Sentiment Analysis from Audio Recordings

Muhammad Hassan 23100199@lums.edu.pk

Abdullah Naveed 23100239@lums.edu.pk

Muhammad Ammar Ibrahim 23110345@lums.edu.pk

Abstract:

Audio sentiment analysis is a field of natural language processing which recognize and interprets the emotional tone in an audio. This paper proposes a comprehensive methodology for building a classifier to perform audio sentiment analysis using various natural language processing techniques and machine learning algorithms. The method used by this paper involves several steps to perform sentiment analysis on audio data, including noise removal, feature extraction, and classifier training. There are vast applications of sentiment analysis which ranges from monitoring customer satisfaction during phone calls and social media monitoring to tracking the trends of sentiments around specific topics. Additionally, audio sentiment analysis can be used to gauge audience reactions and adjust content accordingly. This study uses the Crema D dataset, which contains 7442 audio clips. These audio clips are labeled with different emotions and their levels.

1. Introduction:

Sentiment analysis is a field in natural language processing which aims to identify the emotional tone in a text or speech. However, most research in sentiment analysis has been focused on textual data, and less attention has been given to audio data. Our paper on Audio sentiment analysis fills this gap by recognizing and interpreting the emotional tone through audio data. This paper proposes a comprehensive methodology for building a classifier to perform audio sentiment analysis. Our proposed methodology involves several steps. First, we remove noise from the audio data. Next, we extract features from the audio data using various natural language processing techniques, such as Mel-Frequency Cepstral Coefficients (MFCCs), Fourier transform, Melspectrogram, Chromagram, etc. These extracted features are then used to train a classifier using various machine learning algorithms, such as Logistic Regression, Naïve Bayes, Support Vector Machines (SVMs), K nearest Neighbour, and Neural Networks, to perform sentiment analysis. In this study, we use the Crema D dataset, a diverse dataset consisting of 7442 audio clips from actors of various races and ethnicities. Each audio clip is labeled with six different emotions. The six different emotions are

- Anger
- Disgust
- Fear
- Happy
- Neutral
- Sad

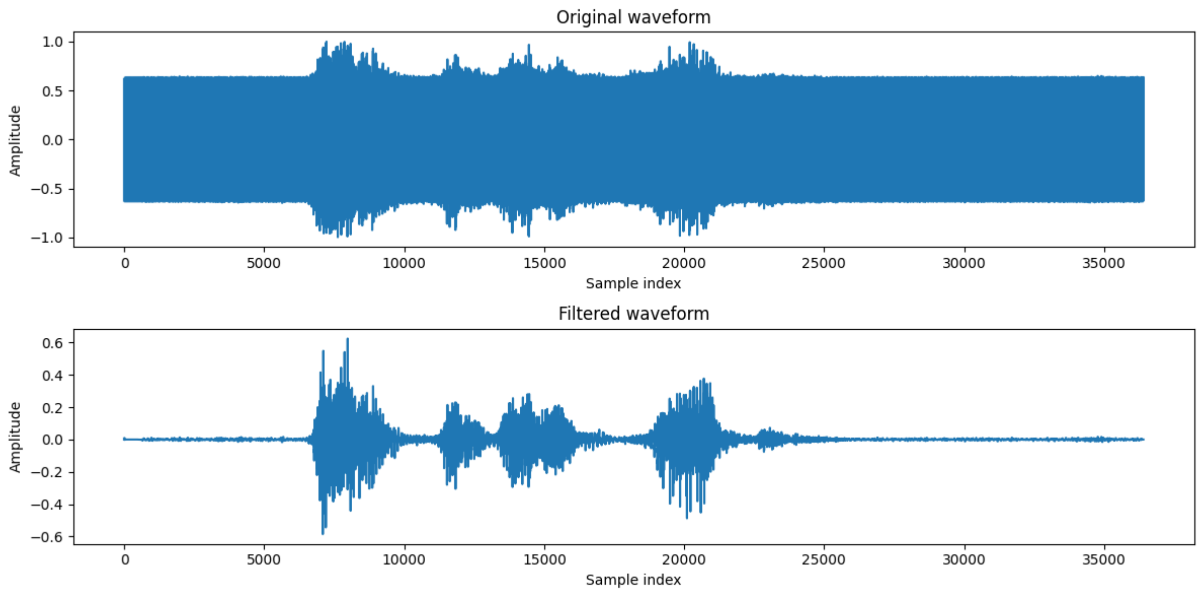
All these emotions come in four different levels: High, Medium, Low, and unspecified.

There are vast number of applications for audio sentiment analysis. One significant application is monitoring customer satisfaction during phone calls. Sentiment analysis can also be used for social media monitoring, where it can track the trends of sentiments around specific topics. Additionally, audio sentiment analysis can be used in the entertainment industry to gauge audience reactions and regulat the content accordingly to the audience reaction, leading to improved customer satisfaction which in turn results in higher revenue generate by the industry.

The paper is organzied as follows: Methodology, Mathematical Formulation, Identification and Extraction of Features, Results, and Conclusion.

2. Methodology:

To create a sentiment classifier for audio data, we will start by preprocessing the audio data to remove any unwanted or irrelevant information that might affect the sentiment classification accuracy. This may involve techniques such as noise reduction, removing silent segments, and normalizing the audio volume. These steps help create a clean and consistent audio dataset suitable for sentiment analysis. We used the Butterworth method to clean the noise from the dataset, which uses the Nyqist rate, the maximum frequency represented in the sampled signal without anti-aliasing. Secondly, we use the cutoff frequency, above which all noise should be removed. We have used a cutoff of 2000 and then normalized it by dividing it with the Nyqist rate. Then we used the filter method from Scipy to apply the filter on the signal, and as a result, we got a filtered signal with lower noise.



The figure above shows the data with noise and filtered data with the noise removed.

We also need to divide our dataset into two parts. We will use one part to train our models and the second to test our model's performance. We have used an 80-20 split where 80 percent will be used for training and 20 percent for testing.

Next, we need to develop a mathematical model that can take in the audio features extracted from the audio data and classify them into 6 emotions. In this paper, we have used multiple models. These models include KNN, Logistical regression, Naïve Bayes, SVM, and Neural, and their respective formulations can be found later in the paper.

After developing our model, we extract features from the audio data. It is necessary to get meaningful information from the raw audio signal, which we can pass into our models. Feature extraction is also necessary to reduce the dimensions of data. With the set of features relevant to the context of sentiment analysis, we can improve the accuracy of data. Feature extraction for audio data involves techniques such as Mel-frequency cepstral coefficients (MFCCs), Chromagram, Melspectrogram, and Fourier transform. These techniques help us capture the important audio characteristics that influence the sentiment of the audio data.

Once we have extracted the features, we pass them to our model to train our classifier. Training our classifier involves feeding it with labeled data, where the sentiment of the audio data is already known. The model uses this data to learn the patterns and relationships between the features and the sentiment labels.

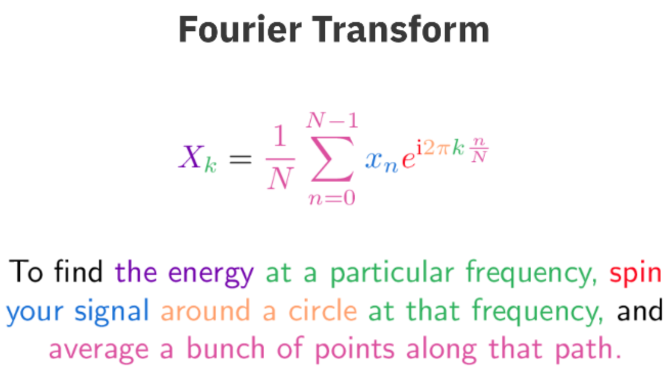
Finally, we must evaluate our model's performance using various metrics. The metrics we use are accuracy and F1 score. Accuracy here measures how well our model can classify the labels of these audios, whereas the F1 score is the harmonic mean of precision and recall. To understand the F1 score, we first need to understand precision and recall. Precision is the ratio of True positive overall predicted positive. Recall, on the other hand, is the ratio of true positives over actual positives. We use the F1 score to balance the trade-off between precision and recall. Using these metrics helps us understand the performance of our model.

3. Identification and Extraction of Features:

There are different methods for feature extraction from audio files. This section will discuss different features of an audio track and what features we will employ in our feature engineering.

3.1 Audio Features

- <u>**Fourier Transform:**</u> This technique transforms a signal's time domain into its corresponding frequency domain, revealing the signal's frequency components [1]. The magnitude of the Fourier transform indicates the frequency component that exists in the initial function. In contrast, the complex component reveals the phase deviation of the fundamental sinusoidal wave at that frequency. Fourier transform is a versatile and robust method that can handle noise and distortions in the dataset [4].

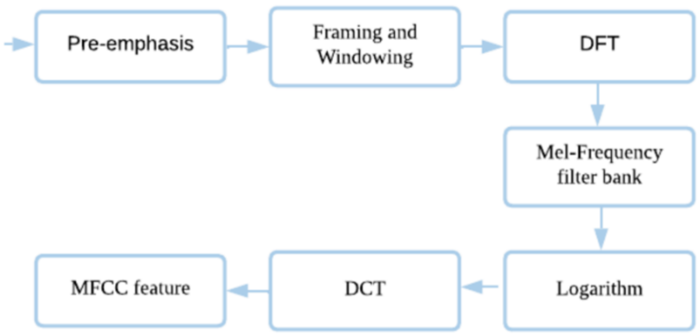


Fourier Transform

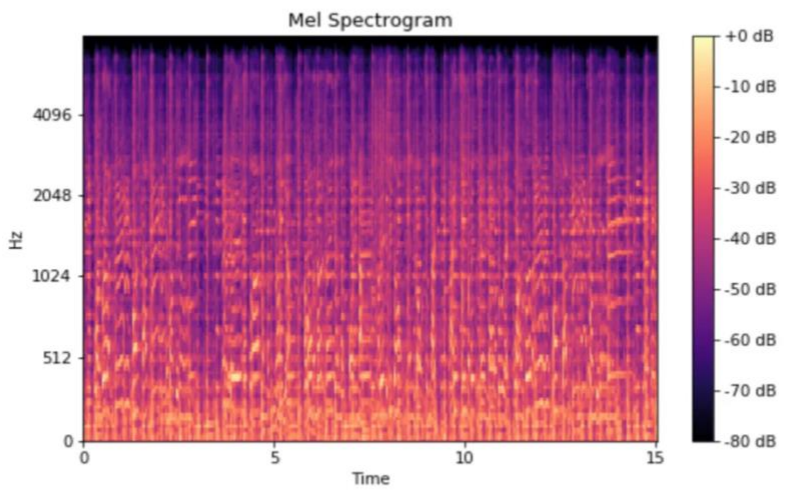
$$X_k = \frac{1}{N}\sum_{n=0}^{N-1} x_n e^{i2\pi k \frac{n}{N}}$$

To find the energy at a particular frequency, spin your signal around a circle at that frequency, and average a bunch of points along that path.

- **Mel-Frequency Cepstral Coefficients (MFCC):** A widely used feature in voice signal processing, MFCC finds applications in various domains, including but not limited to speaker recognition, voice recognition, and gender identification. To compute MFCC, a sequence of five procedures needs to be followed, which involves framing the signal, computing the power spectrum, applying a Mel filter bank on the resulting power spectra, computing the logarithmic values of all the filter banks, and ultimately applying the discrete cosine transform (DCT) [5]. This technique represents the power spectrum of an audio signal as a sum of sinusoids over short-term intervals, generating 39 features for each audio file. The figure below illustrates the process of calculating MFCC.

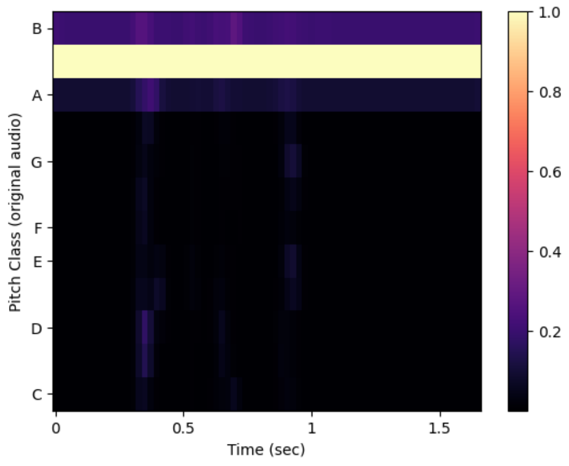


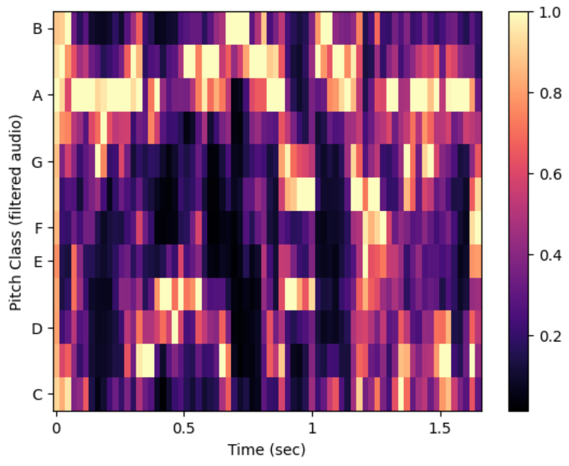
- **Mel-Spectrogram:** Humans are more proficient in distinguishing disparities in lower-frequency sounds compared to higher-frequency sounds. A clear example is our ability to easily differentiate between 500 and 1000 Hz frequencies. However, despite the identical distance between both pairs, we may struggle to detect the difference between 10,000 and 10,500 Hz. In 1937, Stevens, Volkmann, and Newman introduced the concept of the **Mel scale**, which defines a unit of pitch such that equal pitch distances sound equally distant to the human listener [6]. The Mel-spectrogram approach converts the frequencies of the signal into the Mel scale, which is a logarithmic transformation of the signal's frequency. This technique is better suited for applications where we need to mimic the human perception of the audio.



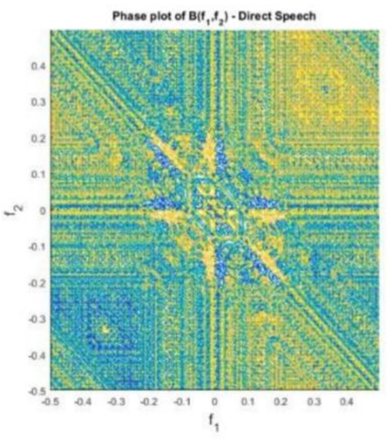
Here is a figure of the Mel-Spectrogram of a soundtrack; on the left side, we have frequencies in the hertz, and on the right side, we have different Mel scale classes with colors.

- **Chromagram:** Chromatography involves the segregation of various constituents present in a mixture. The Chromagram is a representation of an audio signal where the entire spectrum is projected onto pitch classes. Pitch class profiles are valuable instruments for examining audio files. The chromagram denotes the arrangement of pitches within an audio file, making it simpler to comprehend how pitches are classified in the audio. Pitches represent the characteristic of any sound or signal that enables the arrangement of files according to their frequency-based scale. It is a form of assessment of sound quality that assists in determining whether the sound is higher, lower, or medium-pitched [3]. This technique is robust to acoustic dynamics and enables us to evaluate the sound signal perceptually without resorting to physical frequencies [2].



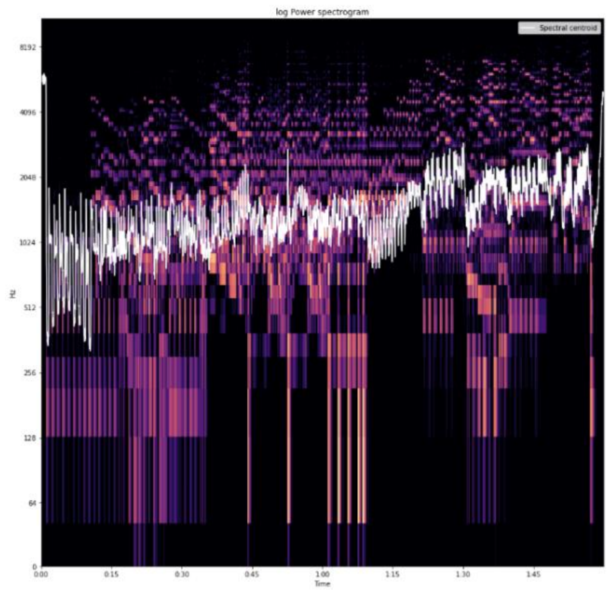


- **Bi-coherence:** is a statistical measure that involves the mean, variance, skewness, and kurtosis to analyze the non-linear interactions between frequency components. It captures non-linearities that are often overlooked by traditional Fourier-based methods. The bicoherence is a metric that determines the fraction of energy in a signal at a specific bi-frequency that is quadratically phase-locked. Typically, this metric is normalized within a range comparable to the classical coherence and correlation coefficient [7].

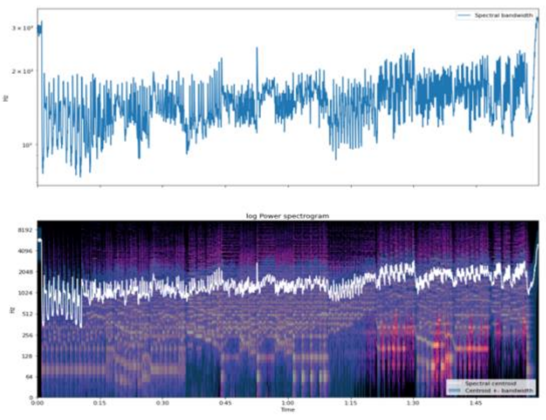


The figure above displays the magnitude plot of bicoherence for the direct recording.

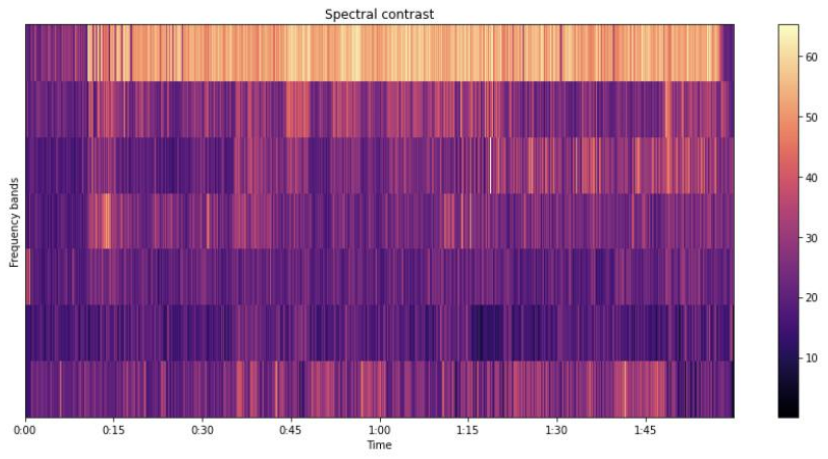
- **Spectral Centroid:** The spectral centroid is a metric that indicates the center of mass of a spectrum It is an important measurement for characterizing the spectrum of digital audio signals [3]. The spectral centroid is a measurement commonly used to evaluate the brightness of a sound. This measurement is derived by computing the "center of gravity" of the frequency and magnitude data of the Fourier transform [9]. The white line in the figure below represents every spectrum's centroid and amplitudes.



- **Spectral Bandwidth:** Bandwidth refers to the variation between the upper and lower frequencies within a continuous frequency band. Since signals oscillate around a point, we can determine the bandwidth of the signal at a specific timeframe by adding the maximum deviations of the signal from both sides of the point of oscillation, which is the signal's centroid [3].



- **<u>Spectral contrast:</u>** It is a metric that measures the frequency energy at each timestamp. Measuring the level of energy in audio files is challenging since most of them contain frequencies whose energy changes with time. Spectral contrast is a method used to quantify this energy variation [3].



The image shown above illustrates the spectral contrast of an audio file. Typically, high contrast values correspond to clear signals with narrow bands, whereas low contrast values correspond to broad-banded noise [3].

3.2 Extracted Features and Reasoning

The paper outlines seven techniques that were used for feature extraction. Firstly, the MFCC technique was utilized to extract 20 features from the audio data. Then, we used choromagram, melspectogram, spectral centroid, spectral bandwidth, spectral rollff and spectral flatness to extract further features from the audios. A feature vector was created by combining all the features extracted using these techniques. These techniques are commonly used in audio processing, robust and efficient and provide us with a variety of features.

The MFCC technique is particularly useful in extracting different features of audio and its acoustic properties. Fourier transform helped in extracting the different frequency components from the dataset. The Mel spectrogram enabled the extraction of power density on the Mel scale, and the spectral centroid enabled us to extract the spectral shape of the audio signal. We used chromagram as it robust to acoustic dynamics and enables us to evaluate the sound signal perceptually without resorting to physical frequencies. Spectral bandwidth helps us determine the bandwidth of the audio signal at a certain time frame. Using these multiple techniques, we were able to extract valuable information from the dataset, resulting in high-accuracy performance. Combining the features extracted using these techniques allows the model to better understand the audio data's characteristics and make more accurate predictions.

4. Mathematical Formulation:

In this section, we will delve into the mathematical formulations of the various machine learning models we have employed for audio classification in this paper. Specifically, we have used five different models: K-nearest neighbor, Logistic Regression, Naive Bayes, Support Vector Machine (SVM), and a neural network. K-nearest neighbor is a supervised learning classifier that uses a distance metric to classify an audio sample. Logistic Regression estimates the probability of a class occurring given the input features. Naive Bayes is a probabilistic classifier that assumes independence between features. SVM identifies an optimal boundary separating data points into different classes. Lastly, the neural network is a complex model that learns to distinguish between different sounds inspired by the functioning of the human brain. By utilizing a combination of these models and carefully selecting appropriate features, we can achieve accurate and robust classification performance for audio-based applications. The problem under consideration is a multi-class classification problem.

4.1 K Nearest Neighbor

KNN is an instance-based learning algorithm that can easily adapt to unseen data. KNN uses a distance matrix to calculate the distance of the test point with all its neighbors and assign the label of the most frequent class in the neighbor. The prediction's complexity (time and storage) increases with the training data's size. K is a hyperparameter, and we learn K using data (minimizing the validation loss).

In KNN, we assume that we have training data D given by:

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

$$\mathcal{Y} = \{1, 2, \ldots, M\} \text{ (M-class classification)}$$

KNN can also be used for regression. The KNN model's complexity increases with the training data's size. Examples of some of the commonly used distance matrices:

$$\text{Euclidean} \qquad \text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_2 = \sqrt{\sum_{i=1}^{d} (x_i - x_i')^2}$$

$$\text{Manhattan} \qquad \text{dist}(\mathbf{x}, \mathbf{x}') = \|\mathbf{x} - \mathbf{x}'\|_1 = \sum_{i=1}^{d} |x_i - x_i'|$$

For categorical variables, Hamming distance is used:

$$\text{dist}(\mathbf{x}, \mathbf{x}') = \sum_{i=1}^{d} 1 - \delta_{x_i - x_i'}$$

Let D(i, j) denote the distance between the ith and jth recordings in X_{train}, and let $Y_{\text{train(i)}}$ denote the class label of the ith recording in X_{train}.

Calculate the distance of an audio point with all points in the X_{train}.

Sort D(i, j) and use the mod(kth nearest neighbor) to classify the point.

N_k = {i: d_i is among the K smallest distances}

Where k is the user-defined parameter, the KNN algorithm makes predictions for a given test point by comparing it to nearby data points that are similar to the test point. Since there may not be any neighbors in high-dimensional space, the KNN algorithm is susceptible and responsive to the effects of the Curse of Dimensionality.

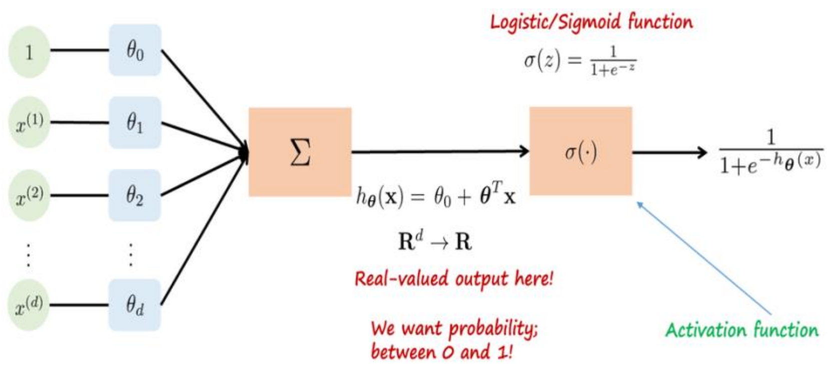
4.2 Logistic Regression

Logistic regression is a discriminative classifier that estimates the probability of a class-given data. Rather than simply predicting a class, we calculate the likelihood of an instance belonging to that class. Here we again assume we have d-dimensional training data:

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

$$\mathcal{Y} = \{1, 2, \ldots, M\} \text{ (M-class classification)}$$

For binary class classification, we have a simple model:



Logistic/Sigmoid function
$\sigma(z) = \frac{1}{1+e^{-z}}$
1
$x^{(1)}$
$x^{(2)}$
$x^{(d)}$
θ_0
θ_1
θ_2
θ_d
Σ
$\sigma(\cdot)$
$\frac{1}{1+e^{-h_{\boldsymbol{\theta}}(x)}}$
$h_{\boldsymbol{\theta}}(\mathbf{x}) = \theta_0 + \boldsymbol{\theta}^T\mathbf{x}$
$\mathbf{R}^d \to \mathbf{R}$
Real-valued output here!
We want probability; between 0 and 1!
Activation function

Posterior Probability is given by:

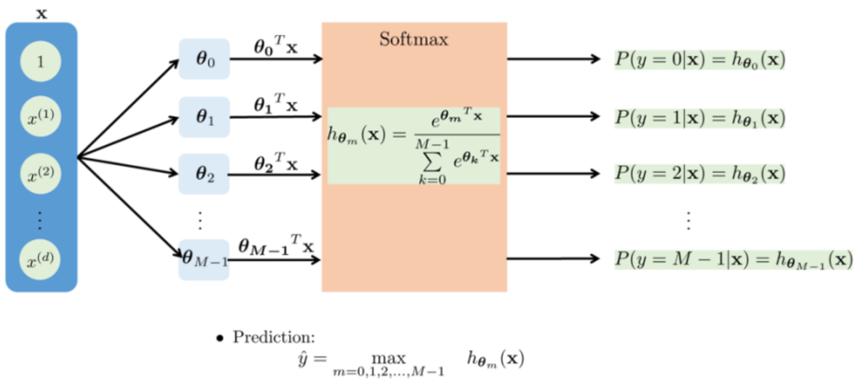
$$P(y = 1|\mathbf{x}) = \frac{e^{\boldsymbol{\theta}^T\mathbf{x}}}{e^{\boldsymbol{\theta}^T\mathbf{x}} + e^0}$$

$$P(y = 0|\mathbf{x}) = \frac{e^0}{e^{\boldsymbol{\theta}^T\mathbf{x}} + e^0}$$

Here we are using the sigmoid function to get the probability between 0 and 1. Usually, a threshold of 0.5 is used to assign the class label, but it can adjust based on our model and data. Since the problem at hand is of multiclass classification, we have 2 options:

1. Build a one-vs-all (OvA) one-vs-rest (OvR) classifier
2. Build an all-vs-all classifier

We extend the definition of binary classification of logistic regression for multi-class classification (as we have 6 classes). Now instead of using sigmoid, we use Softmax. Here our model is the weighted sum of features followed by Softmax.



$\mathbf{x}$
1
$x^{(1)}$
$x^{(2)}$
$x^{(d)}$
$\boldsymbol{\theta}_0$
$\boldsymbol{\theta}_1$
$\boldsymbol{\theta}_2$
$\boldsymbol{\theta}_{M-1}$
${\boldsymbol{\theta}_0}^T\mathbf{x}$
${\boldsymbol{\theta}_1}^T\mathbf{x}$
${\boldsymbol{\theta}_2}^T\mathbf{x}$
${\boldsymbol{\theta}_{M-1}}^T\mathbf{x}$
Softmax
$h_{\boldsymbol{\theta}_m}(\mathbf{x}) = \frac{e^{{\boldsymbol{\theta}_m}^T\mathbf{x}}}{\sum_{k=0}^{M-1} e^{{\boldsymbol{\theta}_k}^T\mathbf{x}}}$
$P(y=0|\mathbf{x}) = h_{\boldsymbol{\theta}_0}(\mathbf{x})$
$P(y=1|\mathbf{x}) = h_{\boldsymbol{\theta}_1}(\mathbf{x})$
$P(y=2|\mathbf{x}) = h_{\boldsymbol{\theta}_2}(\mathbf{x})$
$P(y=M-1|\mathbf{x}) = h_{\boldsymbol{\theta}_{M-1}}(\mathbf{x})$
• Prediction:
$\hat{y} = \max_{m=0,1,2,\ldots,M-1} h_{\boldsymbol{\theta}_m}(\mathbf{x})$

Posterior Probability in case of multi-class classification:

$$P(y=m|\mathbf{x}) = h_{\boldsymbol{\theta}_m}(\mathbf{x}) = \frac{e^{{\boldsymbol{\theta}_m}^T\mathbf{x}}}{\sum_{k=0}^{M-1} e^{{\boldsymbol{\theta}_k}^T\mathbf{x}}}$$

Learning the weights of our model using Gradient Descent. For each iteration, we update the weights using the following:

$$\theta_j \leftarrow \theta_j - \alpha \frac{\partial \mathcal{L}}{\partial \theta_j}, \quad \alpha > 0$$

Here alpha is the learning or step size, and dL/d(theta) is the change in loss function with respect to weight.

For the binary case, the loss function we have:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n} y_i \log(h_{\boldsymbol{\theta}}(\mathbf{x}_i)) + (1-y_i)\log(1-h_{\boldsymbol{\theta}}(\mathbf{x}_i))$$

Here y_i is the true label, and $h_{\text{theat}}(x_i)$ is the predicted label. For multi-class classification:

$$\mathcal{L}(\boldsymbol{\theta}) = -\sum_{i=1}^{n}\sum_{m=0}^{M-1} \delta(y_i - m)\ \log\left(\frac{e^{{\boldsymbol{\theta}_m}^T\mathbf{x}_i}}{\sum_{k=0}^{M-1} e^{{\boldsymbol{\theta}_k}^T\mathbf{x}_i}}\right)$$

4.3 Naive Bayes

Naïve Bayes is a generative classifier. It attempts to develop a generative statistical model that enables us to understand how a specific class generates input data. Generative classifiers determine the most probable class that would have generated a given observation for a test point. This involves calculating the conditional probability P(y|x) for input x and selecting the label y with the highest probability. Naive Bayes follows the assumption that the features in the data are independent, and we estimate the P(y|x) from the data using the Bayes theorem.

We assume we have d-dimensional training data:

$$D = \{(\mathbf{x_1}, y_1), (\mathbf{x_2}, y_2), \ldots, (\mathbf{x_n}, y_n)\} \subseteq \mathcal{X}^d \times \mathcal{Y}$$

$$\mathcal{Y} = \{1, 2, \ldots, M\} \text{ (M-class classification)}$$

Our dataset has 6 different class labels. We formulate our hypothesis function as

$$\begin{aligned} h_{\mathrm{NB}}(\mathbf{x}) &= \underset{y \in \mathcal{Y}}{\text{maximize}} \quad \sum_{i=1}^{d} \log \left(P(x^{(i)} \mid y)\, P(y) \right) \\ &= \underset{y \in \mathcal{Y}}{\text{maximize}} \quad \sum_{i=1}^{d} \log P(x^{(i)} \mid y) \;+\; \log P(y) \end{aligned}$$

Computing P(y) is straightforward if y takes a discrete value, as in our case. We compute probabilities during the training stage. For continuous features can use Gaussian distribution to model the probability density function.

$$p(x^{(i)} \mid y = k) \;=\; \frac{1}{\sigma\sqrt{2\pi}} \exp\left(-\frac{(x^{(i)} - \mu)^2}{2\sigma^2}\right)$$

Given data, we learn hyperparameters for each i and k

$$\mu = \frac{1}{\mathrm{count}(y = k)} \sum_{j=1}^{n} \delta(y_j - k)\; x_j^{(i)}$$

$$\sigma^2 = \frac{1}{\mathrm{count}(y = k)} \sum_{j=1}^{n} \delta(y_j - k)\; \left(x_j^{(i)} - \mu\right)^2$$

Naïve Bayes is a fast and easy-to-implement algorithm that can handle missing values, outliers, and irrelevant features. It performs well with textual data and converges quickly but is not robust to redundant features and has high bias with no regularization available.

4.4 Support Vector Machine

The support vector machine (SVM) aims to find a hyperplane boundary that separates the classes with maximum margin. The margin refers to the distance between the decision boundary and the closest data points. The ideal boundary is one that maximizes the margin or the distance between the boundary and the "difficult points" near the decision boundary. To achieve this, we aim to choose a roughly equidistant hyper-plane from the nearest positive and negative data points. The classification margin is twice the width of the distance between the boundary and these nearest points. The Support Vector Machine (SVM) classifier is a popular algorithm that identifies the hyper-plane that maximizes this margin.

Soft SVM:

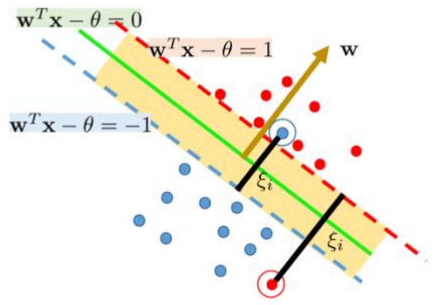
$$\begin{aligned} \underset{\mathbf{w},\theta}{\text{minimize}} \quad & \|\mathbf{w}\|^2 + C\,(\sum_{i=1}^{n} \xi_i) & \\ \text{subject to} \quad & y_i(\mathbf{w}^T\mathbf{x}_i - \theta) \geq (1 - \xi_i) & i = 1, 2, \ldots, n \\ & \xi_i \geq 0 & i = 1, 2, \ldots, n \end{aligned}$$

We dislike the points to be misclassified.

We allow the points to be misclassified.

$$\|\mathbf{w}\|^2 + C\,(\sum_{i=1}^{n} \xi_i) \quad \rightarrow \quad \|\mathbf{w}\|^2 + C \sum_{i=1}^{n} \max\left(0, 1 - y_i(\mathbf{w}^T\mathbf{x}_i - \theta)\right)$$

We introduce a slack variable to allow miss-classification. There is a penalty in the objective function for Slack. We have incorporated the distance in the objective function. The points far from the boundary (wrong side) are penalized more. The trade-off between maximizing the margin and fit of training data is maintained by the parameter C. We can use gradient descent to find the optimal W vector and theta value.



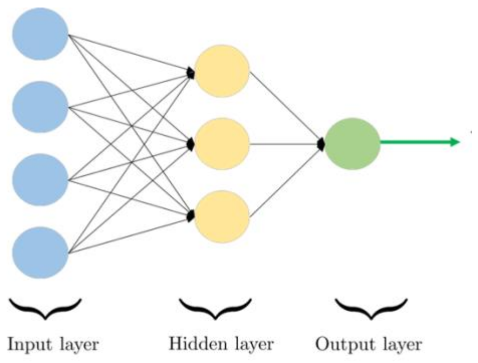
$\mathbf{w}^T\mathbf{x} - \theta = 0$
$\mathbf{w}^T\mathbf{x} - \theta = 1$
$\mathbf{w}$
$\mathbf{w}^T\mathbf{x} - \theta = -1$
ξ_i
ξ_i

In this, w refers to the weight vector, b is the bias term, x is the feature vector, and ξi is the slack variable. Once we have found the optimal values for w and b, the decision will be

$$W^Tx \text{ - theta} = 0$$

4.5 Neural network

A neural network is a set of neurons organized in layers. By taking the input and parameters of the neurons into account, we can use a process known as a forward pass to determine the output. This involves traversing the layers from input to output in the neural network.



Forward Propagation:

During forward propagation, we compute the output of each layer in the neural network using the following equations:

For a layer l: 1, 2, ..., L-1:

$$z^1 = w^l a^0 + b^l$$

$$a^l = f(z^l)$$

where w^l is the weight matrix for layer l, b^l is the bias vector for layer l, a^0 is the activation vector for layer 0 (input layer), z^1 is the pre-activation output to layer l, and f is the activation function. In general

For layer L (the output layer):

$$z^L = w^L a^{(L-1)} + b^L$$

$$a^L = f(z^L)$$

Backward Propagation and Weight Update:

Using the gradient descent to update the weights as:

$$\text{Gradient descent:} \quad w_{i,j}^{[\ell]} = w_{i,j}^{[\ell]} - \alpha \frac{\partial \mathcal{L}}{\partial w_{i,j}^{[\ell]}}$$

To compute the derivative, we utilize backpropagation to propagate the total loss at the output node back through the neural network. This allows us to determine the contribution of each node to the overall loss.

$$\frac{\partial \mathcal{L}}{\partial \mathbf{W}^{[\ell]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[\ell]}} \mathbf{a}^{[\ell-1]T} \qquad \frac{\partial \mathcal{L}}{\partial \mathbf{b}^{[\ell]}} = \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[\ell]}}$$

$$\frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[\ell]}} = \left(\mathbf{W}^{[\ell+1]T} \frac{\partial \mathcal{L}}{\partial \mathbf{z}^{[\ell+1]}} \right) \odot g'(\mathbf{z}^{[\ell]})$$

Here z^l is the pre-activation of layer l, and a^{l-1} is the output of the previous layer

Once we have computed the gradients, we can update the weights and biases of the network using the following equations:

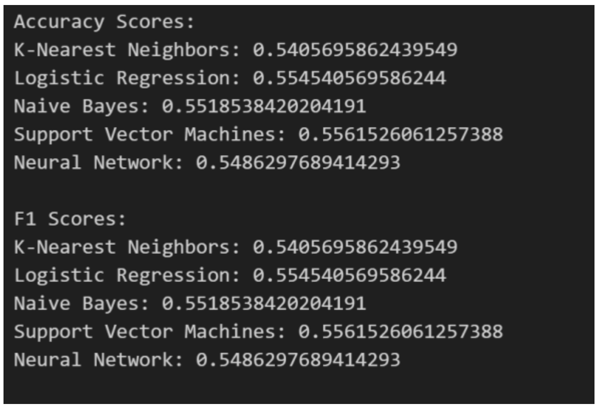
For layer l =1,2,...,L-1:

$$w^l = w^l - \alpha \, \delta^l \, (a^{l-1})^T$$

$$b^l = b^l - \alpha \, \delta^l$$

where α is the learning rate, δ^l is the gradient of the cost function with respect to the input to layer l, and $(a^{l-1})$is the transpose of the activation vector for layer l.

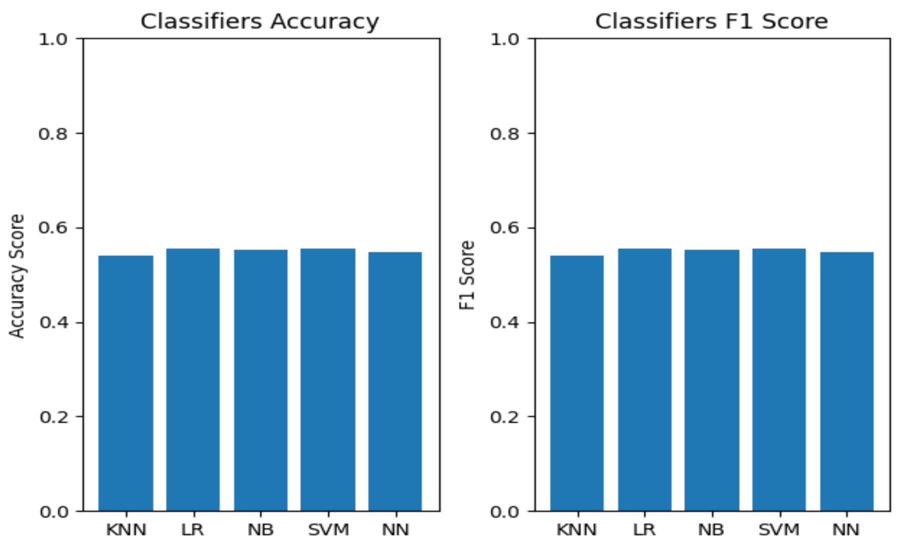
5. Finding and Results



```
Accuracy Scores:
K-Nearest Neighbors: 0.5405695862439549
Logistic Regression: 0.554540569586244
Naive Bayes: 0.5518538420204191
Support Vector Machines: 0.5561526061257388
Neural Network: 0.5486297689414293

F1 Scores:
K-Nearest Neighbors: 0.5405695862439549
Logistic Regression: 0.554540569586244
Naive Bayes: 0.5518538420204191
Support Vector Machines: 0.5561526061257388
Neural Network: 0.5486297689414293
```

From our trained classifiers, we observe the accuracy and F1 scores of the classifiers to evaluate their performance. The KNN model gives us an accuracy of 0.541 and an F1 score of 0.541. Next, we evaluate the naive Bayes model, which gives us an accuracy of 0.552 and an F1 score of 0.552. The logistic regression classifier gives us an accuracy of 0.554 combined with an F1 score of 0.554. Next, we have the SVM classifier, which has an accuracy of 0.556 and an F1 score of 0.556. Lastly, the neural net model gives us an accuracy of 0.549 and an F1 score of 0.549. We observe that the SVM model performs the best out of all these models, whereas the K-nearest neighbor comes in at last. Our best choice for the classifier for audio sentiment detection would be using the SVM model owing to its highest accuracy and F1 score.



6. Conclusion

To summarize, this paper presents a sentiment analysis approach to audio data. The first step in our methodology was to preprocess the data by removing any noise using the Buttercup method to improve the training and testing data. The preprocessed data was then passed through various models, which included K nearest neighbor, Naive Bayes, SVM, Logistic regression, and Neural Network to analyze the sentiment. Among the models that were tested, the SVM (Support Vector Machine) classifier was observed to outperform all others, achieving an accuracy of 56 percent. Although this accuracy seems low, it is important to note that sentiment analysis on audio data is a challenging task due to different variations in speech patterns and different emotions. Furthermore, the dataset came from different ethnicities and backgrounds, adding complexity to the dataset. Moreover, we can improve the accuracy by providing more training data to the model. Our finding is that the SVM is the best performer for this and can be used in real-world applications of sentiment analysis.

Reference list

[1] Abdusalomov, A.B., Safarov, F., Rakhimov, M., Turaev, B. and Whangbo, T.K., 2022. Improved Feature Parameter Extraction from Speech Signals Using Machine Learning Algorithm. *Sensors*, *22*(21), p.8122.

[2] Qu, Y., Li, X., Qin, Z. and Lu, Q., 2022. Acoustic scene classification based on three-dimensional multi-channel feature-correlated deep learning networks. *Scientific Reports*, *12*(1), p.13730.

[3] Verma, Y. (2021). *A Tutorial on Spectral Feature Extraction for Audio Analytics*. [online] Analytics India Magazine. Available at: https://analyticsindiamag.com/a-tutorial-on-spectral-feature-extraction-for-audio-analytics/.

[4] Rouse, M. (2020). *What is the Fourier Transform? - Definition from Techopedia*. [online] Techopedia.com. Available at: https://www.techopedia.com/definition/7292/fourier-transform#:~:text=The%20Fourier%20transform%20is%20a%20mathematical%20function%20that%20decomposes%20a.

[5] Abdul, Z.K. and Al-Talabani, A.K., 2022. Mel Frequency Cepstral Coefficient and its applications: A Review. *IEEE Access*.

[6] Roberts, L. (2020). *Understanding the Mel Spectrogram*. [online] Medium. Available at: https://medium.com/analytics-vidhya/understanding-the-mel-spectrogram-fca2afa2ce53.

[7] Wikipedia. (2022). *Bicoherence*. [online] Available at: https://en.wikipedia.org/wiki/Bicoherence#:~:text=The%20bicoherence%20measures%20the%20proportion [Accessed 30 Apr. 2023].