Sentiment Analysis in Kannada using NLP

Dhruthi GH, 21MID0012, Anoushka Ghosh, 21MID0054, Samay Raj Adhikari, 21MID0181, Shravan S, 20MID0064

Abstract —This project presents a sentiment analysis system specifically designed for the Kannada language, aiming to bridge the gap in emotional understanding for regional languages through Natural Language Processing (NLP) and machine learning. While sentiment analysis has advanced significantly for global languages like English, Kannada lacks robust tools for emotion detection. Our system classifies key emotions—joy, anger, sadness, and fear—using both text and speech inputs. A curated dataset of Kannada- labeled sentences forms the foundation for training. For voice inputs, we integrate the Google Cloud Speech-to-Text API, enabling users to record custom audio clips which are transcribed and analyzed for sentiment. To achieve this sentiment classification, we employ machine learning models including Multinomial Naive Bayes, Random Forest, and One-vs-Rest classifiers, further optimized using ensemble learning with a Hard Voting Classifier. Textual data is rigorously preprocessed through normalization, tokenization, stopword removal, and vectorization to ensure consistency and accuracy. Testing of Eight different Voting Classifier combinations helped achieved the best performance, reaching up to 87% accuracy. The implementation effectively demonstrates the technical feasibility of emotion recognition in Kannada and promotes digital inclusivity by enabling native language interaction with intelligent systems. It holds strong potential in domains like healthcare, education, and governance, where capturing public sentiment can enhance service delivery and informed decision-making.

Index Terms— Emotion Recognition, Ensemble Learning, Indian Languages NLP, Kannada Language, Low-Resource Language Processing, Machine Learning, Natural Language Processing (NLP), Speech Emotion Recognition, Supervised Learning, Text Classification.

I. INTRODUCTION

In recent years, Natural Language Processing (NLP) has become a cornerstone of advancements in artificial intelligence, enabling applications such as sentiment analysis, chatbots, and language translation. However, despite its proliferation in global languages, the scope of sentiment analysis in regional languages like Kannada remains underdeveloped [1] [2]. Kannada, spoken by millions in India, suffers from a lack of large annotated datasets and mature speech-to-text technologies, which significantly hinder the performance of sentiment analysis systems tailored for the language [3].

The absence of adequate resources for Kannada sentiment analysis significantly limits its accessibility and utility for native speakers, leaving a critical gap in the broader landscape of Natural Language Processing (NLP). As one of the most widely spoken regional languages in India, Kannada still lacks the comprehensive datasets and tools necessary for accurate sentiment analysis. This scarcity hinders the development of systems capable of understanding and

processing the emotional nuances present in both written and spoken Kannada, which are essential for applications like customer feedback analysis, opinion mining, and digital assistants [4] [5].

Moreover, the lack of robust speech-to-text technologies tailored for Kannada further compounds the issue, restricting the scope of sentiment analysis to text inputs alone and excluding a vast population that communicates more comfortably through speech [3] [5]. These limitations present significant challenges to building sentiment predictors that are inclusive, accurate, and effective in real-world scenarios.

This project seeks to address these challenges by introducing a sentiment analysis system that leverages advanced machine learning techniques and integrates speech-to-text capabilities. By combining these approaches, the system aims to enable sentiment prediction from both text and voice inputs, providing a more comprehensive and accessible solution [1] [4]. This innovative approach not only bridges existing gaps but also lays the foundation for expanding sentiment analysis capabilities in Kannada, making it a valuable tool for native speakers and a significant contribution to regional language NLP research [2].

The primary objective of this project is to develop a robust sentiment analysis system tailored for the Kannada language using advanced machine learning models. This system aims to process both text and voice inputs, thereby ensuring versatility and inclusivity for users. A key aspect of the project is the integration of speech-to-text capabilities, enabling the efficient conversion of voice inputs into text for sentiment analysis [6]

To achieve high performance and accuracy, multiple machine learning models will be evaluated and optimized, ensuring that the best possible model or combination of models is implemented. The sentiment analysis will be based on a pre-prepared dataset comprising 2,000 labeled sentences, covering four emotion categories: Sadness, Fear, Anger, and Joy [7].

While the project incorporates both textual and vocal inputs, its scope is limited to these predefined emotion categories. It does not address dialectical variations within Kannada or expand beyond the specified dataset. This focus ensures the development of a specialized and efficient system, laying the groundwork for future advancements in Kannada sentiment analysis [8] [9].

This research addresses a critical gap in the field of NLP for regional languages. By providing a robust framework for Kannada sentiment analysis, it contributes to the growing body of knowledge in multilingual NLP and enhances the accessibility of AI-driven solutions for native Kannada speakers. The integration of speech-to-text technology further ensures inclusivity, empowering users who prefer voice interactions over text-based communication [1][10].

II. LITERATURE SURVEY

Eshwarappa et al. (2024) [1] introduces a novel approach for Kannada sentiment analysis by using a dataset obtained from SemEval 2014 Task4 via Google Translate to overcome the scarcity of labeled datasets and feature selection issues. They proposed the Kannada-BERT (K-BERT) model integrated with a probability-clustering approach for topic aspect extraction, leading to the K-BERT-PC classifier, which achieved 91% accuracy, improving on existing models. However, the study's limitation was its minimal exploration of feature selection, which may have impacted the model's ability to capture all relevant features.

Chattu et al. (2024) [2] propose a deep learning-based approach for sentiment analysis in the Telugu language using multi-domain datasets. The methodology utilizes BiLSTM and BiGRU networks to capture contextual information in Telugu feature sequences. The model performs better than traditional machine learning methods on four benchmark datasets, achieving an F1-score of 86% for song datasets. However, limited data curtails generalizability for various Telugu language scenarios.

Divate (2021) [3] presents a polarity-based sentiment analysis system using Long Short-Term Memory (LSTM) networks for Marathi e-news, aiming to improve the quality of online content by filtering news based on sentiment polarity. The LSTM model identified the polarity of Marathi e-news with an accuracy of 72%. However, performance depends on input diversity, and the model could be inconsistent across various topics in news coverage.

S, S., & KV, P et al. (2020) [4] performed sentiment analysis on Malayalam Tweets using machine learning approaches such as Naive Bayes, Support Vector Machine, and Random Forest. They found that the highest accuracy for Random Forest was obtained with Unigram and Sentiwordnet, including negation words at 95.6%. This method showed the best performance, but the paper does not address handling sarcasm or informal language in social media text, which could hinder performance in real-world applications.

Hoque et al. (2024) [5] focused on Bengali sentiment analysis using five transformer-based pre-trained models: mBERT, BanglaBERT, Bangla-Bert-Base, DistilmBERT, and XLM-RoBERTabase. Their Transformer-ensemble model achieved an impressive accuracy of 95.97% and an F1-score of 95.96%, surpassing state-of-the-art methods for Bengali sentiment analysis. However, the study struggled with Bengali-English code-mixed data and the low percentage of negative class data.

Li (2021) [6] developed a deep learning model for German sentiment classification employing a self-attentive mechanism considering word-level features like German morphology, slang, and emotion scores. The model, combining CNN and LSTM with the selfattention mechanism, classifies German social media texts. However, with a small corpus of German tweets, its performance is poor, particularly when using punctuation and morphology-based features.

Hande et al. (2021) [7] proposed a multi-task learning framework for sentiment analysis and offensive language identification, targeting code-mixed YouTube comments in Tamil, Malayalam, and Kannada. The approach outperformed single-task models, reducing computational time and space. However, the need for strong computational resources affects its practical operational relevance in low-resource conditions.

Lopes et al. (2022) [8] used BERTimbau for Aspect-Based Sentiment Analysis in Portuguese, achieving a BACC of 77% in reviews of accommodations. However, post-training beyond 10k steps led to unstable outcomes, likely due to the small post-training dataset size. This demonstrates BERT's potential for Portuguese ABSA but highlights the risk of overfitting with small datasets.

Kakde and Padalikar (2022) [9] conducted sentiment analysis on Marathi text using Multinomial Naïve Bayes (MNB) and fine-tuned Bidirectional Encoder Representations from Transformers (BERT). BERT achieved 68.90% accuracy, outperforming MNB and showing good performance even with large text inputs. However, the study did not account for emoticons or emojis, which could enhance sentiment understanding, leaving space for future exploration.

Lal Khan et al. (2025) [10] evaluate different machine learning classifiers (RF, NB, SVM, AdaBoost, MLP, LR) and deep learning models (1D-CNN, LSTM) for Urdu text sentiment analysis. Two text representations are used: word n-grams and fastText pre-trained word embeddings. Combining word n-gram features with Logistic Regression (LR) achieves the best F1 score of 82.05%, outperforming other models. SVM also performed strongly, ranking second in the overall results. However, the study only considered positive and negative sentiment classes, excluding a neutral class. Future work will include the integration of a neutral sentiment class and advanced classifiers like BERT. Despite the promising results, the limited sentiment classes and lack of BERT leave room for future improvement.

Dutta et al. (2021) [11] focuses on sentiment analysis of Dravidian code-mixed Kannada language comments. The dataset used in this study was scraped from YouTube comments, where each instance is labelled with one of the sentiment polarities: "positive, negative, neutral, mixed feelings or not in the intended languages". The authors experimented with various machine learning and deep learning techniques, including Random Forest, Support Vector Machine, Logistic Regression, a hybrid CNN-BiLSTM model, and two BERT-based models. The BERT-based models achieved the best performance, with a weighted F1-score of 0.66 on the validation dataset and 0.619 on the test dataset. The limitation of this study is that the dataset used in this study may not be representative of the entire Dravidian codemixed language spectrum, and the performance of the models may vary on different datasets.

Kalaivani et al. (2021) [12] presents the approaches to the Dravidian-CodeMix-FIRE2021 shared task on sentiment analysis for Dravidian languages (Tamil, Malayalam, Kannada) in code-mixed text. The dataset consists of social media posts from YouTube forums, with the training data sizes of 39,336 for Tamil, 16,970 for Malayalam, and 6,578 for Kannada. The posts are categorized

into positive, negative, unknown state, mixed feelings, or not in the intended language. The authors used the pre-trained Multilingual BERT (MBERT) model and fine-tuned it using the ktrain library. In the final test results, the authors achieved weightedaverage F1-scores of 0.603, 0.698, and 0.595 for the Tamil, Malayalam, and Kannada codemixed languages, respectively. Some drawbacks of this article include the lack of detailed analysis on the impact of the data preprocessing and transliteration/translation techniques, as well as the absence of comparisons with other state-of the-art models or approaches.

Shetty et al. (2022) [13] focuses on performing sentiment analysis of Twitter posts in three different languages, English, Kannada, and Hindi. The dataset used for the English language is the sentiment 140 dataset, which contains about 1.6M samples of posts that are already classified with their sentiment. For Hindi, the authors collected the dataset by crawling posts written in Hindi, which amounted to about 2500 samples. For Kannada, the dataset contained articles in Kannada as well as text translated from English and Hindi that were already classified. A Convolutional Neural Network was then trained and evaluated on each of these datasets and the model achieved high accuracy for all 3 models. The high computation cost of the model as well lack of multinomial classification present in the datasets were some of the limitations of this study.

Rakshitha et al. (2021) [14] proposes a model that addresses these challenges by scraping tweets from Twitter, using text analysis tools to assign sentiment scores, and then classifying the tweets as positive, negative, or neutral. The dataset used in this study consists of tweets collected from Twitter. The methodology involves using Twitter APIs to scrape the tweets, then applying the TextBlob library to assign sentiment scores to the tweets, and finally classifying them as positive, negative, or neutral using a text classification model. The results of the study demonstrate the ability to analyse sentiment in Indian regional languages on social media, which can provide valuable insights into people's opinions and sentiments. However, a potential drawback of this study is that it focuses solely on Twitter data, which may not be representative of the broader social media landscape. Additionally, the accuracy of the sentiment analysis and classification may be limited by the capabilities of the text analysis tools and the text classification model used.

Chandrika et al. (2021) [15] focuses on a rule-based approach to categorise the emotions involved in documents written in Kannada language. The authors have used a labelled dataset of 5000 statements which is a mixture of product reviews, article reviews, and people's emotions. The data is pre-processed by removing the stop words, tagging parts of speech (POS) and creating a bag of words. The rule-based algorithm is then applied on this bag of words. An accuracy of 85% was achieved using this rule-based approach as compared to the 75% achieved in machine learning approaches that were reviewed by the authors. However, a few limitations of this study are the limited dataset, and the limited scope of rules applied to create the algorithm, rules for a grammar are vast and ever-changing.

Abbaschian et al. (2021) [16] reviews several datasets utilized in speech emotion recognition (SER), highlighting that the EMO-DB dataset was predominantly used in older studies,

while the IEMOCAP dataset has gained popularity in more recent research due to its larger sample pool, which is better suited for training deep learning architectures. The authors conducted a thorough review of existing literature, summarizing the techniques and databases used, and identifying challenges that need to be addressed in future research. The findings indicate that deep learning methods, particularly convolutional neural networks, have shown superior performance in emotion recognition tasks due to their ability to capture low-level and short-term discriminative features. Despite its comprehensive review, the paper has some limitations, pointing out the scarcity of well-designed datasets for deep learning tasks, which restricts the training of deep architectures. Additionally, while it covers various methods, it lacks an in-depth analysis of more modern techniques, which could enhance the understanding of current trends in SER.

Shah et al. (2022) [17] proposes a machine learning based sentiment analysis approach using a dataset created by extracting movie reviews in Gujarati language. The dataset included about 500 reviews and the authors labelled the reviews as having positive or negative sentiment. After pre-processing the data by removing stop words and tokenizing the reviews, features were extracted using two methods, TF-IDF and Count Vectorizer. The features extracted using both methods were fed into two machine learning models, Multinomial Naïve Bayes and K Nearest Neighbours. The results of this study show that for features extracted using the TF-IDF method, the MNB model performed significantly better than the KNN model, achieving 87% accuracy compared to 81% for the KNN model. When it comes to features extracted using the Count Vectorizer, both models achieved similar results. A limitation of this study is the limited dataset as well as the lack of multiple labels with regards to the sentiment.

Liu et al. (2024) [18] explores the effectiveness of deep learning models such as BERT and LSTM in multilingual sentiment analysis, particularly in handling subtle emotional nuances across different languages. The dataset used in this study involved collecting data in multiple languages from several sources such as social media, news etc. The authors used deep learning models like BERT and LSTM during the training phase. The study shows that BERT performs better than LSTMs due to the complex attention mechanism. A drawback of this study lies in the different ways emotion is expressed in different languages, which caused complexity in training the models.

Muhammad et al. (2023) [19] studies the AfriSenti dataset, which is the largest sentiment analysis benchmark for African languages, comprising over 110,000 tweets in 14 languages. The tweets were annotated by native speakers and utilized in the AfriSenti SemEval shared task, which attracted over 200 participants. The paper also reveals the performances of several large language model that were trained on this dataset. The best performing model were able to achieve an F1 score of 71.2%, while worst performing models score below 60. However, the dataset suffers from limitations such as imbalances due to the reliance on keywords and geographic locations for data collection. Despite efforts to mitigate label bias, the subjective nature of sentiment analysis means that the data can still reflect biases inherent in the annotation process.

Hegde et al. (2022) [20] contributes by releasing the gold-standard trilingual code-mixed Tulu dataset to perform SA and presents the comprehensive results of using traditional ML classification methods to set the benchmark for the dataset. In most of the cases usually code-mixing includes two languages. The corpus is created using YouTube comments on Tulu movies, trailers, songs etc. A variety of baseline classifiers were used to evaluate the created dataset. All models performed moderately on the dataset with SVM and MLP performing slightly better than the rest. Some drawbacks of this study include the complexity and skewness of the dataset. A lot of the examples in the dataset are skewed towards the positive class, while some examples fail to demonstrate any emotion.

Table 1. Comparison of Literature Survey

Ref	Authors & Year	Methodology	Dataset Source & Size	Performance	Limitations
[1]	Eshwarappa et al. (2024)	K-BERT + Probability-Clustering (K-BERT-PC)	Translated SemEval 2014 Task 4	91% accuracy	Minimal feature selection exploration
[2]	Chattu et al. (2024)	BiLSTM, BiGRU (deep learning)	Multi-domain Telugu datasets	86% F1 (songs)	Limited generalizability due to small datasets
[3]	Divate (2021)	LSTM for sentiment polarity	Marathi e-news	72% accuracy	Performance varies across news topics
[4]	S, S., & KV, P (2020)	ML (Naive Bayes, SVM, Random Forest) with Unigrams + SentiWordNet	Malayalam tweets	95.6% accuracy (RF)	Does not handle sarcasm/informal text
[5]	Hoque et al. (2024)	Transformer ensemble (mBERT, BanglaBERT, etc.)	Bengali text dataset	95.97% accuracy, 95.96% F1	Issues with code-mixing, class imbalance
[6]	Li (2021)	CNN + LSTM + Self-Attention	Small German tweet corpus	Poor with morphology/punctuation	Small corpus and weak handling of specific linguistic features
[7]	Hande et al. (2021)	Multi-task learning (SA + Offensive Detection)	YouTube comments	Better than single-task models	Requires high computational resources
[8]	Lopes et al. (2022)	BERTimbau for ABSA	Reviews of accommodations (Portuguese)	77% BACC	Overfitting due to small post-training set
[9]	Kakde & Padalikar (2022)	MNB vs fine-tuned BERT	Marathi text	68.90% accuracy (BERT)	No emoji/emoticon handling
[10]	Lal Khan et al. (2025)	ML (LR, SVM) & DL (CNN, LSTM) with n-gram + fastText	Urdu text	82.05% F1 (LR)	No neutral class; lacks BERT integration
[11]	Dutta et al. (2021)	BERT, CNN-BiLSTM, SVM, RF	YouTube comments (code-mixed)	F1: 0.66 (val), 0.619 (test)	Dataset not representative of full code-mix diversity
[12]	Kalaivani et al. (2021)	Multilingual BERT with ktrain	YouTube posts (Tamil: 39k, Malayalam: 17k, Kannada: 6.5k)	F1: 0.603 (Ta), 0.698 (Ml), 0.595 (Ka)	No detailed pre-processing analysis, no SOTA comparison

[13]	Shetty et al. (2022)	CNN	Twitter data + translations	High accuracy (unspecified)	High computation; lacks multi-label classification
[14]	Rakshitha et al. (2021)	TextBlob + classifier	Tweets scraped via Twitter API	Effective for basic SA	Only Twitter data; basic sentiment tools used
[15]	Chandrika et al. (2021)	Rule-based sentiment classification	5000 labeled statements	85% accuracy	Small dataset, limited rule coverage
[16]	Abbaschian et al. (2021)	Review of CNN-based SER methods	EMO-DB, IEMOCAP	CNN models best for SER	Lacks modern method discussion, dataset scarcity
[17]	Shah et al. (2022)	MNB & KNN with TF-IDF/Count Vectorizer	500 movie reviews	87% (MNB with TF-IDF)	Small dataset, no multi-label sentiment
[18]	Liu et al. (2024)	BERT vs LSTM for emotion nuances	Social media/news data in many languages	BERT better than LSTM	Complex language-specific emotion expressions
[19]	Muhammad et al. (2023)	Multiple LLMs on AfriSenti dataset	AfriSenti (110k tweets)	Best F1: 71.2%	Keyword/location bias, subjective annotations
[20]	Hegde et al. (2022)	ML models (SVM, MLP, etc.)	YouTube comments (Tulu)	Moderate; SVM/MLP best	Positive skew; emotion-less examples

III. METHODOLOGY

3.1 Data Acquisition

This study utilizes a dataset consisting of Kannada language phrases categorized into four sentiment classes: *Joy*, *Anger*, *Sad*, and *Fear*. The dataset comprises a total of 2,000 sentences collected from various online sources, ensuring diversity in language use and contextual representation. The distribution of the dataset across the sentiment categories includes 604 sentences labeled as Joy, 520 sentences labeled as Anger, 520 sentences labeled as Sad, and 356 sentences labeled as Fear.

The dataset was acquired from a GitHub repository and is stored in Microsoft Excel (.XLSX) format, facilitating structured organization and accessibility. Each entry in the dataset consists of a Kannada phrase along with its corresponding sentiment label, allowing for supervised learning-based sentiment analysis. The data collection process involved manually curating and categorizing the phrases to ensure high-quality annotation.

This dataset serves as the foundational resource for training and evaluating sentiment analysis models tailored to the Kannada language. Leveraging this dataset enhances sentiment classification techniques for low-resource languages and contributes to the broader field of natural language processing (NLP).

3.2 Preprocessing of Kannada Sentences for Sentiment Analysis

3.2.1 Tokenization and Stopword Removal

Tokenization

Tokenization splits sentences into individual words (tokens), allowing models to process a unit at a time. The Kannada sentence "ನಾನು ಸಂತೋಷವಂತ" ("I am happy") is tokenized as ["ನಾನು", "ಸಂತೋಷವಂತ"]. Because structured input is needed by machine learning models, tokenization makes the model able to examine words one at a time, enhancing feature extraction.

Stop-word Removal

Stop words (e.g., "the", "is", "and" in English or "ಮತ್ತು", "ಇದು", "ಅಥವಾ" in Kannada) are typically discarded to eliminate noise and computational overhead. Stop word removal enables the model to concentrate on significant words that have an impact on sentiment classification.

3.2.2 Handling Complex Kannada Grammar Structures and Suffixes

The intricate syntax of Kannada texts, especially the usage of suffixes that change word meanings, is one of the primary processing difficulties. This is addressed by classifying suffixes into three categories using a rule-based stemming approach: those that must be kept, those that require modification before being deleted, and those that can be completely eliminated to reveal the root word. By lowering data sparsity and enhancing feature extraction, our methodical stemming procedure ensures more successful sentiment analysis. Stemming reduces words to their base form so that various word forms are handled as the same. For instance, "ಹಸಿವು" (hunger) and "ಹಸಿವಿನ್ದ" (due to hunger) are stemmed to "ಹಸಿವ". This minimizes feature dimensionality and avoids duplicate representations of the same meaning.

3.2.3 Feature Extraction Using TF-IDF Vectorization

Following preprocessing, Term Frequency-Inverse Document Frequency (TF-IDF) vectorization is used to extract features. By giving terms relevance scores determined by how frequently they occur in individual sentences in relation to the corpus as a whole, this method converts textual data into numerical representations. Rare but important words are given more weight in categorization thanks to TF-IDF, which also makes sure that frequently occurring but uninformative terms are downweighed. The TF-IDF score is computed using the equation 1.

$$TF\text{-}IDF(w,d) = TF(w,d) \times IDF(w) \quad (1)$$

where:

$$TF(w,d) = \frac{\text{Number of times word } w \text{ appears in document } d}{\text{Total number of words in document } d}$$

$$IDF(w) = \log\left(\frac{\text{Total number of documents}}{\text{Number of documents containing word } w}\right)$$

For sentiment classification to be reliable, preprocessing is essential, especially for underrepresented languages like Kannada. The accuracy and effectiveness of sentiment analysis models are increased by eliminating unnecessary information, normalizing language through stemming, and converting textual input into machine-readable vectors. This preprocessing pipeline creates a strong basis for subsequent machine learning and deep learning applications in Kannada text analysis by tackling particular linguistic difficulties.

3.3 Model

To obtain an optimal balance between accuracy, interpretability, and computational efficiency, we investigated a combination of linear, non-linear, probabilistic, and ensemble models. Each model class has different strengths in sentiment classification, so it is important to consider several approaches. Figure 1. visualizes the system architecture of the proposed model.

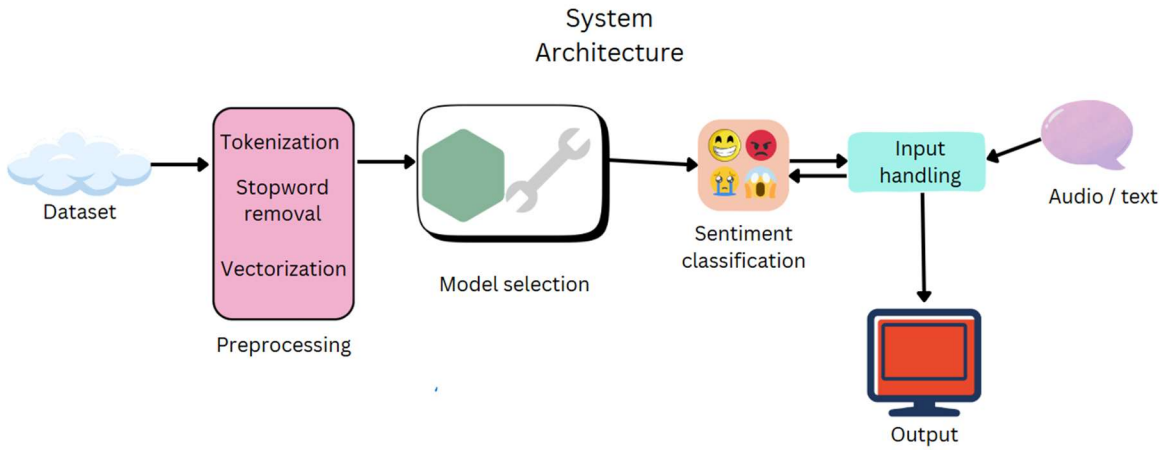

Figure 1. System architecture

3.3.1 Individual Models

(i) Logistic Regression:

Unlike linear regression, which predicts continuous values, logistic regression predicts the probability of a categorical outcome. This is especially common in binary classification, where there are two possible outcomes (e.g., yes/no, true/false, 0/1). The core of logistic regression is the logistic function, also known as the sigmoid function. This function takes any real-valued number and maps it to a value between 0 and 1, representing a probability. This "S"-shaped curve allows the model to estimate the likelihood of an instance belonging to a specific class. Like linear

regression, logistic regression starts with a linear combination of input features (independent variables) and their corresponding weights (coefficients) as seen in equation 2. The linear combination (z) is then passed through the sigmoid function (Equation 3).

$$z = (w_1x_1 + w_2x_2 + \cdots + w_nx_n) + b \quad (2)$$

$$\sigma(z) = \frac{1}{(1+e^{-z})} \quad (3)$$

Where: z – Linear combination, w – Weights, x – Input features, b – bias, σ(z) – Probability output

The output of the sigmoid function, σ(z), represents the probability that the instance belongs to the positive class (e.g., 1). A threshold (often 0.5) is typically used to classify instances. If σ(z) is greater than the threshold, the instance is classified as belonging to the positive class; otherwise, it's classified as belonging to the negative class (e.g., 0).

(ii) Support Vector Machines:

Support Vector Machines (SVMs) are supervised learning models that excel at classification and regression by identifying the optimal hyperplane to separate data classes, maximizing the margin between this hyperplane and the nearest data points, known as support vectors. To handle non-linearly separable data, SVMs employ kernel functions, which map data into higher-dimensional spaces, enabling linear separation. Technically, this involves defining hyperplane equations, computing margin distances, and selecting appropriate kernel types, such as linear, polynomial, radial basis function (RBF), or sigmoid. The training process entails solving a quadratic optimization problem, often incorporating soft margins to account for real-world datasets with imperfect separation. This approach makes SVMs particularly effective in high-dimensional spaces, offering robustness and versatility for complex classification and regression tasks. Some commonly used kernels are:

Linear Kernel - $K(x, x') = x \cdot x'$ (4)

Polynomial Kernel - $K(x, x') = (x \cdot x' + c)^d$ (5)

Radial Basis Function (RBF) Kernel - $K(x, x') = e^{(\gamma(x-x')^2)}$ (6)

(iii) KNN Classifier:

The k-Nearest Neighbors (KNN) classifier is a non-parametric, instance-based learning algorithm that classifies new data points by identifying the "k" closest training examples based on a chosen distance metric, such as Euclidean, Manhattan, or Minkowski, and assigning the majority class among those neighbors; its performance is highly dependent on the selection of "k," which balances overfitting and underfitting, and is sensitive to feature scaling, requiring preprocessing; while simple and versatile for both classification and regression, KNN can be computationally expensive for large datasets due to the need to calculate distances to all training points during prediction.

(iv) Multinomial Naïve Bayes:

Multinomial Naive Bayes is a probabilistic machine learning algorithm primarily used for text classification tasks. It's a variant of the Naive Bayes classifier, which assumes that features are conditionally independent given the class. Specifically, Multinomial Naive Bayes is designed for data that can be represented as counts, such as word frequencies in documents. It calculates the probability of a document belonging to a certain class by multiplying the probabilities of each word occurring in that class, using Laplace smoothing to handle zero-frequency issues. The model is trained by estimating the probability of each word given each class from the training data and then uses Bayes' theorem to predict the class of new documents. Despite its "naive" assumption of feature independence, it often performs surprisingly well in practice, especially for high-dimensional text data, and is known for its simplicity and computational efficiency, making it a popular choice for tasks like spam detection and sentiment analysis.

(v) Random Forest Classifier:

The Random Forest classifier is an ensemble learning method that operates by constructing a multitude of decision trees during training and outputting the mode of the classes (classification) or mean prediction (regression) of the individual trees. Each tree is built from a random subset of the training data (bootstrapping) and, at each node, considers only a random subset of features for splitting, reducing variance and preventing overfitting. This "randomness" makes the model robust and less prone to memorizing the training data. The final prediction is determined by a majority vote across all trees, leading to improved accuracy and generalization compared to single decision trees. Random Forests are known for their ability to handle high-dimensional data, their relative insensitivity to outliers, and their provision of feature importance measures, making them a powerful and versatile tool for various classification and regression tasks.

(vi) OneVsRest Classifier:

The One-vs-Rest (OvR) classifier, also known as One-vs-All, is a strategy for extending binary classification algorithms to multi-class problems. It works by training a separate binary classifier for each class, where each classifier is trained to distinguish that specific class from all other classes combined. During prediction, the classifier that produces the highest confidence score is selected as the predicted class. Essentially, it decomposes a multi-class problem into multiple binary classification problems, allowing the use of any binary classifier, such as logistic regression or support vector machines, for multi-class classification. While simple and widely applicable, OvR can suffer from class imbalance issues if one class significantly outweighs the others, and its performance can be affected by the choice of the underlying binary classifier.

(vii) Stochastic Gradient Descent Classifier:

The Stochastic Gradient Descent (SGD) classifier is a linear model that optimizes a loss function by iteratively updating its parameters based on the gradient of the loss with respect to a single, randomly selected training example or a small batch of examples. This "stochastic"

approach makes it computationally efficient, particularly for large datasets, as it avoids calculating the gradient over the entire training set at each step. SGD can be used with various loss functions, such as hinge loss (for SVMs) or logistic loss (for logistic regression), and can handle a variety of classification tasks. Its performance is sensitive to the learning rate and regularization parameters, which require careful tuning. While it can converge faster than batch gradient descent, the stochastic nature of its updates can lead to noisy convergence, requiring techniques like learning rate scheduling or momentum to improve stability and performance.

3.3.2 Ensemble Learning

Ensemble learning improves classification performance by averaging several models, minimizing overfitting through the reduction of biases in single classifiers. It enhances stability and accuracy by reducing misclassification risks, providing more accurate predictions. It further exploits diversity in models, as various classifiers extract different sentiment features, resulting in a more detailed analysis. By combining multiple models, ensemble learning also increases generalization, making the system more robust to real-world text variations and enhancing its overall performance in sentiment classification.

Voting Mechanism

We employ Hard Voting, where each model votes for a sentiment class, and the majority determines the final prediction. This improves robustness in discrete classification tasks.We evaluate eight ensemble classifiers, each exploring different model combinations. This approach allows us to identify the most effective configuration based on performance metrics (accuracy, recall, precision, F1-score) on each emotion (anger, joy, fear sad).

3.3.3 Input

Text cleaning and tokenization remove unwanted characters and split sentences into individual words. Stop-word removal and stemming then filter out uninformative words and standardize word forms to enhance consistency. The cleaned text is converted into numerical representations using TF-IDF feature extraction, ensuring that significant words contribute effectively to the classification process. Finally, the processed input is classified using our trained ensemble model, providing accurate sentiment predictions. To further enhance usability, we integrated the Google Speech Recognition API for speech-based sentiment classification.

Google Cloud Speech-to-Text API:

The Google Cloud Speech-to-Text API leverages advanced machine learning to transcribe audio into text. Integration involves setting up a Google Cloud project, enabling the API, and using client libraries to send audio data, either streaming or as files. Customization options, like language selection and speaker diarization, enhance transcription quality. Advantages include high accuracy, multi-language support, and real-time processing. However, costs can accrue with high usage, and accuracy is sensitive to audio quality and background noise. This API finds applications in diverse

areas. It powers transcription services, voice-controlled applications, and customer service solutions. It also improves accessibility through live captioning and enables the creation of searchable audio archives. Its versatility makes it a valuable tool for developers seeking to integrate voice recognition into their applications. Proper integration of the Google Speech-to-Text API involves several key steps. First, you must set up a Google Cloud Platform (GCP) project and enable the Speech-to-Text API. This involves creating a service account and obtaining the necessary authentication credentials. Developers can then utilize client libraries available in various programming languages to send audio data to the API. Audio can be sent in real-time streaming or as pre-recorded files. The API offers customization options, such as specifying the language, audio encoding, and enabling features like speaker diarization. Understanding the API's documentation and best practices is crucial for optimizing performance and ensuring accurate transcriptions.

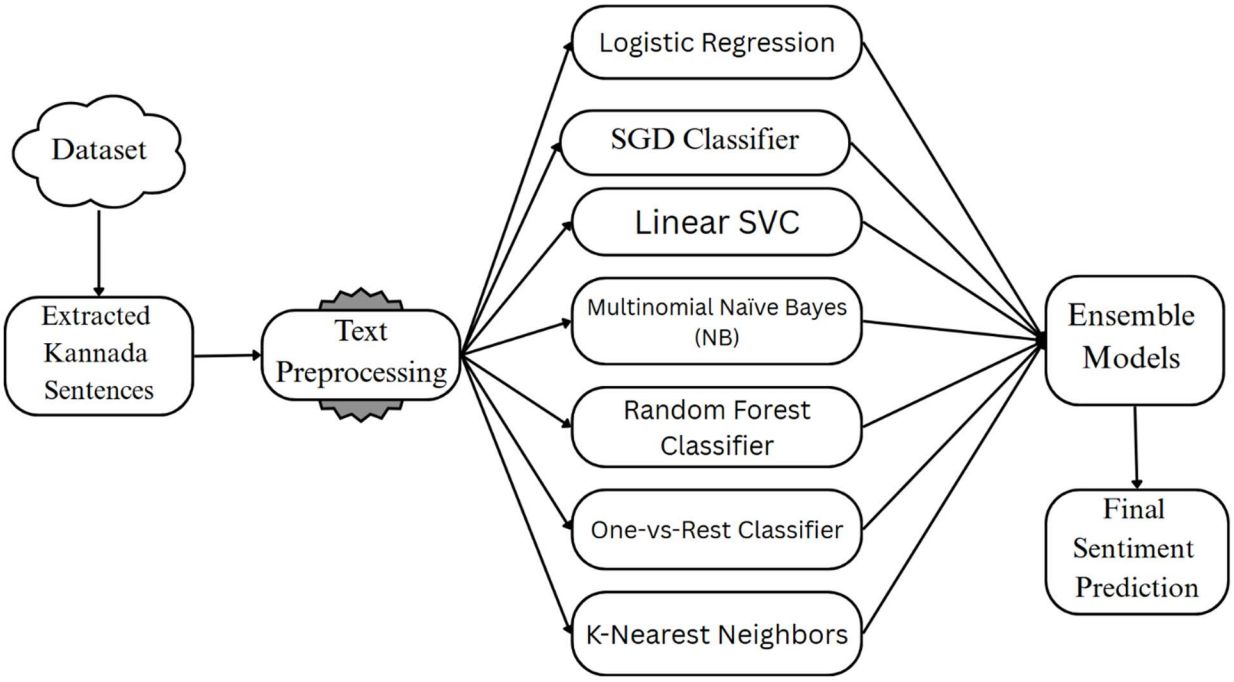

Figure 2. Ensemble Model architecture

3.4 Performance Measures for Kannada Sentiment Analysis

The evaluation of the proposed Kannada sentiment analysis model relies on key classification metrics that measure how well the model predicts sentiment labels. These metrics—accuracy, precision, recall, and F1-score—help assess the balance between correctly classified sentiments and the misclassifications, ensuring the effectiveness of the classifier in understanding Kannada text.

3.4.1 Accuracy:

Accuracy measures the overall correctness of the model's predictions by computing the ratio of correctly classified samples to the total number of samples. It is mathematically represented as:

$$\text{Accuracy} = (TP + TN) / (TP + TN + FP + FN) \quad (7)$$

where TP (True Positives) and TN (True Negatives) indicate correctly classified positive and negative sentiments, while FP (False Positives) and FN (False Negatives) represent misclassified instances. While accuracy provides a general idea of model performance, it may not be reliable when class distribution is imbalanced, which is a common scenario in Kannada sentiment analysis, where negative or neutral sentiments might be more frequent than positive ones.

3.4.2 Precision:

Precision focuses on the correctness of positive sentiment predictions by measuring the proportion of predicted positive instances that are actually positive. It is defined as:

$$\text{Precision} = TP / (TP + FP) \quad (8)$$

A high precision score indicates that the model makes fewer false positive errors, which is crucial for Kannada sentiment analysis when distinguishing between strongly positive and neutral sentiments. For instance, if a classifier incorrectly marks neutral opinions as positive, it may distort the sentiment distribution in business or social media applications where sentiment polarity is critical.

3.4.3 Recall:

Recall, also known as sensitivity, measures how well the model identifies actual positive cases by calculating the proportion of true positives captured out of all actual positive instances. The formula is:

$$\text{Recall} = TP / (TP + FN) \quad (9)$$

In Kannada sentiment analysis, recall is essential when missing out on positive sentiments is more problematic than including a few false positives. This is especially relevant in applications like customer feedback analysis, where failing to identify dissatisfaction in Kannada text may lead to overlooked customer concerns.

3.4.4 F1-score:

F1-score provides a balanced measure between precision and recall by computing their harmonic mean, ensuring that both false positives and false negatives are accounted for equally. It is expressed as:

$$F1 = 2 \times (\text{Precision} \times \text{Recall}) / (\text{Precision} + \text{Recall}) \quad (10)$$

Since Kannada sentiment datasets often contain varying proportions of sentiments, F1-score is useful in ensuring that the model does not favor one class over the other. A classifier with a high F1-score effectively balances the trade-off between misclassifications, making it particularly relevant when dealing with linguistic nuances in Kannada, where sentiment words may have different contextual meanings.

By analyzing these metrics collectively, the Kannada sentiment analysis model can be refined to ensure optimal performance, addressing potential biases in classification and improving the overall sentiment prediction accuracy.

IV. RESULTS

The ensemble classification approach was implemented using multiple Voting Classifiers, each integrating different machine learning algorithms to enhance sentiment classification accuracy and robustness. The goal of this ensemble methodology was to leverage the strengths of multiple models, reducing bias, variance, and overfitting while ensuring more reliable sentiment predictions. The performance of each classifier was evaluated based on accuracy, precision, recall, and F1-score, and misclassifications were analyzed using confusion matrices.

4.1 Performance of Individual Models

In this study, a diverse set of machine learning classifiers has been implemented and evaluated for multi-class sentiment analysis across four emotion categories: Anger, Fear, Joy, and Sad. The classifiers examined include K-Nearest Neighbors, Linear Support Vector Classifier (LinearSVC), Logistic Regression, Multinomial Naïve Bayes, OneVsRestClassifier, Random Forest Classifier, Stochastic Gradient Descent (SGD) Classifier, and Support Vector Classifier (SVC). These models were systematically compared using key performance metrics such as Accuracy, F1-Score, Precision, and Recall to assess their effectiveness in accurately detecting and classifying emotional sentiments from textual data.

The performance comparison of various classifiers across four sentiment categories—Anger, Fear, Joy, and Sad—is presented using metrics such as Accuracy, F1-Score, Precision, and Recall as observed in Table 2. Among all models, the OneVsRestClassifier achieves the highest overall accuracy (0.87), with consistently strong performance across all sentiment classes, particularly in F1-Score and Recall. It is closely followed by SVC (0.86 accuracy) and LinearSVC (0.85), both showing high precision and recall, especially for Joy and Sad sentiments. K-Nearest Neighbors shows the lowest performance overall, with the lowest accuracy (0.73) and less consistent scores in precision and recall. Multinomial Naïve-Bayes and Logistic Regression perform well in precision but lag slightly in recall compared to the top classifiers. Overall, the OneVsRestClassifier stands out as the most balanced and effective model for multi-class sentiment classification across these metrics.

Table 2. Performance comparison of individual classifiers

Metrics	Accuracy	F1-Score				Precision				Recall			
		Sentiments				Sentiments				Sentiments			
Models		**Anger**	**Fear**	**Joy**	**Sad**	**Anger**	**Fear**	**Joy**	**Sad**	**Anger**	**Fear**	**Joy**	**Sad**
K-Neighbours Classifier	0.73	0.73	0.72	0.74	0.72	0.82	0.89	0.63	0.73	0.67	0.61	0.9	0.7
LinearSVC	0.85	0.82	0.87	0.88	0.82	0.83	0.89	0.87	0.82	0.82	0.85	0.9	0.83
Logistic Regression	0.84	0.84	0.84	0.87	0.82	0.84	0.93	0.85	0.8	0.84	0.78	0.89	0.84
Multinomial Naïve-Bayes	0.83	0.82	0.83	0.84	0.82	0.83	0.95	0.78	0.82	0.81	0.74	0.92	0.82
OneVsRestClassifier	0.87	0.86	0.88	0.9	0.85	0.86	0.92	0.87	0.86	0.86	0.85	0.93	0.84
Random Forest Classifier	0.85	0.85	0.89	0.85	0.82	0.89	0.95	0.8	0.81	0.81	0.83	0.91	0.84
SGD Classifier	0.84	0.81	0.86	0.88	0.8	0.82	0.87	0.87	0.8	0.8	0.86	0.9	0.8
SVC	0.86	0.85	0.85	0.9	0.84	0.86	0.93	0.88	0.81	0.85	0.79	0.92	0.87

4.2 Performance of Voting Classifiers

The ensemble classifiers' performance analysis shows differing levels of classification efficacy and accuracy. With an accuracy of 87%, Voting Classifier 1, which combines SVC, Random Forest, and SVM, proved to be highly capable in sentiment analysis. Remarkably, it properly identified 152 cases of sadness, demonstrating a high recall for this emotion. Joy and Fear came in second and third, respectively, with 140 and 135 correctly classified cases. Minor misclassifications were noted despite its efficacy, such as Happy being mistakenly identified as Fear (17 times) and Fear being mistaken for Joy (10 times).

Combining SGD, Logistic Regression, and KNN, Voting Classifier 2 achieved an accuracy of 83%, which is marginally less than the first ensemble. Joy had the most accurate classifications (147), followed by anger (88) and fear (132). There were a few misclassifications, too, most notably the 13 occasions in which Fear was mistaken for Happy and the 6 instances in which Sadness was mistaken for Anger. The classifier demonstrated its efficacy in sentiment prediction by doing well in spite of these flaws.

With an accuracy of 87%, Voting Classifier 3, which combines Multinomial NB, SVC, and Linear SVC, demonstrated its reliability in operation. With an F1-score of 0.90 for Joy, this classifier demonstrated exceptional recall and precision. Anger and Fear received F1-scores of 0.86 and 0.85, respectively, while Sadness came in second with an F1-score of 0.85. There were just a few but significant misclassifications, such as Sadness being mistaken for Anger five times and Fear being mistaken for Joy eleven times.

Voting Classifier 4, which included SVC, Random Forest, SVM, and Gradient Boosting, performed well, with an accuracy of 86%. Fear demonstrated its efficacy in recognizing this

emotion with the maximum recall of 88% and an F1-score of 0.89. Anger and Sadness each received an F1-score of 0.84, while Joy came in second with a 93% recall and 0.88. Although there were a few misclassifications, such as Sadness being mistaken for Fear in 14 cases and Happy being mistaken for Fear in 17 cases, the classifier generally functioned dependably.

With an accuracy of 87%, Voting Classifier 5, which combines SVC, Random Forest, SVM, Decision Trees, and Extra Trees, demonstrated strong performance. With an F1-score of 0.88, Fear showed the highest recall of 86%, followed by Anger and Sadness, with F1-scores of 0.85 and 0.86, respectively. Joy was one of the easiest emotions to identify, with an F1-score of 0.87 and a 92% recall rate. Among the misclassifications were the confusion of Sadness with Fear (14 cases) and Happy with Sadness (12 cases).

With the lowest accuracy of 70%, voting classifier 6, which combined AdaBoost and XGBoost, had mediocre performance. Joy had a 94% recall rate, but its F1-score of 0.67 suggested that it was less precise. Sadness has a poor F1-score of 0.50 due to its poor recall (34%), despite its excellent precision (0.97). With F1-scores of 0.80 and 0.82, respectively, Anger and Fear outperformed. There were a lot of misclassifications, such as Sadness being mistaken for Happy (93 times) and Joy being mistaken for Sadness (28 times), which suggested that more work needed to be done.

With an accuracy of 82%, Voting Classifier 7, which combines Random Forest, KNN, Multinomial NB, AdaBoost, and XGBoost, demonstrated dependable sentiment categorization. Fear performed well, as evidenced by its highest precision (0.95) and F1-score of 0.87. Joy's performance was impacted by its high recall (0.95) and low precision (0.71). The accuracy values for anger and sadness were 0.88 and 0.85, respectively, showing balanced results. Among the misclassifications were Happy being mistaken for Fear or Sadness and Joy being mistaken for Sadness (26 cases).

Lastly, the resilience of Voting Classifier 8, which integrates One Vs Rest, SVM, SVC, Random Forest, and Logistic Regression, was demonstrated with an accuracy of 87%. Fear had a little lower recall (0.83) but a good precision of 0.93, whilst Anger and Joy showed excellent precision and recall scores of roughly 0.86 and 0.92, respectively. Sadness ensured a balanced classification by consistently maintaining an F1-score of 0.85. Although there were a few minor misclassifications, such as Joy being mistaken for Sadness (6 times) and Fear being mistaken for Happy (11 times), the classifier produced robust and consistent findings generally.

Table 3 presents a comparison of all the classifiers' performance metrics, including accuracy, precision, recall, and F1-score for each sentiment class. The confusion matrices further illustrate the distribution of correct and incorrect classifications, providing deeper insight into model performance.

Table 3. A comparison of the classifier's performance

Metrics	Accuracy	F1-Score				Precision				Recall			
		Sentiments				Sentiments				Sentiments			
Ensemble Models		**Anger**	**Fear**	**Joy**	**Sad**	**Anger**	**Fear**	**Joy**	**Sad**	**Anger**	**Fear**	**Joy**	**Sad**
Classifier 1	0.87	0.84	0.88	0.9	0.85	0.84	0.91	0.88	0.85	0.85	0.85	0.92	0.84
Classifier 2	0.83	0.83	0.84	0.86	0.8	0.82	0.91	0.84	0.79	0.83	0.79	0.89	0.81
Classifier 3	0.87	0.85	0.86	0.9	0.85	0.84	0.93	0.87	0.84	0.85	0.8	0.93	0.86
Classifier 4	0.86	0.84	0.89	0.88	0.84	0.83	0.92	0.84	0.88	0.85	0.88	0.93	0.8
Classifier 5	0.87	0.85	0.88	0.87	0.86	0.87	0.91	0.83	0.87	0.84	0.86	0.92	0.86
Classifier 6	0.70	0.8	0.82	0.67	0.5	0.83	0.88	0.53	0.97	0.78	0.78	0.94	0.34
Classifier 7	0.82	0.81	0.87	0.81	0.81	0.88	0.95	0.71	0.85	0.75	0.8	0.95	0.77
Classifier 8	0.87	0.86	0.88	0.89	0.85	0.86	0.93	0.86	0.84	0.86	0.83	0.92	0.85

The analysis reveals that Voting Classifiers 1, 3, 5, and 8 consistently delivered the best results, with accuracy rates of 87%. Voting Classifier 6, with AdaBoost and XGBoost, had the weakest performance, primarily due to imbalanced recall values. Overall, the integration of multiple classifiers through ensemble learning improved classification stability, but some models exhibited specific weaknesses that require further refinement.

The results of our custom sentiment classifier for Kannada text input are shown in Figure 3. Before being classified, the preprocessed text goes through a number of steps, beginning with tokenization, stopword removal, and lemmatization to make sure the input text is clear and organized for precise analysis. The text is then run through the ensemble model, which creates a more robust and trustworthy sentiment label by combining predictions from several machine learning models (such as Support Vector Machines, Random Forests, and neural networks). The system then provides information on the overall sentiment of the input text by assigning a sentiment label, such as positive, negative, or neutral, based on the aggregated forecast of the ensemble model.

```
[]
Enter a Kannada sentence
ಅವನು ಅಳುತ್ತಿದ್ದಾನೆ
After Cleaning and Stopwords Removal
ಅವನು ಅಳುತ್ತಿದ್ದಾನೆ

After Stemming
ಅವನು ಅಳು

Output of each Algorithms
[np.str_('Sad'), 'Sad', 'Sad', np.str_('Sad'), 'Anger', np.str_('Anger'), 'Sad']
{'Joy': 0, 'Sad': 5, 'Anger': 2, 'Fear': 0}
The Sentence is classified as Sad
'Sad'
```

Figure 3. Output utilizing custom text input. Text translates to "He is crying"

The speech-to-text sentiment classification pipeline, which combines sentiment analysis and audio input processing, is shown in Figure 4. Initially, a speech-to-text model is used to convert an audio file into text. After being translated to text, the speech is subjected to preprocessing procedures such lemmatization, tokenization, and stopword removal, as shown in Figure 3. The text is fed into the same sentiment classification pipeline following the preprocessing step, which uses the ensemble model to ascertain the sentiment of the speech that has been transcribed. This method enables smooth sentiment analysis of spoken language as well as text inputs, giving the system flexibility in managing Kannada communication in both written and spoken formats.

```
Listening to the audio...
Recognized Text: ಇದು ಒಳ್ಳೆಯ ದಿನ

Extracted Text: ಇದು ಒಳ್ಳೆಯ ದಿನ

Extracted text:
ಇದು ಒಳ್ಳೆಯ ದಿನ

Output of each Algorithms
[np.str_('Joy'), 'Sad', 'Joy', np.str_('Sad'), 'Joy', np.str_('Joy'), 'Joy']
{'Joy': 5, 'Sad': 2, 'Anger': 0, 'Fear': 0}
The Sentence is classified as Joy
'Joy'
```

Figure 4. Output utilizing custom audio recording. Text translates to “It is a good day”

V. ANALYSIS

5.1 Limitations of the study:

There are several limitations to keep in mind when interpreting the results of this study. One key issue is the lack of emoji support in the dataset. Emojis play a significant role in online communication, often conveying sentiment more clearly and subtly than text alone. Their absence could limit the system’s ability to fully capture the nuances of sentiment expression.

Another concern is the relatively small dataset size—just 2,000 sentences—which may restrict how well the findings can be generalized. This dataset might not represent the full range of sentiment expressions in the Kannada language. Although efforts were made to ensure diversity by sourcing data from different online platforms, the specific details about these sources are not clearly defined, which could introduce biases related to the type of content or the demographics of the users represented.

The rule-based stemming method used, while customized for Kannada’s linguistic structure, may not handle complex variations as effectively as more advanced techniques. This could result in issues like over-stemming or under-stemming, affecting the accuracy of the analysis. Additionally, the study relies on traditional machine learning models like Logistic Regression and SVM. While these provide solid baseline performance, they may not capture the deeper contextual relationships and subtle sentiment cues that more sophisticated deep learning models could uncover.

Lastly, when it comes to audio input, the performance of the Google Cloud Speech-to-Text API can be influenced by factors such as audio quality, background noise, speaker accents, and speaking styles. These variables might lead to transcription errors, which in turn could impact the accuracy of the sentiment analysis.

5.2 Implications of the findings:

This research highlights the significant impact of ensemble classification techniques in improving both the accuracy and reliability of sentiment classification systems. A closer look at different Voting Classifiers shows that specific combinations of machine learning algorithms—particularly in Voting Classifiers 1, 3, 5, and 8—deliver consistently strong performance, with accuracy rates reaching up to 87%. This suggests that ensemble methods can be highly effective in handling the complexities of sentiment analysis, especially in linguistically diverse contexts.

The study also points out the importance of using a range of evaluation metrics, such as accuracy, precision, recall, and F1-score. By considering these different measures, we gain a clearer picture of each model's strengths and limitations, which helps in choosing the best model for a given task.

In practical terms, the findings open doors to developing more robust and versatile sentiment analysis tools for various applications. Additionally, the successful deployment of a custom sentiment classifier for Kannada text, along with the integration of Google Speech Recognition API, represents a major leap forward for real-time sentiment analysis. This progress not only enhances the accessibility of sentiment analysis technologies but also makes it possible for users to interact with systems through both text and voice, bringing us closer to more intuitive, user-friendly AI solutions.

VI. CONCLUSIONS

This project developed a sentiment analysis system specifically for the Kannada language using Natural Language Processing (NLP) and machine learning techniques. It focuses on classifying core emotions—joy, anger, fear, and sadness—from both text and speech inputs. To improve prediction reliability, we employed classifiers such as Multinomial Naive Bayes, Random Forest, and One-vs-Rest, and enhanced their performance through ensemble learning using a Hard Voting strategy. Eight different Voting Classifier combinations were tested, with Classifier 0(87.21 %), Classifier 1(86.38%), Classifier 4(84.88%), and Classifier 7(84.72%), achieving the best performance, reaching up to 87% accuracy. In contrast, models incorporating boosting techniques like AdaBoost and XGBoost showed lower effectiveness due to imbalanced recall values.

The motivation behind this project was to bridge the digital divide for Kannada speakers and promote emotion-aware interaction in regional languages. The system supports both typed and spoken inputs, allowing users to express themselves more naturally. To enable interaction via

speech, we integrated the Google Speech Recognition API, allowing users to record or upload Kannada audio, which is transcribed and processed using the same sentiment classification pipeline as text. This ensures consistent emotion detection across modalities and broadens accessibility. While real-time analysis is not yet implemented, the current system provides accurate sentiment classification from both voice and text. Looking ahead, we aim to support real-time feedback and include emoji-based sentiment detection, transforming the platform into a fully multimodal and inclusive sentiment analysis solution.

VI. REFERENCES

[1] Eshwarappa, S. M., & Shivasubramanyan, V. (2024). Enhancing sentiment analysis in Kannada texts by feature selection. International Journal of Power Electronics and Drive Systems/International Journal of Electrical and Computer Engineering, 14(6), 6572. https://doi.org/10.11591/ijece.v14i6.pp6572-6582

[2] Chattu, K., & Sumathi, D. (2024). Sentiment analysis using deep learning approaches on Multi-Domain Dataset in Telugu language. Journal of Information & Knowledge Management, 23(03). https://doi.org/10.1142/s0219649224500187

[3] Divate, M. S. (2021). Sentiment analysis of Marathi news using LSTM. International Journal of Information Technology, 13(5), 2069–2074. https://doi.org/10.1007/s41870-021-00702-1

[4] S, S., & KV, P. (2020). Sentiment analysis of malayalam tweets using machine learning techniques. ICT Express, 6(4), 300–305. https://doi.org/10.1016/j.icte.2020.04.003

[5] Hoque, M. N., Salma, U., Uddin, M. J., Ahamad, M. M., & Aktar, S. (2024). Exploring transformer models in the sentiment analysis task for the under-resource Bengali language. Natural Language Processing Journal, 8, 100091. https://doi.org/10.1016/j.nlp.2024.100091

[6] Li, X. (2021). Text sentiment analysis of German multilevel features based on SelfAttention Mechanism. Security and Communication Networks, 2021, 1–12. https://doi.org/10.1155/2021/8309586

[7] Hande, A., Hegde, S. U., Priyadharshini, R., Ponnusamy, R., Kumaresan, P. K., Thavareesan, S., & Chakravarthi, B. R. (2021). Benchmarking Multi-Task Learning for Sentiment Analysis and Offensive Language Identification in Under-Resourced Dravidian Languages. arXiv (Cornell University). https://doi.org/10.48550/arxiv.2108.03867

[8] Lopes, É. P., Freitas, L., Gomes, G., Lemos, G., Hammes, L. O., & Corrêa, U. B. (2022). Exploring BERT for Aspect-based Sentiment Analysis in Portuguese Language. Proceedings of the . . . International Florida Artificial Intelligence Research Society Conference, 35. https://doi.org/10.32473/flairs.v35i.130601

[9] Kakde, K., & Padalikar, H. M. (2022). Context-based Sentiment analysis of Indian Marathi Text using Deep Learning. International Journal on Recent and Innovation Trends in Computing and Communication, 10(11), 71–76. https://doi.org/10.17762/ijritcc.v10i11.5782

[10] Khan, L., Amjad, A., Ashraf, N., Chang, H., & Gelbukh, A. (2021). Urdu sentiment analysis

with deep learning methods. IEEE Access, 9, 97803–97812. https://doi.org/10.1109/access.2021.3093078
[11] Dutta, S., Agrawal, H., & Roy, P. K. (2021, December). Sentiment Analysis on Multilingual Code-Mixed Kannada Language. In FIRE (Working Notes) (pp. 908-918).
[12] Kalaivani, A., & Thenmozhi, D. (2021). Multilingual Sentiment Analysis in Tamil, Malayalam, and Kannada code mixed social media posts using MBERT. FIRE (Working Notes), 1020-1028.
[13] Shetty, S., Hegde, S., Shetty, S., Shetty, D., Sowmya, M. R., Shetty, R., ... & Shetty, Y. (2022). Sentiment Analysis of Twitter Posts in English, Kannada and Hindi languages. In Recent Advances in Artificial Intelligence and Data Engineering: Select Proceedings of AIDE 2020 (pp. 361-375). Springer Singapore. https://doi.org/10.1007/978-981 16-3342-3_29
[14] Rakshitha, K., Ramalingam, H. M., Pavithra, M., Advi, H. D., & Hegde, M. (2021). Sentimental analysis of Indian regional languages on social media. Global Transitions Proceedings, 2(2), 414-420. https://doi.org/10.1016/j.gltp.2021.08.039
[15] Chandrika, C. P., Kallimani, J. S., Adarsha, H. P., Nagabhushan, A., & Chavan, A. (2021). Rule-Based Approach for Emotion Detection for Kannada Text. In Proceedings of International Conference on Communication and Artificial Intelligence: ICCAI 2020 (pp. 463-472). Springer Singapore. https://doi.org/10.1007/978-981-33-6546-9_44
[16] Abbaschian, B. J., Sierra-Sosa, D., & Elmaghraby, A. (2021). Deep learning techniques for speech emotion recognition, from databases to models. Sensors, 21(4), 1249. https://doi.org/10.3390/s21041249
[17] Shah, P., Swaminarayan, P., & Patel, M. (2022). Sentiment analysis on film review in Gujarati language using machine learning. International Journal of Electrical and Computer Engineering, 12(1), 1030. https://doi.org/10.11591/ijece.v12i1.pp1030-1039
[18] Liu, J., Li, K., Zhu, A., Hong, B., Zhao, P., Dai, S., & Su, H. (2024). Application of deep learning-based natural language processing in multilingual sentiment analysis. Mediterranean Journal of Basic and Applied Sciences (MJBAS), 8(2), 243-260. https://doi.org/10.46382/MJBAS.2024.8219
[19] Muhammad, S. H., Abdulmumin, I., Ayele, A. A., Ousidhoum, N., Adelani, D. I., Yimam, S. M., & Arthur, S. (2023). Afrisenti: A twitter sentiment analysis benchmark for african languages. arXiv preprint arXiv:2302.08956. https://doi.org/10.48550/arXiv.2302.08956 .
[20] Hegde, A., Anusha, M. D., Coelho, S., Shashirekha, H. L., & Chakravarthi, B. R. (2022, June). Corpus creation for sentiment analysis in code-mixed Tulu text. In Proceedings of the 1st Annual Meeting of the ELRA/ISCA Special Interest Group on Under-Resourced Languages (pp. 33-40).
[21]Preprocessing Steps for Natural Language Processing (NLP): A Beginner's Guide https://medium.com/@maleeshadesilva21/preprocessing-steps-for-natural-languageprocessing-nlp-a-beginners-guide-d6d9bf7689c9 Accessed on: 24.01.2025
[22] NLP — Getting started with Sentiment Analysis https://medium.com/analyticsvidhya/nlp-getting-started-with-sentiment-analysis-126fcd61cc4a Accessed on: 24.01.2025

[23]Speech Recognition and Synthesis — Google Cloud Speech-to-Text & Text-to-Speech APIs (Python) https://medium.com/@pysquad/speech-recognition-and-synthesis-google-cloudspeech-to-text-text-to-speech-apis-python-211c8154bd3b Accessed on: 24.01.2025