Novel Ensemble Sentiment Classification through Speech Processing and Stacking Generalization

Shriya Chowdhury
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
shriya.chowdhury2021@vitstudent.ac.in

Soumo Roy
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
soumo.roy2021@vitstudent.ac.in

Mathew John
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
mathew.john2021@vitstudent.ac.in

V Rama Chandra Chathurvedi
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
ramachandra.chathur2021@vitstudent.ac.in

Deepanjali Das
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
deepanjali.das2021@vitstudent.ac.in

Aparna Mohanty
School of Electronics Engineering
Vellore Institute of Technology
Vellore, India
aparna.mohanty@vit.ac.in

***Abstract*— This study introduces an innovative method for speech sentiment analysis, employing a stacking classifier. By directly transcribing audio, the system extracts essential features for sentiment analysis. The stacking classifier, which combines multiple classifiers, enhances predictive performance. Intermediate models utilized include k-NN, Random Forest, SVM, Gaussian Naive Bayes, and Logistic Regression. OpenAI Whisper's transformer was used for audio-to-text conversion. The methodology encompasses audio data preprocessing, text conversion, natural language processing, and model training. Evaluation metrics, such as accuracy, precision, recall, and F1-score, are used to assess performance. Through empirical investigation, the study examines the effectiveness of various classifiers and feature sets in sentiment analysis of speech data. Experimental results demonstrate the system's ability to accurately classify sentiment, providing valuable insights into sentiment analysis methods tailored for spoken language.**

***Keywords*— Sentiment Analysis, Transformers, Stacking Classifier, Natural Language Processsing, Speech Processing**

I. Introduction

In today's fast-paced world, effectively managing human emotions through speech is crucial. Traditional emotion assessment methods struggle with conversational nuances, necessitating more sophisticated strategies. With communication disorders affecting around 10% of the global population, understanding and navigating emotions through dialogue is essential. Machine learning algorithms, offer promising avenues for speech analysis. This research explores advanced sentiment analysis techniques, vital for fields like market research and psychology. Our aim is to develop robust speech recognition systems, integrating sentiment analysis to discern emotional states accurately. Beyond academia, this integration enhances user experiences, refines customer interactions, and improves financial forecasting models. By integrating text and voice data, we can enhance forecast accuracy and inform decision-making processes. Through interdisciplinary research, we aim to advance philosophy, machine learning, and technology to address societal challenges and support diverse human experiences.

A. Literature review

Speech sentiment analysis involves extracting emotional nuances from spoken language, a task historically challenging due to the complexity of human speech. Recent advancements in deep learning have significantly improved sentiment analysis accuracy, paving the way for real-world applications [1]. A crucial stage in speech sentiment analysis is feature extraction, which entails locating and removing pertinent acoustic characteristics from speech data [2]. To obtain greater accuracy, the extraction of the unique discriminating feature of the speaker is crucial. The correctness of this phase is crucial since it provides the input for the next phase [3]. Several studies have explored the fusion of PyAudio and stacking classifiers for speech sentiment analysis. By harnessing PyAudio's real-time audio processing capabilities and the ensemble learning power of stacking classifiers, researchers have achieved remarkable accuracy in sentiment classification. This integration facilitates efficient sentiment analysis on streaming audio data, making it suitable for various applications [4]. The method utilizes ASR features, specifically log-mel features, extracted from an end-to-end ASR model, followed by sentiment analysis using a pre-trained sentiment decoder, such as an RNN with self-attention to mitigate overfitting during training, SpecAugment is applied to the model [5]. This work introduces sentiment profiles (SPs) to capture segment-level sentiment fluctuations in speech sentiment analysis, utilizing multi-modal wild video data and facial expression knowledge to generate soft labels. Additionally, a Speech Encoder Module is designed to encode audio segments into SPs, with further refinement achieved through the sentiment profile purifier (SPP), resulting in state-of-the-art performance on CH-SIMS and IEMOCAP datasets [6]. This paper proposes a speech sentiment analysis model using spectrogram features, deep learning, and bi-directional-recurrent neural networks. The model outperforms traditional Automatic Speech Recognition models in Word Error Rate (WER) and Character Error Rate (CER), with improved accuracy of 90% using logistic Regression algorithms [7]. This study proposes a deep learning architecture that combines managerial emotional states extracted from speech emotion recognition with Fin-BERT-based sentiment analysis of earnings conference call transcripts, achieving more accurate financial distress prediction, compared to traditional sentiment indicators [8]. This paper presents a sentiment analysis model using 1D CNN architecture, incorporating acoustic features extracted from audio files, including wave plot and spectrogram features, augmented with data augmentation techniques, resulting in improved classification accuracy compared to existing methodologies [9]. This article proposes a hybrid approach combining unsupervised machine learning and natural language processing techniques to extract sentiment and opinions from reviews of scientific articles. The method utilizes part-of-speech tagging and dictionaries to determine the semantic orientation of reviews, showing improvements over classical machine learning algorithms in binary, ternary, and 5-point scale classifications, while also presenting challenges in multiclass classification [10].

B. *Motivation*

Speech Traditional sentiment analysis methods often fall short when applied to speech due to the complexities of vocal expression, including tone, intonation, and emphasis. The goal is to create more reliable and accurate spoken language sentiment analysis systems by utilizing real-time audio processing capabilities with stacking classifiers' ensemble learning capabilities. Furthermore, there is an increasing need for speech-based sentiment analysis solutions due to the widespread use of voice-activated products and services. This study meets this need by offering practical methods for real-time analysis and comprehension of spoken language's emotional content.

II. Methodology

The speech-to-text conversion was conducted using two distinct methods: first, leveraging the speech_recognition module, and second, by using transformers.

A. *Python Speech Recognition Library*

The first way this was achieved was by making using of the speech_recognition library in python in conjunction with the pydub library, the latter used for detecting silences within audio files.

The transcription function serves as the main procedure for converting audio to text. It begins by loading an audio file using AudioSegment from pydub, which represents the audio data. Then, the audio is split into chunks based on silence using the split_on_silence function, with parameters such as minimum silence length and silence threshold specified. These chunks are stored in a designated folder named "audio-chunks". Each segment undergoes individual processing within a loop. Sequentially, each chunk is exported as an autonomous WAV file within the specified directory. Employing the Recognizer object from speech_recognition, the script transcribes each chunk leveraging Google's speech recognition API (recognize_google). In the event of encountering any unrecognizable segments, the script gracefully handles such occurrences, emitting an error message as notification.

The transcribed text from each chunk is capitalized and appended to a cohesive string. Upon the completion of processing all segments, this string containing the transcriptions is returned. Additionally, the program facilitates saving the transcribed text to a CSV file, which can be made use of for further processing.

B. *Transformers*

The second method transcribes long audio recordings using the OpenAI Whisper model, and then saving the transcription into a CSV file.

It uses the Transformers library to implement a speech transcription pipeline. The pipeline, employs a pre-trained model for automatic speech recognition (ASR). The specific model used is the OpenAI Whisper model, tailored for ASR tasks. The process begins by checking for GPU availability to decide whether to utilize CUDA acceleration for computational efficiency. Once initialized, the pipeline is set up with the chosen ASR model and the appropriate device configuration. Initially, it loads the audio file specified by the user. The audio is then resampled to a standardized sample rate of 16,000 samples per second to ensure consistency. Subsequently, the resampled audio is passed through the ASR pipeline for transcription. To handle long audio files efficiently, the transcription process is chunked into smaller segments. This allows for more manageable processing, especially for lengthy recordings. Parameters can be passed to control the duration of each chunk and the overlap between consecutive chunks as well.

The obtained transcription is returned for further analysis or processing. Additionally, the transcription is saved to a CSV file as well.

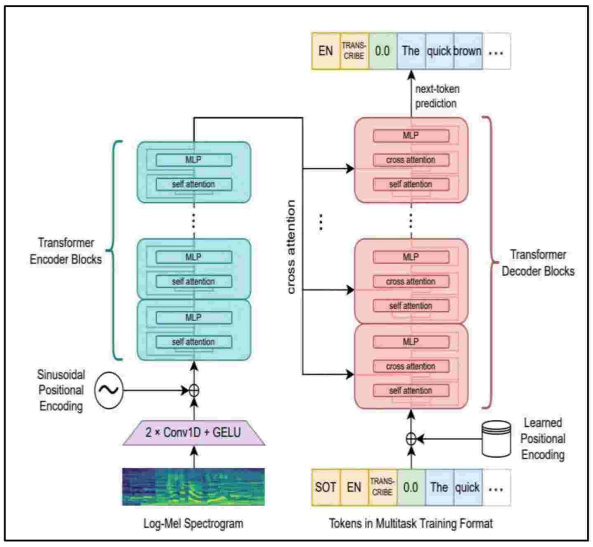


Fig. 1. OpenAI Whisper internal neural network structure

C. *Intermediate Models for Stacking Classifier*

The five intermediate machine learning models used are the K-Nearest Neighbours Classifier, Random Forest Classifier, Support Vector Classifier, Gaussian Naïve Bayes Classifier and Logistic Regression.

The k-Nearest Neighbors (k-NN) algorithm is a classification method where the class of an unseen instance is determined by the classes of its k nearest neighbors in the feature space. For a new instance x, compute the distance between x and each training instance x_i using a distance metric such as Euclidean distance shown in equation 1 –

$$d(x, xi) = \sqrt{\sum_{j=1}^{n}(x - x_i)^2} \qquad (1)$$

The training instances with the smallest distances to x are selected. The class label y is assigned to x based on a majority vote among its k nearest neighbors shown in equation 2 -

$$y = \arg\max \sum_{i=1}^{k} \mathrm{I}(yi = c) \qquad (2)$$

Where, y is the predicted class label for x, c iterates over each unique class label, y_i is the class label of the i-th nearest neighbor to x, I() is the indicator function, returning 1 if its argument is true and 0 otherwise. The value of k influences the model's bias-variance tradeoff, with smaller k values leading to more complex decision boundaries but potentially higher variance. Method is shown in Fig. 1.

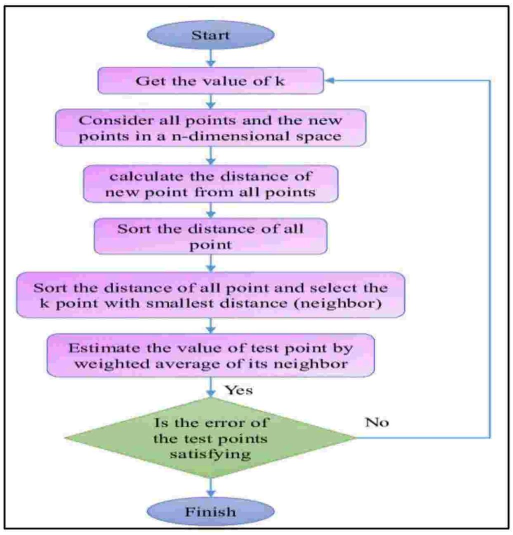

Start
Get the value of k
Consider all points and the new points in a n-dimensional space
calculate the distance of new point from all points
Sort the distance of all point
Sort the distance of all point and select the k point with smallest distance (neighbor)
Estimate the value of test point by weighted average of its neighbor
Yes
Is the error of the test points satisfying
No
Finish

Fig. 2. Working of KNN Algorithm

The next model used was Random Forest Classifier. Random Forest 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 the mean prediction (regression) of the individual trees.

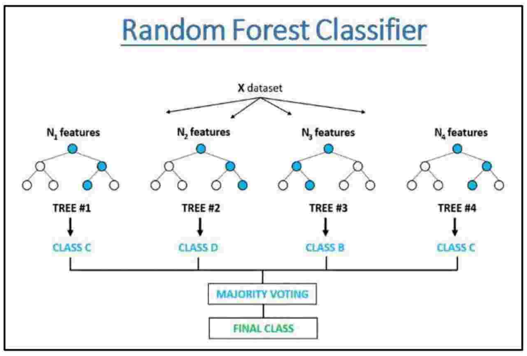


Fig. 3. Working of Random Forest Classifier

The method, shown in Fig. 2, involves the processes of Bootstrapped Sampling, Feature Randomness, Tree Building and Voting. Bootstrapped Sampling involves randomly selecting samples from the dataset with replacements to create multiple subsets of the data. For each tree, randomly select a subset of features at each split point, adding an element of randomness, and this process is called Feature Randomness. Tree Building is based on these subsets. At each node, choose the best split among a random subset of features. For classification, each tree "votes" for the class. The class with the most votes is the predicted class. For regression, the average prediction of all the trees is taken.

Support Vector Classifier (SVC) is a powerful supervised learning algorithm used for classification tasks. It works by finding the hyperplane that best separates different classes in the feature space. Given a set of training data points (x_1, y_1), (x_2, y_2), …, (x_n, y_n), where x_i represents the feature vector and y_i represents the class label (either -1 or 1), the optimal hyperplane is the one that maximizes the margin, which is the distance between the hyperplane and the nearest data point from either class. The equation for the hyperplane is given by equation 3, where w is the weight vector, perpendicular to the hyperplane, x is the feature vector and b is the bias term (intercept) –

$$wx + b = 0 \qquad (3)$$

The decision function for the classification through support vector machine is shown in equation 4, where f(x) is the predicted class label, α_i are Langrange's multipliers and $K(x, xi)$ is the kernel function, allowing the algorithm to handle non-linear decision boundaries:

$$f(x) = \mathrm{sign}(\textstyle\sum_{i=1}^{n} \alpha_i y_i K(x, xi) + b) \qquad (4)$$

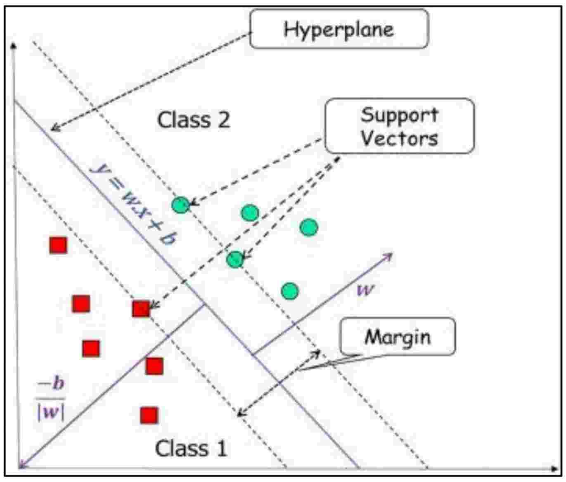


Fig. 4. Working of Support Vector Machine

The fourth model used was Gaussian Naïve Bayes model. Gaussian Naive Bayes (GNB) is a classification algorithm based on Bayes' theorem with the assumption of feature independence. Given a feature vector x and a set of classes C, the posterior probability of class C_k given x is calculated as shown in equation 5 -:

$$P(Ck|x) = P(x)P(x|Ck)\cdot P(Ck) \qquad (5)$$

$P(Ck \mid x)$ is the posterior probability of class C_k given x, $P(x \mid Ck)$ is the likelihood of observing x given class C_k , modeled as a multivariate Gaussian distribution with mean μ_k. GNB assumes that features are conditionally independent given the class, simplifying the likelihood calculation to equation 6:

$$P(x \mid Ck) = \textstyle\prod_{i=1}^{n} P(xi \mid Ck) \quad (6)$$

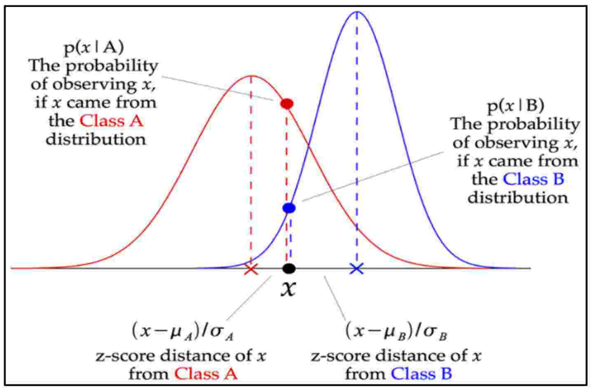


Fig. 5. Gaussian Naïve Bayes Distribution

The fifth intermediate model was Logistic Regression. Logistic Regression is a widely used supervised learning algorithm for binary classification tasks. Despite its name, logistic regression is a classification algorithm rather than a regression one. It models the probability that a given input data point belongs to a particular class. Logistic regression constructs a model shown in equation 7 -

$$p(X; b, w) = \frac{e^{w \cdot X + b}}{1 + e^{w \cdot X + b}} = \frac{1}{1 + e^{-w \cdot X + b}} \quad (7)$$

During training, the model parameters are optimized to maximize the likelihood of the observed data. This is typically achieved through techniques such as maximum likelihood estimation or gradient descent. Pictorial representation is shown in Fig. 6.

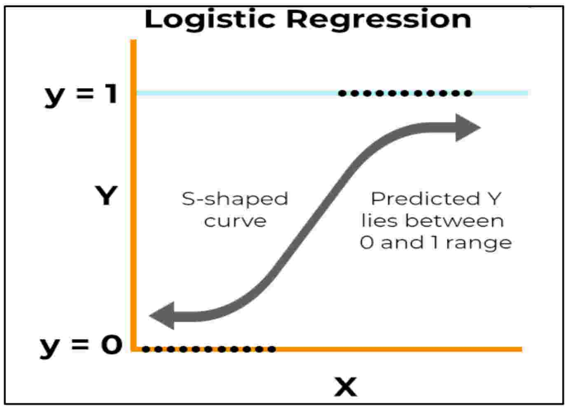


Fig. 6. Logistic Regression Representation

D. *Stacking Classifier*

Stacking, also known as Stacked Generalization, is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The basic idea is to train several base models, use their predictions as input features, and then use a meta-model to learn how to best combine the base models.

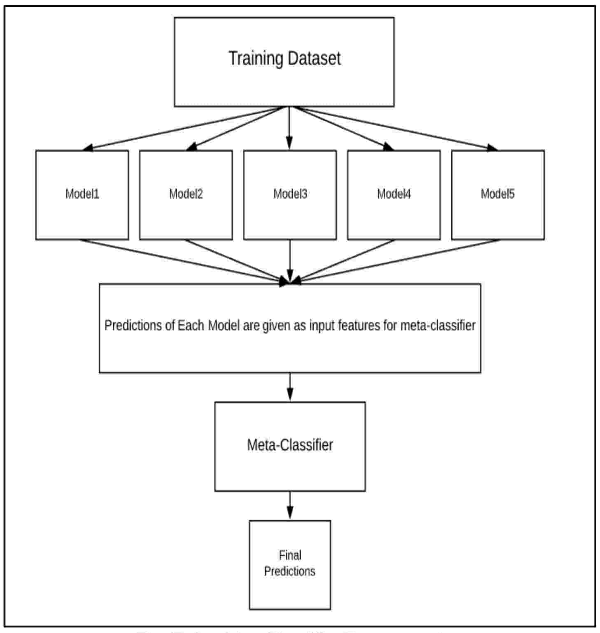


Fig. 7. Stacking Classifier Representation

Stacked Generalization, is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The basic idea is to train several base models, use their predictions as input features, and then use a meta-model to learn how to best combine the base models. Stacking classifier works by training the classifiers on the training data. These classifiers can be different algorithms or the same algorithm with different hyperparameters. First-level Predictions are made on the validation (or test) set using these base models. A new feature matrix is created by concatenating the predictions from the base models with the original features. Train a meta-classifier (or meta-regressor) on the new feature matrix. In this project we have used the logistic regression model as the eta classifier. Make predictions on the new data using the base models, then use the predictions as input features for the meta-model to get the final prediction.

The training dataset contains 20,800 entries of positive, negative and neutral sentiment texts. This dataset is divided into 80 percent as the training and 20 percent into the test dataset. Natural language processing is used to clear the text of the stopwords present in the English dictionary. The text column of the dataset is converted into numerical weights data array using a TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. The TF-IDF vectorizer carries out these calculations in equations 8, 9,10 and returns the TF-IDF weighted document-term matrix where t is a term, d is a document, D is a corpus, N is the total number of documents in the corpus and df(t,D) is the number of documents in the corpus that contain the term t–

$$TF(t, d) = \frac{\text{Number of times term t appears in document d}}{\text{Total number of terms in document d}} \quad (8)$$

$$IDF(t, D) = \log(1 + \mathrm{df}(t, D)N) + 1 \quad (9)$$

$$TFIDF(t, d, D) = TF(t, d) \times IDF(t, D) \quad (10)$$

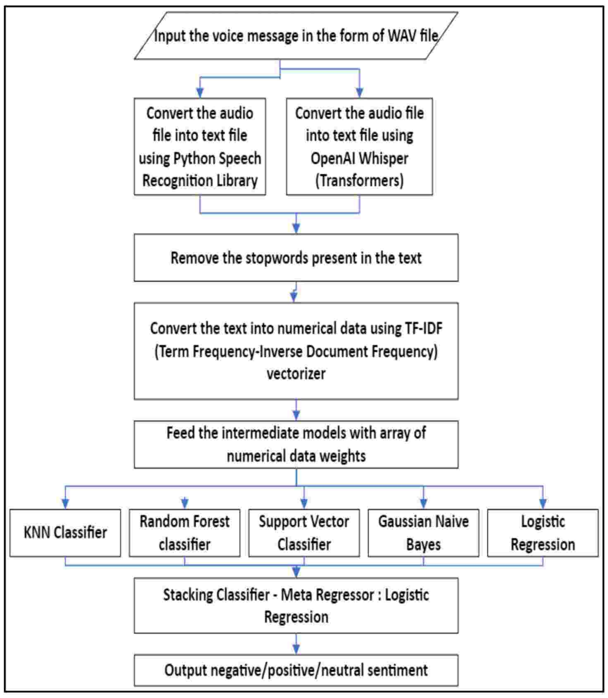

Input the voice message in the form of WAV file
Convert the audio file into text file using Python Speech Recognition Library
Convert the audio file into text file using OpenAI Whisper (Transformers)
Remove the stopwords present in the text
Convert the text into numerical data using TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer
Feed the intermediate models with array of numerical data weights
KNN Classifier
Random Forest classifier
Support Vector Classifier
Gaussian Naive Bayes
Logistic Regression
Stacking Classifier - Meta Regressor : Logistic Regression
Output negative/positive/neutral sentiment

Fig. 8. Working of Sentiment Classification Ensemble Stacking Generalizer

III. Results

A. Performance Metrics

The performance metrics consist of Accuracy (A), Precision (P), Recall (R), F1 Score (F1). These performance metrics are calculated on the basis of 4 parameters, True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). True Positive (TP) refers to cases where the model correctly predicts a positive sentiment, such as "I love this product!". True Negative (TN) is when the model correctly predicts a negative sentiment, like "This product is terrible." False Positive (FP) occurs when the model incorrectly predicts a positive sentiment when the actual sentiment is negative, like "I expected better quality." False Negative (FN) is when the model incorrectly predicts a negative sentiment when the actual sentiment is positive, like "It exceeded my expectations." Understanding these types of sentiments is crucial for effective sentiment analysis.

Accuracy is calculated as the ratio of correctly predicted instances to the total number of instances. Precision measures the proportion of correctly predicted positive instances among all instances predicted as positive. It focuses on the accuracy of positive predictions. Recall measures the proportion of correctly predicted positive instances out of all actual positive instances. It focuses on how well the model captures all positive instances. The F1 score is the harmonic mean of precision and recall. It provides a balance between precision and recall, making it a useful metric when both false positives and false negatives are important. A is accuracy, P is precision, R is recall and F1 is F1 score as shown in equations 11,12,13,14 -

$$A = \frac{tp + tn}{tp + tn + fp + fn} \tag{11}$$

$$P = \frac{tp}{tp + fp} \tag{12}$$

$$R = \frac{tp}{tp + fn} \tag{13}$$

$$F1 = \frac{2PR}{P + R} \tag{14}$$

Name of the model	Accuracy	Precision	Recall	F1 score
KNN	0.58701	0.58701	0.58701	0.58701
Random forest	0.94038	0.94038	0.94038	0.94038
SVM	0.99134	0.99134	0.99134	0.99134
Gaussian Naïve Bayes	0.80384	0.80384	0.80384	0.80384
Logistic Regression	0.97908	0.97908	0.97908	0.97908
Stacking Classifier	0.98317	0.98317	0.98317	0.98317

IV. Conclusion

Speech to Text conversion of the spoken vernacular and sentiment analysis of the converted text using various Stacking Classifiers. We present a comparison of success rates achieved with different stacking classifiers. Standard machine learning models like KNN, Random Forest, SVM, Gaussian Naïve Bayes, Logistic Regression produced an accuracy of 0.58701, 0.94038, 0.99134, 0.80384, 0.97908 but our novel stacking classifier model produced a better accuracy of 0.98317. We plan to expand this work in the future by comparing our result to existing transformers like BERT for sentiment analysis in real-time.

Acknowledgment

We would like to thank our professor Dr. Aparna Mohanty for her support, guidance, and valuable time.

REFERENCES

[1] C. M. Lee and S. Narayanan, "Toward detecting emotions in spoken dialogs," IEEE Transactions on Speech and Audio Processing, vol. 13, no. 2, pp. 293–303, Mar. 2005, doi: 10.1109/tsa.2004.838534.

[2] D. H. Wolpert, "Stacked generalization," Neural Networks, vol. 5, no. 2, pp. 241–259, Jan. 1992, doi: 10.1016/s0893-6080(05)80023-1.

[3] T. Kumar, M. Mahrishi, and S. Nawaz, "A review of Speech Sentiment Analysis using Machine Learning," in Lecture notes in networks and systems, 2022, pp. 21–28. doi: 10.1007/978-981-16-8826-3_3.

[4] F. Pedregosa et al., "SciKit-Learn: Machine Learning in Python," HAL (Le Centre Pour La Communication Scientifique Directe), Oct. 2011, [Online]. Available: https://hal.inria.fr/hal-00650905

[5] Z. Lu, L. Cao, Y. Zhang, C. Chiu, and J. Fan, "Speech Sentiment Analysis via Pre-trained Features from End-to-end ASR Models," arXiv (Cornell University), Nov. 2019, doi: 10.48550/arxiv.1911.09762.

[6] J. Chen, C. Sun, S. Zhang, and J. Zeng, "Cross-modal dynamic sentiment annotation for speech sentiment analysis," *Computers & Electrical Engineering*, vol. 106, p. 108598, Mar. 2023, doi: 10.1016/j.compeleceng.2023.108598.

[7] M. Syamala and N. Nalini, "A Speech-based Sentiment Analysis using Combined Deep Learning and Language Model on Real-Time Product Review," *International Journal of Engineering Trends and Technology*, vol. 69, no. 1, pp. 172–178, Jan. 2021, doi: 10.14445/22315381/ijett-v69i1p226.

[8] P. Hájek and M. Munk, "Speech emotion recognition and text sentiment analysis for financial distress prediction," *Neural Computing and Applications*, vol. 35, no. 29, pp. 21463–21477, Mar. 2023, doi: 10.1007/s00521-023-08470-8.

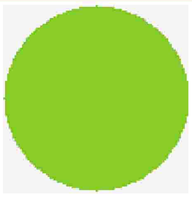
[9] R. K. Chaurasiya, N. S. Priya, P. Karanam, G. V. S. Kumar, M. Jahnavi, and T. P. V. K. Teja, "Sentiment Analysis from Speech Signals using Convolution Neural Network," *ICGSP '23: Proceedings of the 2023 7th International Conference on Graphics and Signal Processing*, Jun. 2023, doi: 10.1145/3606283.3606290.

[10] B. K. Norambuena, E. F. Lettura, and C. M. Villegas, "Sentiment analysis and opinion mining applied to scientific paper reviews," *Intelligent Data Analysis*, vol. 23, no. 1, pp. 191–214, Feb. 2019, doi: 10.3233/ida-173807.

Image not found or type unknown

PLAGIARISM SCAN REPORT

Date	April 21, 2024
Exclude URL:	NO



Unique Content	100
Plagiarized Content	0

Word Count	933
Records Found	0

CONTENT CHECKED FOR PLAGIARISM:

This study introduces an innovative method for speech sentiment analysis, employing a stacking classifier. By directly transcribing audio, the system extracts essential features for sentiment analysis. The stacking classifier, which combines multiple classifiers, enhances predictive performance. Intermediate models utilized include k-NN, Random Forest, SVM, Gaussian Naive Bayes, and Logistic Regression. OpenAI Whisper's transformer was used for audio-to-text conversion. The methodology encompasses audio data preprocessing, text conversion, natural language processing, and model training. Evaluation metrics, such as accuracy, precision, recall, and F1-score, are used to assess performance. Through empirical investigation, the study examines the effectiveness of various classifiers and feature sets in sentiment analysis of speech data. Experimental results demonstrate the system's ability to accurately classify sentiment, providing valuable insights into sentiment analysis methods tailored for spoken language. The first way this was achieved was by making using of the speech_recognition library in python in conjunction with the pydub library, the latter used for detecting silences within audio files.

The transcription function serves as the main procedure for converting audio to text. It begins by loading an audio file using AudioSegment from pydub, which represents the audio data. Then, the audio is split into chunks based on silence using the split_on_silence function, with parameters such as minimum silence length and silence threshold specified. These chunks are stored in a designated folder named "audio-chunks". Each segment undergoes individual processing within a loop. Sequentially, each chunk is exported as an autonomous WAV file within the specified directory. Employing the Recognizer object from speech_recognition, the script transcribes each chunk leveraging Google's speech recognition API (recognize_google). In the event of encountering any unrecognizable segments, the script gracefully handles such occurrences, emitting an error message as notification.

The transcribed text from each chunk is capitalized and appended to a cohesive string. Upon the completion of processing all segments, this string containing the transcriptions is returned. Additionally, the program facilitates saving the transcribed text to a CSV file, which can be made use of for further processing. The second method transcribes long audio recordings using the OpenAI Whisper model, and then saving the transcription into a CSV file.

It uses the Transformers library to implement a speech transcription pipeline. The pipeline, employs a pre-trained model for automatic speech recognition (ASR). The specific model used is the OpenAI Whisper model, tailored for ASR tasks. The process begins by checking for GPU availability to decide whether to utilize CUDA acceleration for computational efficiency. Once initialized, the pipeline is set up with the chosen ASR model and the appropriate device configuration. Initially, it loads the audio file specified by the user. The audio is then resampled to a standardized sample rate of 16,000 samples per second to ensure consistency. Subsequently, the resampled audio is passed through the ASR pipeline for transcription. To handle long audio files efficiently, the transcription process is chunked into smaller segments. This allows for more manageable processing, especially for lengthy recordings. Parameters can be passed to control the duration of each chunk and the overlap between consecutive chunks as well.

The obtained transcription is returned for further analysis or processing. Additionally, the transcription is saved to a CSV file as well. Stacked Generalization, is an ensemble learning technique that combines multiple classification or regression models via a meta-classifier or a meta-regressor. The basic idea is to train several base models, use their predictions as input features, and then use a meta-model to learn how to best combine the base models. Stacking classifier works by training the classifiers on the training data. These classifiers can be different algorithms or the same algorithm with different hyperparameters. First-level Predictions are made on the validation (or test) set using these base models. A new feature matrix is created by concatenating the predictions from the base models with the original features. Train a meta-classifier (or meta-regressor) on the new feature matrix. In this project we have used the logistic regression model as the eta classifier. Make predictions on the new data using the base models, then use the predictions as input features for the meta-model to get the final prediction.

The training dataset contains 20,800 entries of positive, negative and neutral sentiment texts. This dataset is divided into 80 percent as the training and 20 percent into the test dataset. Natural language processing is used to clear the text of the stopwords present in the English dictionary. The text column of the dataset is converted into numerical weights data array using a TF-IDF (Term Frequency-Inverse Document Frequency) vectorizer. The TF-IDF vectorizer carries out these calculations in equations 8, 9,10 and returns the TF-IDF weighted document-term matrix where t is a term, d is a document, D is a corpus, N is the total number of documents in the corpus and df(t,D) is the number of documents in the corpus that contain the term t–. The performance metrics consist of Accuracy (A), Precision (P), Recall (R), F1 Score (F1). These performance metrics are calculated on the basis of 4 parameters, True Positive (TP), True Negative (TN), False Positive (FP), False Negative (FN). True Positive (TP) refers to cases where the model correctly predicts a positive sentiment, such as "I love this product!. True Negative (TN) is when the model correctly predicts a negative sentiment, like "This product is terrible. False Positive (FP) occurs when the model incorrectly predicts a positive sentiment when the actual sentiment is negative, like "I expected better quality. False Negative (FN) is when the model incorrectly predicts a negative sentiment when the actual sentiment is positive, like "It exceeded my expectations. Understanding these types of sentiments is crucial for effective sentiment analysis.