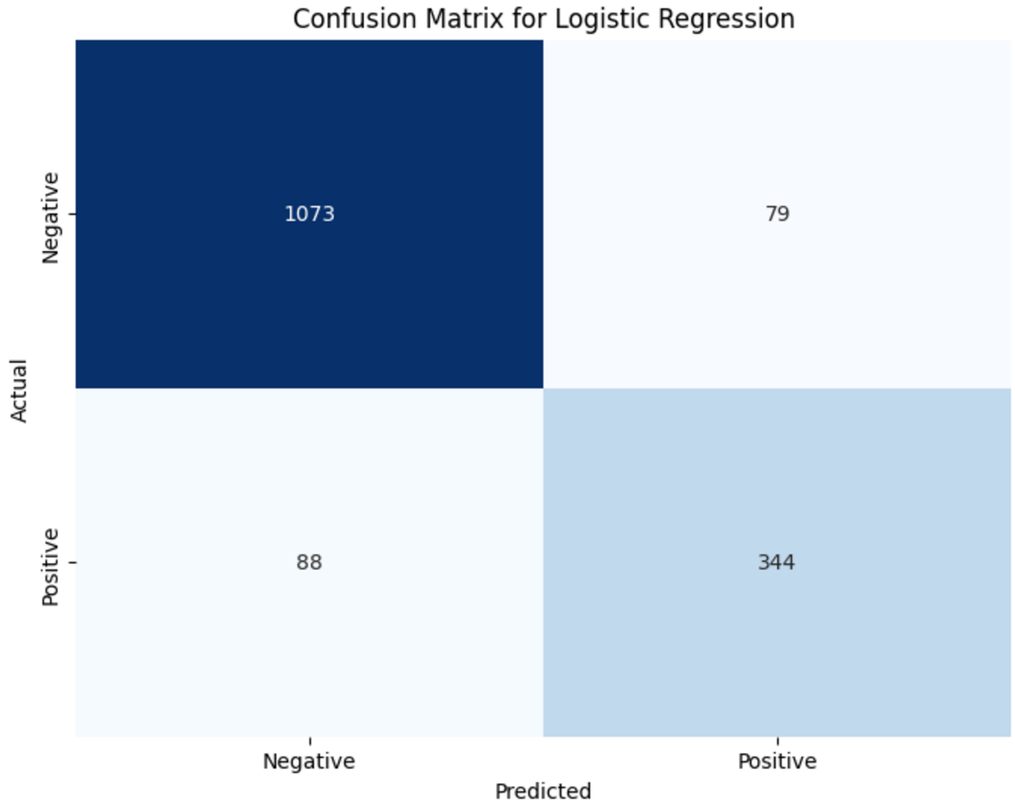
Model Performance

The model's performance was evaluated using four key metrics:

1. **Accuracy:** The ratio of correct predictions, though misleading with imbalanced datasets.
2. **Precision:** The proportion of correct positive predictions. High precision means low false positives.
3. **Recall:** The proportion of actual positives correctly predicted. High recall means low false negatives.
4. **F1 Score:** A balance between precision and recall, important for imbalanced datasets.

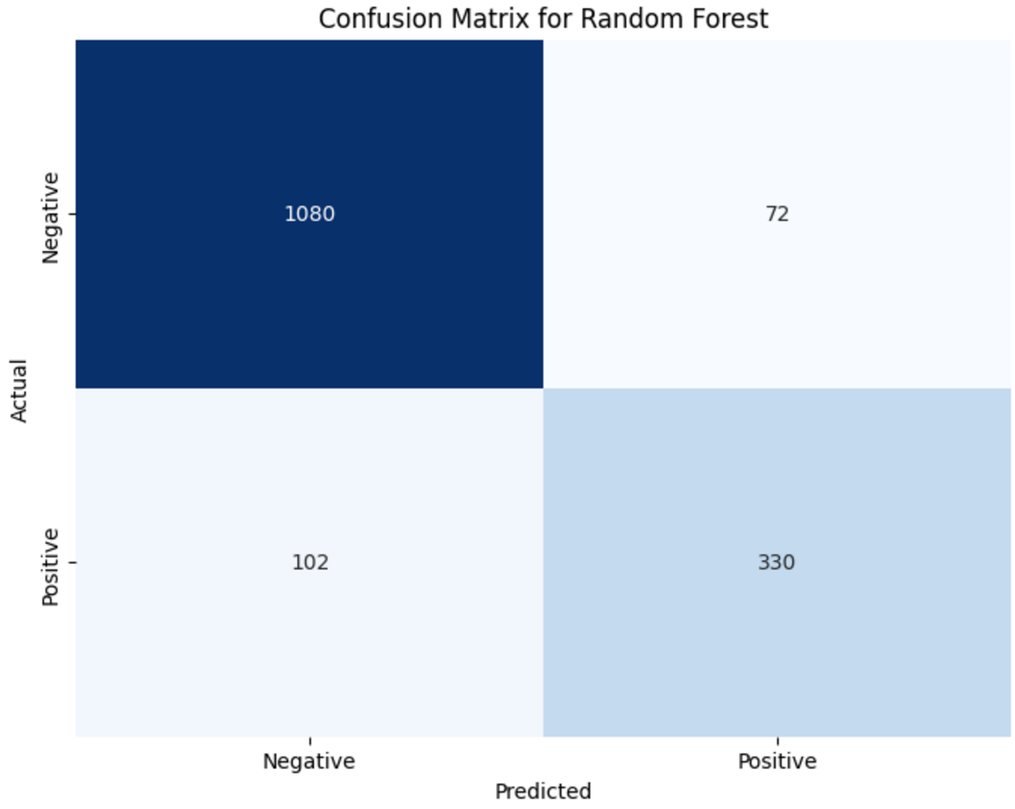


Model Performance



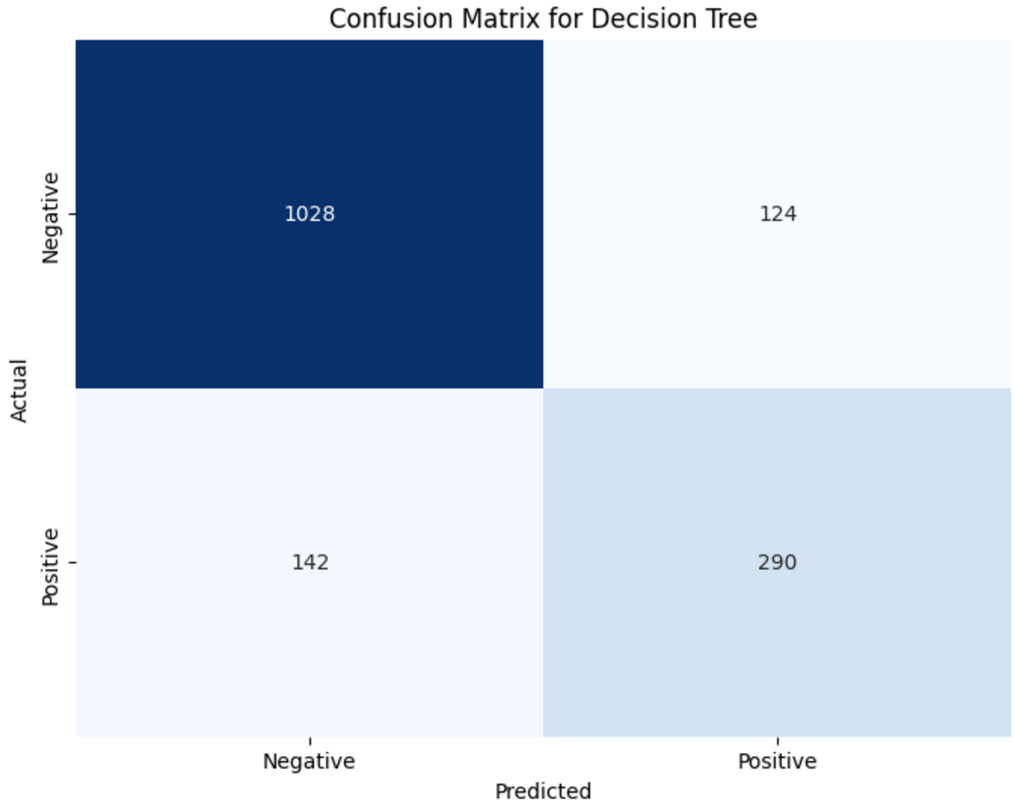
- **True Positives: 344**
- **True Negatives: 1073**
- **False Positives: 79**
- **False Negatives: 88**

Model Performance



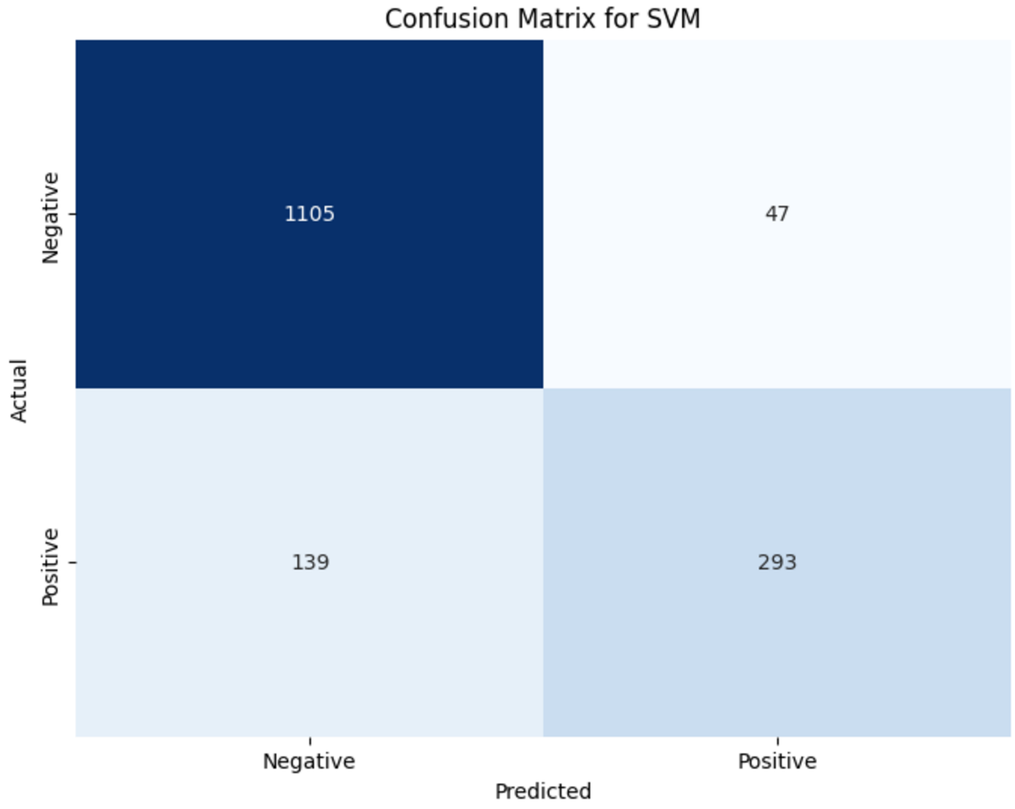
- **True Positives: 330**
- **True Negatives: 1080**
- **False Positives: 72**
- **False Negatives: 102**

Model Performance



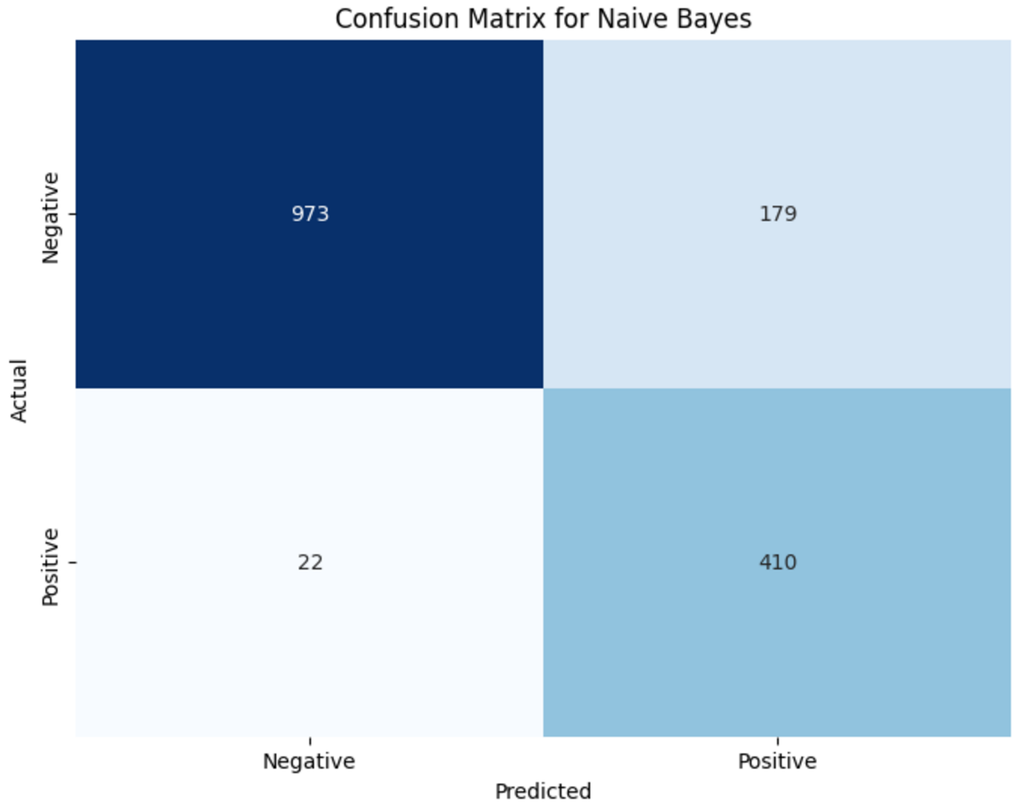
- **True Positives: 290**
- **True Negatives: 1028**
- **False Positives: 124**
- **False Negatives: 142**

Model Performance



- **True Positives: 293**
- **True Negatives: 1105**
- **False Positives: 47**
- **False Negatives: 139**

Model Performance



- **True Positives: 410**
- **True Negatives: 973**
- **False Positives: 179**
- **False Negatives: 22**

Model Performance

TF-IDF VECTORIZER

Model	Accuracy	Precision	Recall	F1 Score
SVM	0.88	0.86	0.67	0.75
Logistic Regression	0.89	0.81	0.79	0.80
Decision Tree	0.83	0.70	0.67	0.68
Naive Bayes	0.87	0.69	0.94	0.79
Random Forest	0.89	0.82	0.76	0.79

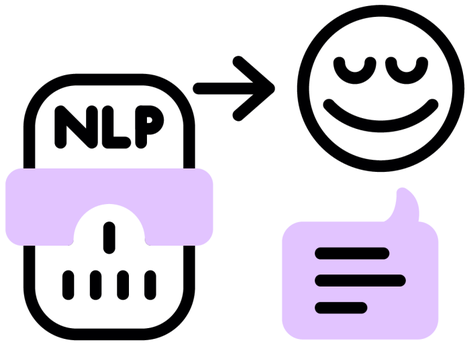
Model Performance

Word2Vec

Model	Accuracy	Precision	Recall	F1 Score
SVM	0.83	0.80	0.85	0.81
Logistic Regression	0.84	0.81	0.85	0.82
Decision Tree	0.79	0.74	0.76	0.75
Naive Bayes	0.69	0.69	0.74	0.68
Random Forest	0.85	0.82	0.84	0.83

Result Analysis

In TF-IDF vectorization Random Forest and Logistic Regression are showed best accuracies and performances compared to other algorithms, but Random Forest is best when using Word2Vec vectorization. In generally, TF-IDF vectorzation is better than Word2Vec in this task. When we use lemmatization, we obtain better solution according to stemming.



Challenges Experienced

One of the major challenges faced during the project was dealing with the imbalanced dataset, where some classes had more samples than others, making it harder for the model to predict accurately. Additionally, using dataset is limited. This made it difficult for the model to make good predictions.

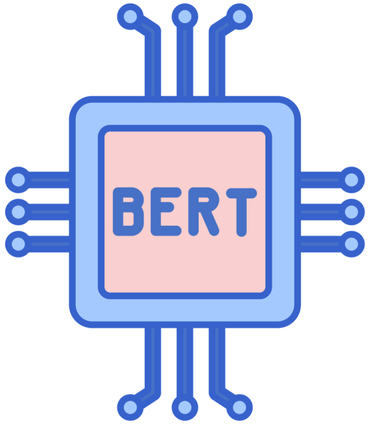
Improvement Suggestions

To improve accuracy, a larger and more balanced dataset should be used, addressing class imbalance and enabling the model to learn each class equally.

Also, to optimize model, we can do hyperparameters tuning.

CNN and RNN-based models such as LSTM and GRU can be used to analyze the context, syntax and word order of the language.

Additionally, we can use pre-trained model like BERT or GPT can analyze text more meaningfully and provide higher accuracy. We can fine-tuning BERT-based models for sentiment analysis.



THANKS FOR LISTENING!