MACHINE LEARNING AND DEEP LEARNING BASED AUDIO CLASSIFICATION

A Project Report

Submitted by

Prathamesh Kulkarni	**111707027**
Sarthak Umarani	**111707058**
Vaishnavi Diwan	**111707060**
Vishakha Korde	**111707061**

Submitted in fulfilment of the requirements of the degree of

B.Tech. Electronics and Telecommunication

Under the guidance of

Dr. Priti Rege

DEPARTMENT OF ELECTRONICS AND

TELECOMMUNICATION ENGINEERING

COLLEGE OF ENGINEERING PUNE

2020-21

CERTIFICATE

This is to certify that the report entitled ‘**Machine Learning and Deep Learning based Audio Classification’** submitted by Prathamesh Kulkarni (111707027), Sarthak Umarani (111707058), Vaishnavi Diwan (111707060) and Vishakha Korde (111707061) in the fulfilment of the requirement for the award of degree of Bachelor of Technology (Electronics and Telecommunication Engineering) of College of Engineering Pune, affiliated to the Savitribai Phule Pune University, is a record of their own work.

Dr. Priti Rege
Project Guide
Department of
Electronics and
Telecommunication
College of Engineering, Pune

Dr. S. P. Mahajan
Head of the Department
Department of
Electronics and
Telecommunication
College of Engineering, Pune

Date:

Place:

Report Approval

This report entitled

MACHINE LEARNING AND DEEP LEARNING BASED AUDIO CLASSIFICATION

By

Prathamesh Kulkarni	**111707027**
Sarthak Umarani	**111707058**
Vaishnavi Diwan	**111707060**
Vishakha Korde	**111707061**

is approved for the degree of

Bachelor of Technology with

Electronics and Telecommunication Engineering

of

Electronics and Telecommunication Department

College of Engineering, Pune

(An Autonomous Institute of Govt. of Maharashtra)

Examiners	Name	Signature
1. External Examiner	____________________	___________
2. Guide/Supervisor	____________________	___________

Abstract

We are surrounded by audio signals. As a result, audio classification is gaining popularity in a variety of applications, ranging from fire alarm identification for hearing impaired people to engine sound analysis for maintenance purposes to baby monitoring. Machine Learning and Deep Learning algorithms have been explored widely for such purposes.

Usually, building machine learning models to identify, explain, or generate audio involves modelling tasks with audio samples as input data. These audio samples are normally interpreted as a time series, with the amplitude of the waveform as the y-axis measurement. The amplitude is normally calculated as a function of the pressure shift around the microphone or receiver unit that picked up the audio in the first place. These time series signals will also be your only input data unless your audio samples have metadata associated with them.

CNN is a form of feed-forward Artificial Neural Network that is commonly used to recognize images quickly and accurately. In the area of audio recognition and classification, this capability is turns out to be crucial. Although the computational complexity remains high, it yields significantly better results as compared to the Machine Learning models.

One of the most useful features for extracting information from audio waveforms (and digital signals in general) has been around since the 1980s and is still cutting-edge: Davis and Mermelstein introduced Mel Frequency Cepstral Coefficients (MFCCs) in 1980.

This report presents a study on the classification of audio signals and is subsequently performed for two cases – child cries and bird songs.

In child cry classification, different spectral and descriptive features like loudness, ZCR, spectral flatness etc. are used along with MFCC. First, the supervised ML models were trained by the individual features and then according to the results, selected features were combined for the combined training.

In case of bird song classification, for the training on supervised ML models, same features along with MFCC and methods as that in child cry classification are followed. Mel Spectrogram is used to train the Convolutional Neural Network (CNN). Performance of unsupervised models is also studied in case of bird classification.

Different models are used for classification in both the cases. Logistic regression, SVM, KNN and Random Forest are the supervised ML models that are used for classification. In Convolutional Neural Network (CNN), EfficientNet is used as the supervised deep learning model. Certain unsupervised learning models like K-means and GMM are also used in case of bird classification.

To optimize the model performance, Taguchi method is used to make design of experiments and select the best value of the parameters.

Multiple datasets are used in this project and their performance evaluation and analysis on different supervised and unsupervised models is done.

Keywords — MFCC, GFCC, KNN, SVM, Random Forest, feature extraction, spectrogram, K-Means, GMM, CNN, Taguchi method, EfficientNet

Declaration

We declare that this written submission represents our ideas in our own words and where others' ideas or words have been included, we have adequately cited and referenced the original sources. We also declare that we have adhered to all principles of academic honesty and integrity and have not misrepresented or fabricated or falsified any idea/data/fact/source in our submission. We understand that any violation of the above will be cause for disciplinary action by the Institute and can also evoke penal action from the sources which have thus not been properly cited or from whom proper permission has not been taken when needed.

Name	MIS No.	Signature
Prathamesh Kulkarni	111707027	_____________
Sarthak Umarani	111707058	_____________
Vaishnavi Diwan	111707060	_____________
Vishakha Korde	111707061	_____________

Date:

Place:

Acknowledgements

We are grateful to our college and to the Department of Electronics and Telecommunication for providing us this opportunity to work on this project. It is their visionary objective to encourage students for this thesis-based project that has blossomed into this extraordinary opportunity.

To begin with, we sincerely thank our project guide Prof. Dr. Priti Rege for her guidance towards the design and development of the project. Her constant motivation, support in bolstering requisite theoretical concepts along with invaluable tips and tricks proved to be crucial. Special thanks to our faculty advisor Mrs. Y. Vaidya for her support. We are also indebted to Dr. Abhishek Bhatt and Ms. Nilima Kolhare for evaluating our work and suggesting invaluable improvements.

The team is immensely grateful to Head of the Department Dr. S. P. Mahajan. It is his support and encouragement to students to work on a project of their field of interest that has made this possible.

Contents

1. Introduction

1.1 Objectives

a. To tackle a real-life problem through including data research and analysis, data preparation, creation of models, analysis of results (or model improvement) and the final presentation.
b. The main objective is to train a model to learn the characterizing features of audio signals (infant cries, bird sounds etc.) and effectively classify them into the specific categories.
c. The categories include hungry, belly pain, discomfort, burping and tiredness in case of infant cries and the species of birds in case of bird sounds.

Fig. 1.1: Reasons for crying babies

d. One of the objectives also comprises of the collection and comparison of reliable and substantial datasets for carrying out the model's training.
e. The approach also aims at developing an in depth understanding of the distinguishing features which play a pivotal role in classifying non - speech audio signals.

1.2 What is Audio Classification?

The method of listening to and analyzing audio recordings is known as audio classification. This method, also known as sound classification, is at the heart of modern AI technology such as virtual assistants, automatic speech recognition, and text-to-speech applications.

Information is needed for machines to learn how to hear and what to listen for. They improve the ability to distinguish between sounds using this information in order to complete specific tasks. Using dedicated audio classifiers, the annotation process often includes classifying audio files based on project-specific needs.

Types of audio classification -

1) Environmental Sound Classification: This is the classification of sounds heard in various settings, as the name suggests.

2) Music classification: It is the method of categorizing music according to factors such as genre or instruments used.

3) Natural Language Utterance Classification: This is a method of categorizing natural language recordings based on the language spoken, dialect, grammar, or other features. To put it another way, classification of human voice.

1.3 Introduction to Case Study 1 – Child Cry Classification

Infant mortality rate is inferred as child death rate before completion of the one year from birth. The datum released by World Health Organization (WHO) infers that the infant mortality rate is million. The main reason behind this is due to health issues. Nearly 75% of infant deaths shall be avoided if the disease is predicted at an earlier stage. Therefore, to rectify this issue, designed a child cry classification system to classify infant cries is in need. From the cry signals of the baby, it is possible to classify the need of the baby by extracting specific features from that sound signal.

Communication is a key aspect for humans, as they are social animals. The human body is a complex machine. It communicates internally via electrical signals. This enables a person to articulate their thoughts verbally. Most humans can speak and voice their concerns, except babies. We aim to bridge the gap in communication caused by this.

Adult voice and the sound of crying babies have similarities and differences, previous researches were found differences in the character of the sound in the aspect of the fundamental frequency (pitch) where the voice of crying babies is higher. The voice of crying babies has short vocal cords and thin so it can be seen from above spectrogram have tidy characters. So child cry classification becomes a study of interest from past few years.

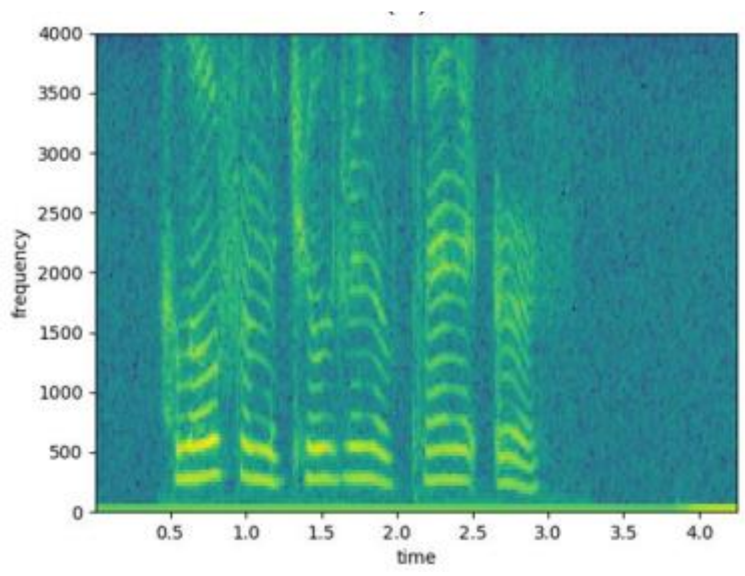


Fig. 1.2: Spectrogram of normal speech signal

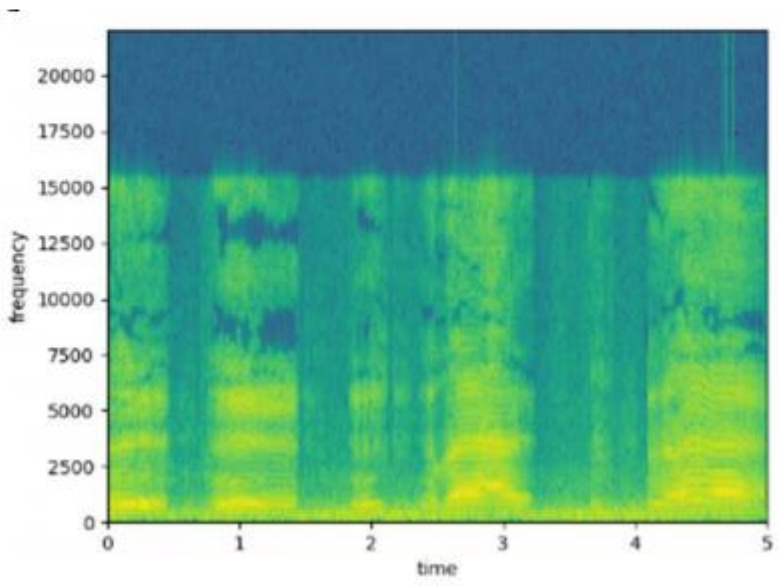
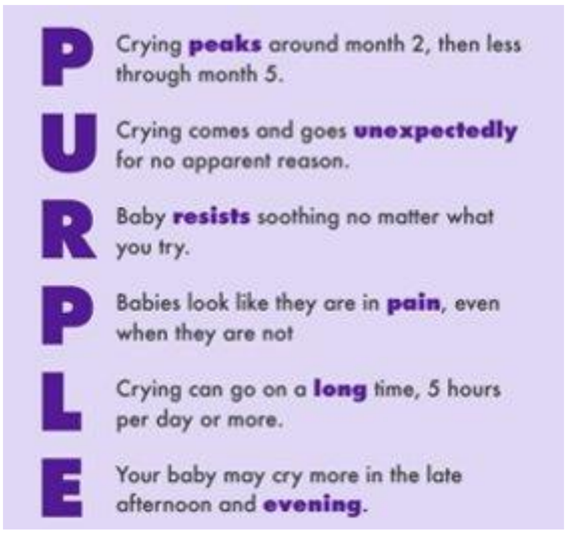


Fig. 1.3: Spectrogram of baby crying signal

As shown in the image, these are few of the factors considered by researchers on this topic. For our research we will focus only on the highlighted 3 points which are unexpected crying (sometimes for no apparent reason), may look like in pain but are actually not, and can cry for a long time. This motivates us further to determine the exact cause for crying.

In absence of verbal communication, the cry often conveys the baby's needs. Using the same, we can identify the reason for crying from categories such as belly pain, hungry, tired, discomfort, burping etc. It will reduce the burden on the mother who is constantly worried by a crying baby, and has to determine the cause by instinct or experience. It will also Aid other caregivers of the baby to handle each cause appropriately. This information can also be used to diagnose the preliminary symptoms of those neonatal diseases to which they are most vulnerable.



P Crying **peaks** around month 2, then less through month 5.

U Crying comes and goes **unexpectedly** for no apparent reason.

R Baby **resists** soothing no matter what you try.

P Babies look like they are in **pain**, even when they are not

L Crying can go on a **long** time, 5 hours per day or more.

E Your baby may cry more in the late afternoon and **evening**.

Fig. 1.4: Focus of research on crying babies

Introduction to Case Study 2 – Bird Song Classification

The birdsong analysis and classification is a very interesting problem to tackle. Birds have many types of voices and the different types have different functions. The most common are song and ‘other voices’ (e.g. call-type). The song is the prettier, using which the birds mark their territory and get partners. It is usually much more complex and longer than “call”.

Call-type voices include contact, enticing and alarm voices. Contact and attracting calls are used to keep birds in a group during flight or foraging, for example in the treetops, alarm ones to alert (e.g. when a predator arrives) or for mating purposes. Most often these are short and simple voices.

Bird songs are longer vocalizations which usually include a variety of notes in a sequence. Sounds that birds make have a grammatical structure; two important levels of organization in this structure are songs and syllables. Syllables are single distinct utterances by a bird and serve as the basic building blocks of bird song. A song consists of a series of syllables arranged in a particular pattern. In this study, our goal is to classify bird species from an interval of sound (containing one or more syllables), which roughly corresponds to the song level of organization.

2. Literature Survey

In [26], a classification model is developed based on three criteria: the degree of overfitting, accuracy, conformability. Total of 468 cry audios are used in the dataset. Different features were extracted for each cry (which included first six formants, intensity, jitter and shimmer values, Harmonic to Noise Ratio (HNR), degree and number of voice breaks, unvoiced pitch fraction and cry duration) Decision Tree, KNN, LDA, LR, SVM, ANN were used for observing the results of the classification models.

In [13] total of 1615 cry samples were analyzed. First, they were pre-processed (silence removal and filtering with 4th order LPF to remove the noise). Each frame was of 25ms with 20% overlap. Different audio features such as pitch, intensity, jitter were proposed and MFCCs were extracted to carry out the classification using PNN (Probabilistic Neural Network).

A radial basis function (RBF) network is implemented for cry classification of infants. Features are extracted from each cry included F0, F1, voiced-ness, energy, first latency, rising melody type, stridor and shift occurrences. [27]

In [7], the work is done to classify the birds in their species according to their sounds. Different features were extracted to train the model for the classification. All the features were extracted on the small frames of the audio. Some of them are described below.

MFCC (Mel Frequency Cepstral Coefficients) - The mel-scale filter banks are used to get the non-linear frequency response like human auditory system. Human Factor Cepstral Coefficients (HFCC) - In HFCC, fc of filters in the filter bank is on the mel scale, but bandwidth for it is different, which computed by the formula, ERB(equivalent rectangular bandwidth) = 6.23*fc^2 + 93.39*fc +28.52).

In [8], all the audios were preprocessed by removing low frequency noise by High Pass Filter and interfering noise by cepstral subtraction. Also, silence was removed by comparing the average energy of a segment with a threshold value) Wavelet Packet Decomposition (WPD) was used to derive the feature from the audio, and K Nearest Neighbors Classifier was used. Along with this, use of 2D cepstral coefficients was also discussed.

Above research work in [7], [8] continued in [9] which used all the features extracted in above papers, and trained models first with the individual features and then by combining them. Features were extracted on the frame basis with the duration of 10ms and overlap of 50%.

In [25], gamma tone frequency cepstral coefficients (GFCC) were explored to classify the bird species. MFCC and GFCC were extracted from the frames of the bird sounds and ANN and SVM models were used for the classification.

Along with this, a study on using Dynamic Time Warping (DTW) was also carried out to get the difference between the spectrograms which could be a useful feature.

LPC coefficients, cepstral coefficients derived from LPC, LPC reflection coefficients, MFCCs, linear mel-filter bank channel and log mel-filter bank channel could also be the useful features.

PCA (Principal Component Analysis), mean computation, Vector Quantization using LBG could be used as the data reduction techniques.

[20]Lung cancer detection was carried out where a 2D convolutional neural network (2D CNN) with Taguchi parametric optimization for automatically recognizing lung cancer from CT images was used. In the Taguchi method, 36 experiments and 8 control factors of mixed levels were selected to determine the optimum parameters of the 2D CNN architecture and improve the classification accuracy of lung cancer. The proposed method is 6.86% and 5.29% more accurate than the original 2D CNN, proving the superiority of proposed model.

3. Features used in Classification

Audio Signals have various temporal and spectral features associated with them. Each of these features contributes to the making of the audio signal the way it is. Here, we are going to take a look at the significant features associated with an audio signal and study them in depth. This would help us understand the need of selecting a particular feature for impending classification in the end which is the ultimate goal.

3.1 MFCC

• Any sound generated by humans is determined by the shape of their vocal tract (including tongue, teeth, etc). If this shape can be determined correctly, any sound produced can be accurately represented.

• The envelope of the time power spectrum of the speech signal is representative of the vocal tract and MFCC (which is nothing but the coefficients that make up the Mel-frequency cepstrum) accurately represents this envelope.

• MFCCs are calculated by collecting the Short Time Fourier Transform (STFT) coefficients of each frame into a prescribed set.

• MFCCs are normally derived as per the following steps:

1. First step is to take the Fourier transform of the pre-processed windowed signal.

2. Using triangular overlapping windows, map the powers of the spectrum obtained on to the mel scale.

3. At each of the mel frequencies take the logs of powers.

4. Finally to the listed mel log power, take Discrete cosine transform.

5. The resulting spectrum’s amplitudes represents the MFCCs

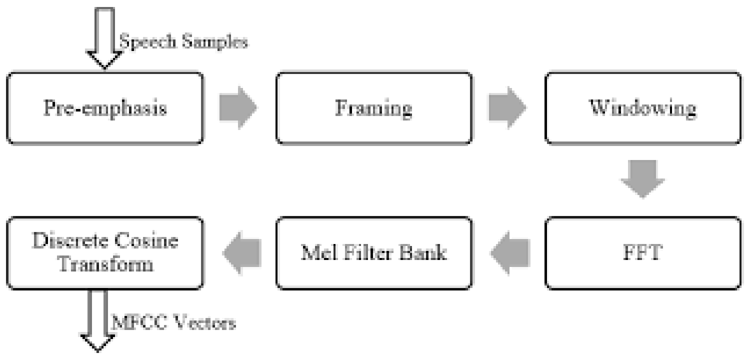


Fig. 3.1: Block Diagram for MFCC

• A frequency measured in Hertz (f) can be converted to the Mel scale using the following formula:

$$\mathrm{Mel}(f) = 2595\log\left(1+\frac{f}{700}\right)$$

Eq. 3.1: Mel scale conversion

• Mel scale is a scale that relates the perceived frequency of a tone to the actual measured frequency. It scales the frequency in order to match more closely what the human ear can hear.

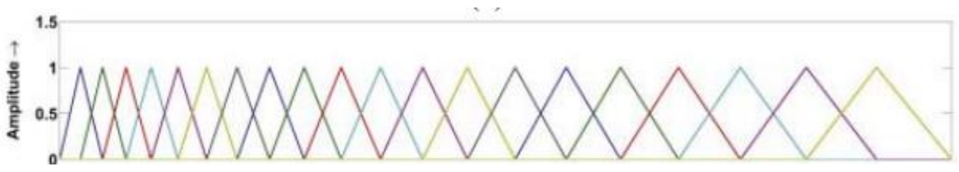


Fig. 3.2: Mel frequency bank

• The final stage in the MFCC process is Discrete Cosine Transformation (DCT) after the results of the previous process is converted back into the time domain, so that the signal can be presented well.

• These results form a row of an acoustic vector which named Mel Frequency Cepstral Coefficient.

3.2 Spectral Flatness

• Spectral flatness or tonality coefficient, also known as Wiener entropy, is a measure used in digital signal processing to characterize an audio spectrum.

• Spectral flatness is typically measured in decibels, and provides a way to quantify how tone-like a sound is, as opposed to being noise-like.

• This is measure of uniformity in the frequency distribution of the power spectrum.

• The spectral flatness is calculated by dividing the geometric mean of the power spectrum by the arithmetic mean of the power spectrum:

$$\text{Flatness} = \frac{\sqrt[N]{\prod_{n=0}^{N-1} x(n)}}{\frac{\sum_{n=0}^{N-1} x(n)}{N}} = \frac{\exp\left(\frac{1}{N}\sum_{n=0}^{N-1} \ln x(n)\right)}{\frac{1}{N}\sum_{n=0}^{N-1} x(n)}$$

Eq. 3.2: Spectral Flatness

3.3 Loudness

• Loudness, in acoustics, attribute of sound that determines the intensity of auditory sensation produced.

• The loudness of sound as perceived by human ears is roughly proportional to the logarithm of sound intensity.

• The perceived loudness depends on the nature of the speech sound, due to loading of the vocal tract system on the vocal source during the production.

• Loudness is also affected by the behavioral characteristics of the speaker, such as emotional state of the speaker, which in turn useful for classification of child cry.

3.4 Spectral Centroid

• The spectral centroid is a measure used in digital signal processing to characterize a spectrum.

• It indicates where the center of mass of the spectrum is located. It has a robust connection with the impression of brightness of a sound.

• This basically means higher the SC, more intense / brighter the sound and a lower SC corresponds to a less intense audio signal.

$$\text{SC} = \frac{\sum_{k=0}^{N/2} k|X(k)|^2}{\sum_{k=0}^{N/2} |X(k)|^2}$$

where $X(k)$ is the DFT of the frame, N is the size of the DFT and k varies from 0 $\ldots N-1$.

Eq. 3.3: Spectral Centroid

3.5 Spectral Flux

• Spectral flux is a measure of how quickly the power spectrum of a signal is changing.

• It is calculated by comparing the power spectrum for one frame against the power spectrum from the previous frame.

$$\mathrm{SF}_i = \sum_{k=0}^{N/2} \left| \left| X_{(i+1)} - X_i \right| \right|$$

Eq. 3.4: Spectral Flux

3.6 Bandwidth

• It is the width of the frequency band of the frame around the center point of the spectrum.

$$\mathrm{BW} = \sqrt{\frac{\sum_{k=0}^{N/2} (k - SC)^2 |X(k)|}{\sum_{k=0}^{N/2} |X(k)|^2}}$$

Eq. 3.5: Bandwidth

3.7 GFCC

• In the context of non-speech audio recognition and classification for multimedia applications, it becomes essential to have a set of features able to accurately represent and discriminate among audio signals.

• Mel frequency cepstral coefficients (MFCC) have become a de facto standard for audio parameterization.

• Taking as a basis the MFCC computation scheme, the GFCCs are a biologically inspired modification employing Gamma-tone filters with equivalent rectangular bandwidth bands.

• A gammatone filter is a linear filter described by an impulse response that is the product of a gamma distribution and sinusoidal tone. It is a widely used model of auditory filters in the auditory system.

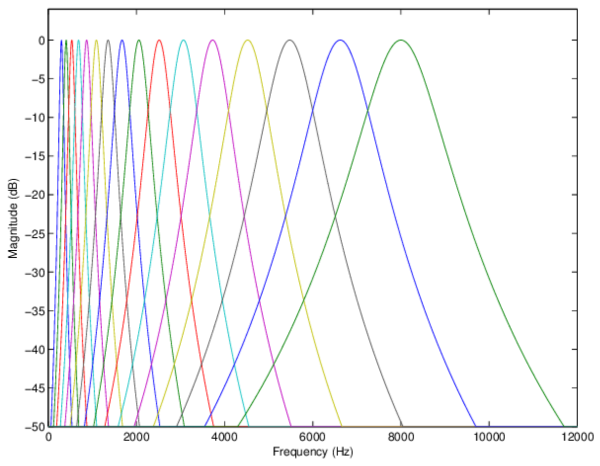


Fig. 3.3: Gammatone filter

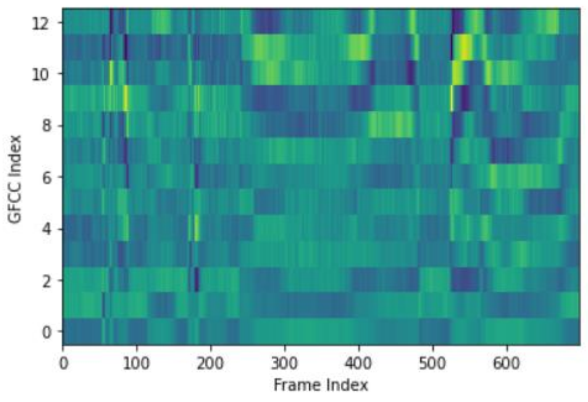


Fig. 3.4: GFCC frames

Block Diagram:

- The block diagram for obtaining GFCCs is similar to that of obtaining MFCCs, the major difference being the usage of Gammatone filters instead of the Mel Frequency Filter Bank in case of MFCC.

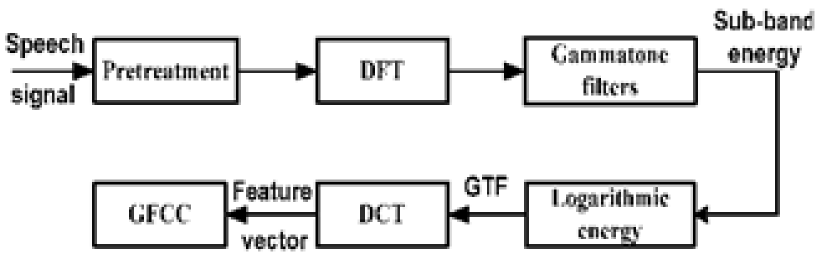


Fig. 3.5: Block Diagram for GFCC

• GFCCs are normally derived as per the following steps:

1. First step is to take the Discrete Fourier transform of the pre-processed windowed signal.

2. Then the signal is passed through the Gammatone filters.

3. The obtained sub band energy is then converted to the Logarithmic energy

4. Finally, upon taking the Discrete Cosine Transform, GFCCs are obtained.

• With a similar computational cost, the GFCC are more effective than MFCC in representing the spectral characteristics of non-speech audio signals, especially at low frequencies.

3.8 STACF

• Autocorrelation is a mathematical representation of the degree of similarity between a given time series and a lagged version of itself over successive time intervals.

• We perform autocorrelation on each frame and hence it is called Short Term Autocorrelation Function.

• STACF can be used to determine whether a given audio signal is periodic or aperiodic as it finds repeating events and hence is an indicator of pitch.

• It is also used in pattern recognition.

• We often normalize this measure for a consistent analysis.

$$\hat{\rho}_k = \frac{\sum_{t=k+1}^{T} (r_t - \bar{r})(r_{t-k} - \bar{r})}{\sum_{t=1}^{T} (r_t - \bar{r})^2}$$

Eq. 3.6: Short Term Autocorrelation function

3.9 ZCR

• The zero-crossing rate is the rate of sign-changes along a signal, i.e., the rate at which the signal changes from positive to zero to negative or from negative to zero to positive.

• This feature has been used heavily in both speech recognition and music information retrieval, being a key feature to classify percussive sounds.

• ZCR can be used to determine the smoothness of a signal and can also determine whether a given audio signal is voiced or unvoiced.

$$ZCR = \frac{1}{2N}\sum_{n=1}^{N} |sign(x[n]) - sign(x[n-1])|$$

Eq. 3.7: Zero Crossing Rate

3.10 LPC

• Linear predictive coding (LPC) is a method used mostly in audio signal processing and speech processing for representing the spectral envelope of a digital signal of speech in compressed form, using the information of a linear predictive coding.

• It is one of the most powerful speech analysis techniques, and one of the most useful methods for encoding good quality speech at a low bit rate and provides highly accurate estimates of speech parameters.

• LPC is the most widely used method in speech coding and speech synthesis.

• LPC works on the human speech production model in which there is an impulse train generator for voiced speech and this passes through the vocal tract, which change its shape to utter different utterances and is represented by the all-pole synthesis filter, whose filter coefficients are represented by LPC coefficients

• As it is the all-pole filter, the next sample can be predicted by the weighted sum of previous samples and as we are taking finite number of coefficients, there is a difference between predicted and actual sample which is denoted by an additional coefficient which is error 'e'

• Thus, N order LPC calculates this N coefficients along with an additional error term 'e' which is the difference between the predicted and actual value of the sample.

• LPC coefficients are evaluated over one frame, and as number of LPC coefficients are much less than the actual number of samples in the frame, it achieves good compression rate for the transmission of speech signal. Using the LPC coefficients and the error term, the original signal can be constructed again.

• As we saw earlier, that LPC coefficients represent the filter coefficients of the all-pole filter in the human speech production mechanism, these coefficients can be effectively used as the features which model the shape of the vocal tract and hence the shape of the spectral envelope as well and can be useful in the audio classification task.

• There are different methods to calculate the LPC coefficients, some of them are having time complexities while some of them are recursive and reduce the time complexity to much extent Levinson-Durbin algorithm, Burg's method are widely used algorithms for computing LPCs.

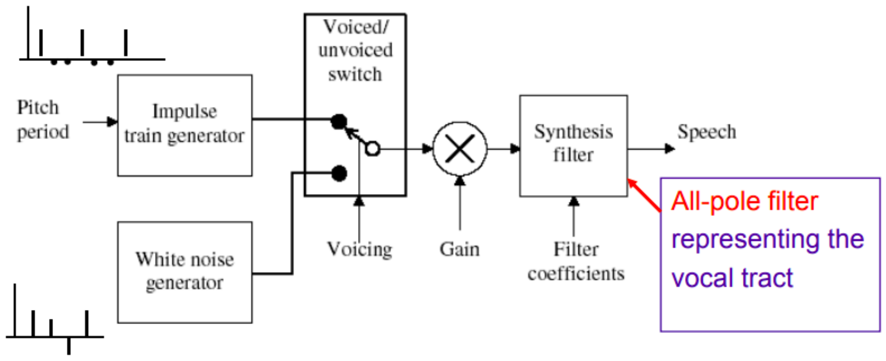


Fig. 3.6: Block Diagram for LPC

3.11 Spectral Roll-off

• Spectral roll off is the frequency below which a specified percentage of the total spectral energy, e.g., 95%, lies. It is related to the skewness of the spectral shape.

• It is given by the formula:

$$\text{SRF} = \max\left(M \sum_{k=0}^{M} |X(k)|^2 < 0.95 \sum_{k=0}^{N/2} |X(k)|^2 \right)$$

Eq. 3.8: Spectral Rolloff

• If roll-off factor is 100%, we get the maximum frequency and if it is 0% then we get the minimum frequency.

• This is evaluated over each frame's spectrum.

3.12 Mel Spectrogram

- Studies have shown that humans do not perceive frequencies on a linear scale. We are better at detecting differences in lower frequencies than higher frequencies. In 1937, Stevens, Volkmann, and Newmann proposed a unit of pitch such that equal distances in pitch sounded equally distant to the listener. This is called the mel scale. We perform a mathematical operation on frequencies to convert them to the mel scale.

- When the FFT is computed on overlapping windowed segments of the audio signal, and all these spectrums are put together with time on x axis and frequency on y axis, what we get what is called the spectrogram. We can think of a spectrogram as a bunch of FFTs stacked on top of each other. When the frequency scale of the spectrogram is converted to the mel-scale, we get the Mel Spectrogram. This Mel-Spectrogram can be used as an input to the CNN model.

- Mel Spectrogram gives us the all the spectral information of all the frames in the audio signal in mel-scale, which is according to the human ear response.

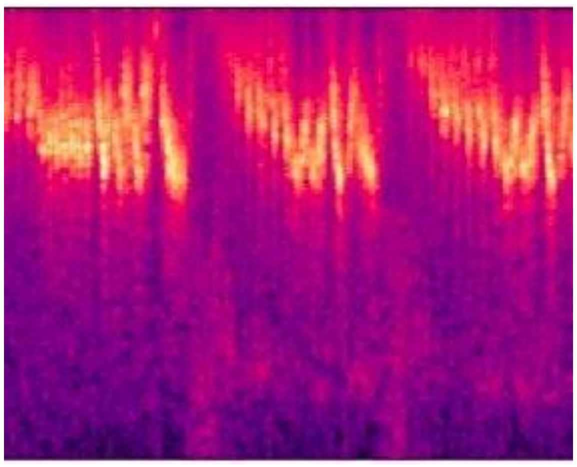


Fig. 3.7: Mel spectrogram

Here is the mel spectrogram of one of the bird songs from our dataset.

Most of the temporal and spectral features such as MFCC, GTCC, ZCR etc were studied in detail and their significance was understood along with the mathematical equations through which these features can be calculated from a raw audio signal. Having a holistic understanding of these features would help a great deal in selecting the best features for differentiating between audio signals and accurately classifying them.

4. Classifiers

The next important step is to discuss the classifiers we are going to use, given the end goal of our discussion is to be able to accurately classify the given audio signals. Here, we are going to take a look at Supervised Learning and Unsupervised Learning. In Supervised Learning, there are two parts namely, Machine Learning and Deep Learning. In case of Machine Learning, four classifying models have been listed in detail along with their algorithms. In Deep Learning, we delve into Convolutional Neural Networks as well as optimization strategies (Taguchi Method). The chapter is concluded with the discussion of important model performance parameters which give us an idea about the efficiency/shortcomings of our model.

4.1 Supervised Learning

4.1.1 Machine Learning

4.1.1.1 Multiclass Logistic Regression (LR)

When we have more than two class instances in input data, then it might get complex to analyze the data, train the model, and predict relatively accurate results. To handle these multiple class instances, we use multi-class classification.

In One vs All algorithm we need N classifiers whereas in One vs One algorithm we will require O(N2) classifiers.

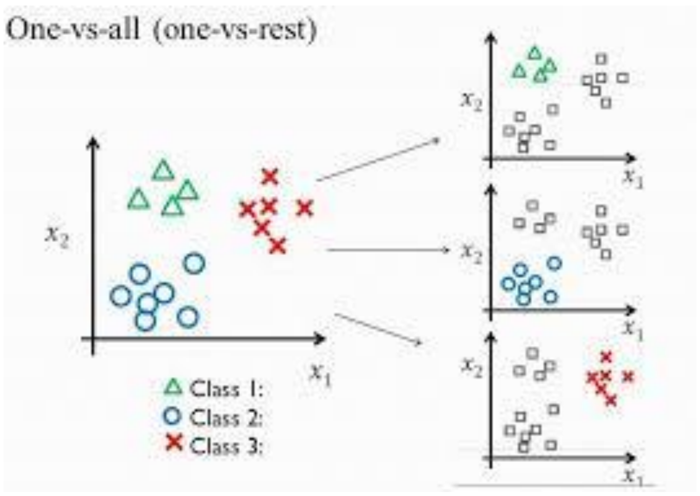


Fig. 4.1: Graphical Representation of Multiclass Logistic Regression

Logistic Regression is the model which used for the classification purpose i.e., when the dependent variable is categorical. This model determines the equation of the boundary (line, plane or hyperplane) by minimizing the cost function which tries to divide the dataset in the parts, which each part belonging to each class.

If we substitute the data-point value in the LHS part of the boundary equation (where RHS is 0) then we get the value giving idea of the perpendicular distance of that point from the boundary. More is the distance from the boundary, more is its probability to belong to any one of the 2 classes. This is given by:

$$Z = WX + B$$

Eq. 4.1: Boundary equation

$$h\Theta(x) = \text{sigmoid}\ (Z)$$

Eq. 4.2: Logistic regression

Hence, to get a probabilistic measure, it is further computed using the sigmoid function, which is given by:

$$S(x) = \frac{1}{1 + e^{-x}}$$

Eq. 4.3: Sigmoid function

And it graph looks like this:

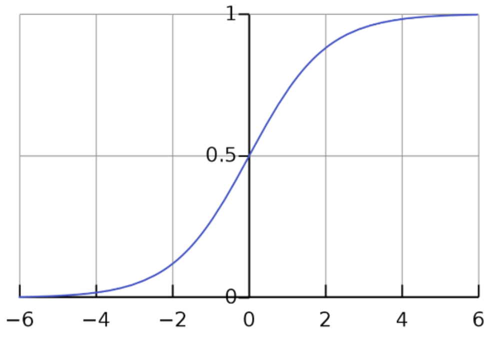


Fig. 4.2: Graph of Sigmoid function

Thus, if a point is lying on the boundary, then it will evaluate to 0 and after applying sigmoid function on it, probability of it to belong to the class 1 will turn out to be 0.5. Sigmoid function’s output ranges from 0 to 1 as it gives the probabilistic measure.

The cost function used to minimize is:

$$\text{Cost}(h_\Theta(x), Y(\text{actual})) = -\log(h_\Theta(x)) \text{ if } y=1$$
$$-\log(1-h_\Theta(x)) \text{ if } y=0$$

Eq. 4.4: Logistic regression cost function

where y is the actual y label.

In case of multiple classes, multiclass logistic regression is used which decides the boundaries in one vs all manner. If there are total N classes, then it finds out the N boundaries classifying this N classes. Each boundary for a particular class is evaluated by taking that class as class 1 and treating all the remaining classes as another class, hence called as one-vs-all. This is done N times to get the N boundaries for all the N classes.

```
Algorithm:
1.Initialize: $\theta_2 = 0$ for all $0 \le j \le m$.
2.Repeat many times:
        Gradient(j) =0  for all $0 \le j \le m$
        For each training example (x,y):
            For each parameter j:
                Gradient (j)  +=  xj(y- $\frac{-1}{1+e^{-\theta^T x}}$)
        $\theta_2$ +* $gradient(j)$ , for all $0 \le j \le m$.
```

Fig. 4.3: Algorithm for Multiclass Logistic Regression

4.1.1.2 Support Vector Machine (SVM)

In a Support Vector Machine, we construct a hyperplane in a higher order dimensional space. The objective of optimization is to maximize the perpendicular distance (margin) between the positive and negative sample space. The plane which maximizes this perpendicular distance defines the boundary between the 2 classes. The 2 samples which result in this maximum margin are called the support vectors and hence the name. This same model can be extended to N classes where we need to define N – 1 boundaries.

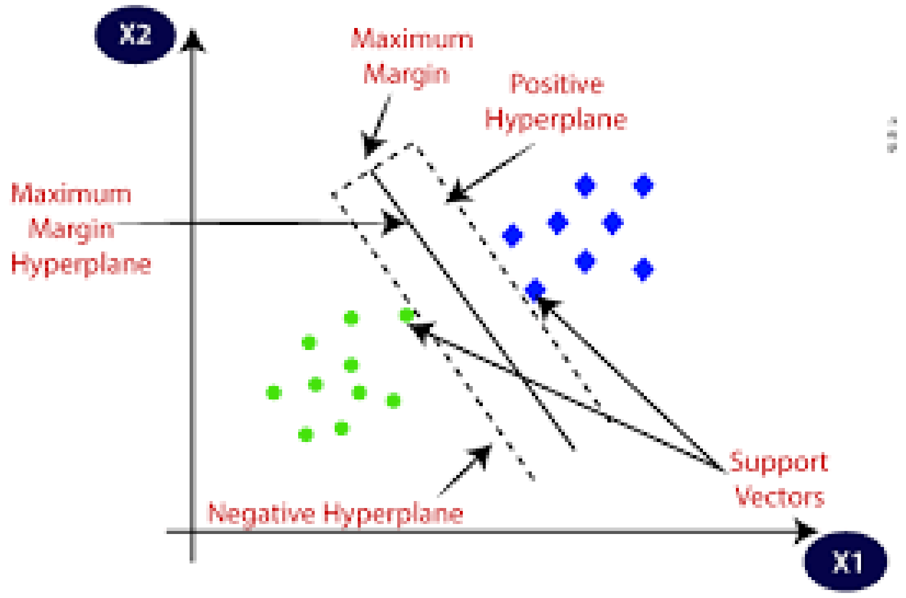
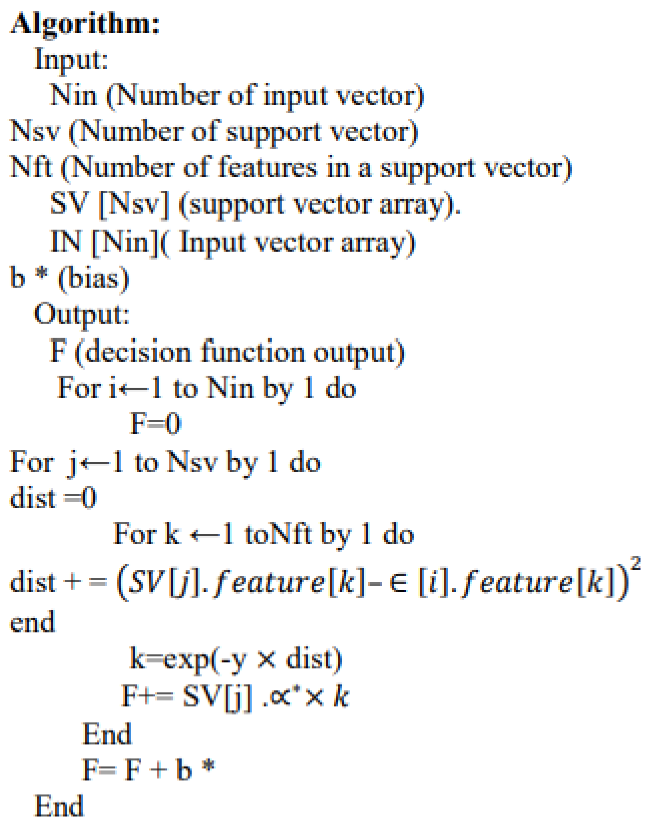


Fig. 4.4: Graphical Representation of SVM

The working of the SVM algorithm can be understood by using an example. Suppose we have a dataset that has two tags (green and blue), and the dataset has two features x1 and x2. We want a classifier that can classify the pair (x1, x2) of coordinates in either green or blue. So, as it is 2D space so by just using a straight line, we can easily separate these two classes. But there can be multiple lines that can separate these classes. If data is linearly arranged, then we can separate it by using a straight line, but for non-linear data, we cannot draw a single straight line. So, to separate these data points, we need to add one more dimension. For linear data, we have used two dimensions x and y, so for non-linear data, we will add a third-dimension z. It can be calculated as: z=x2 +y2. By adding the third dimension, the sample space will become linearly separable.



```
Algorithm:
  Input:
    Nin (Number of input vector)
Nsv (Number of support vector)
Nft (Number of features in a support vector)
    SV [Nsv] (support vector array).
    IN [Nin]( Input vector array)
b * (bias)
  Output:
    F (decision function output)
     For i←1 to Nin by 1 do
            F=0
For  j←1 to Nsv by 1 do
dist =0
          For k ←1 toNft by 1 do
dist += $(SV[j].feature[k] - \in [i].feature[k])^2$
end
            k=exp(-y × dist)
           F+= SV[j] .∝* × k
        End
        F= F + b *
  End
```

Fig. 4.5: Algorithm for SVM

4.1.1.3 K-Nearest Neighbors Classifier (KNN)

We can say that the training data are vectors in a multidimensional feature space and it is labelled with the correct class. In order to classify, first we assign a constant value to k. A test vector is classified by observing k training vectors nearest to that point and then it is assigned the class which is most frequent among these k samples. The distance can be calculated in continuous variables by using Euclidean distance. We can use other metrics such as overlap metric also known as Hamming distance and l-n norms using Euclidean distance.

We can use other metrics such as overlap metric also known as Hamming distance and l-n norms, among others. KNN algorithm uses feature similarity (proximity) to predict the class of test data based on its proximity to the testing data vectors.

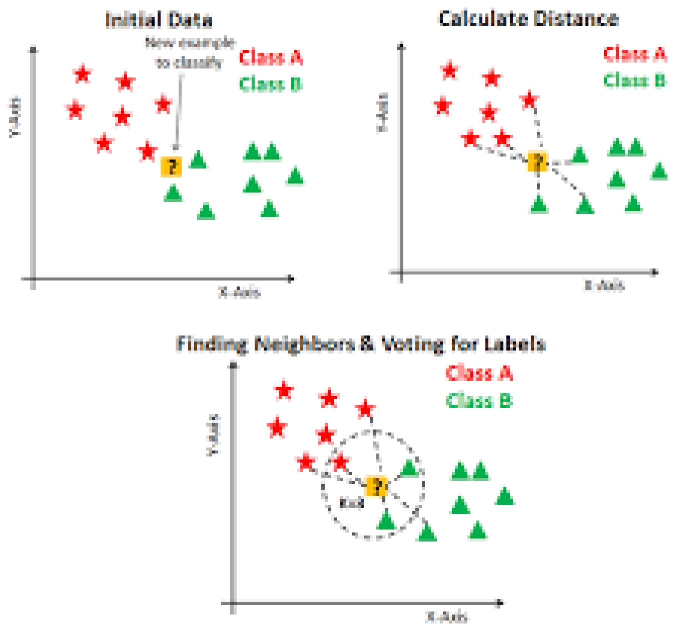


Fig. 4.6: Graphical Representation of KNN

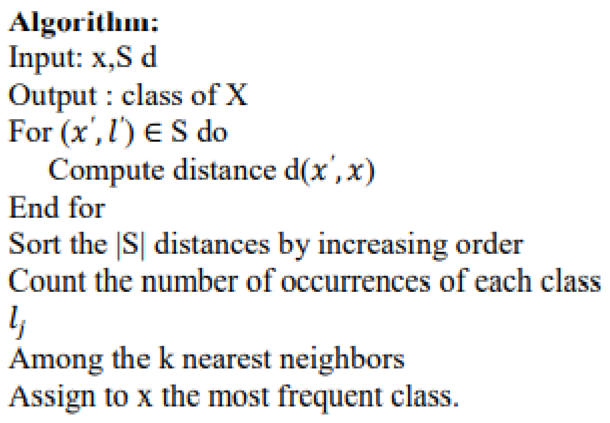
This algorithm and its implementation follow the steps as mentioned: For each test data vector −

1) Calculate the distance between the test vector and each training data sample. The distance can be estimated using Euclidean, Manhattan or Hamming distance. The most popular metric used is the Euclidean distance.

2) Sort the distance values in ascending order.

3) For the top K values, check the labels.

4) Assign the most frequently occurring label to the test vector.



Algorithm:

```
Input: x,S d
Output : class of X
For $(x', l') \in$ S do
    Compute distance d$(x', x)$
End for
Sort the |S| distances by increasing order
Count the number of occurrences of each class
$l_j$
Among the k nearest neighbors
Assign to x the most frequent class.
```

Fig. 4.7: Algorithm for KNN

4.1.1.4 Random Forest Classifier (RF)

Random forest classifier involves multiple decision trees, which collectively predict the label of a sample. Each individual tree in the random forest predicts one of the available classes and the most frequently occurring class is assigned to that sample.[23] The individual decision trees must be random with low correlation. This protects against and prevents the likelihood of misclassification. Except for a few trees, a most decision trees give an accurate prediction and the collective error is low.

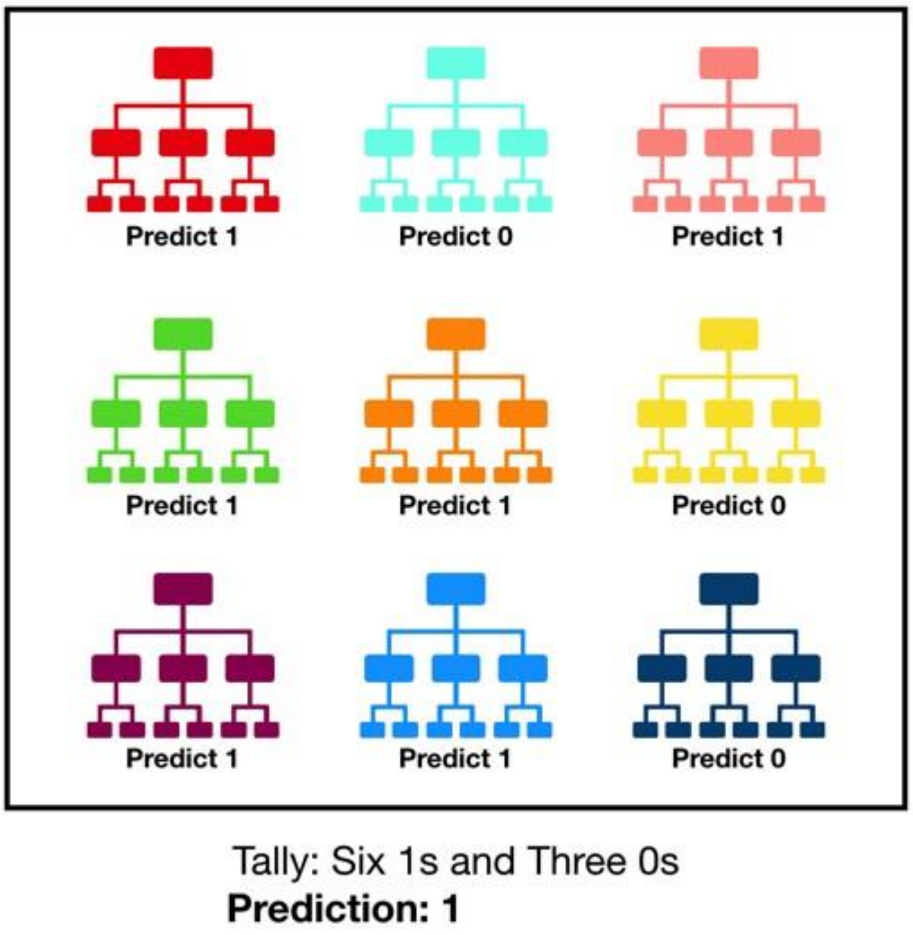


Fig. 4.8: Graphical Representation of RF

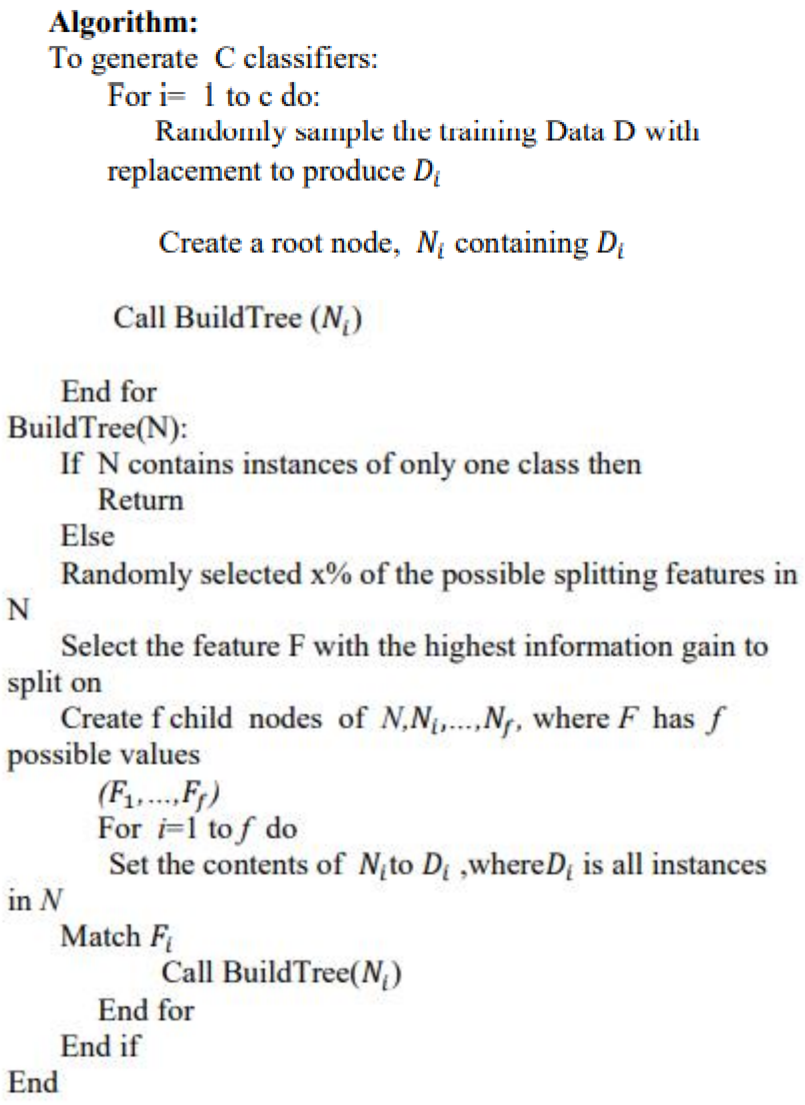
This algorithm proceeds with the following steps −

1. Select random samples from the database.

2. Construct a decision tree for each sample. Obtain the prediction from each decision tree.

3. Count the frequency of results for each class.

4. Select the most frequent result as the final prediction.



```
Algorithm:
To generate C classifiers:
    For i= 1 to c do:
        Randomly sample the training Data D with
    replacement to produce $D_i$

        Create a root node, $N_i$ containing $D_i$

        Call BuildTree ($N_i$)

  End for
BuildTree(N):
  If N contains instances of only one class then
      Return
  Else
  Randomly selected x% of the possible splitting features in
N
  Select the feature F with the highest information gain to
split on
  Create f child nodes of $N,N_1,...,N_f$, where $F$ has $f$
possible values
      $(F_1,...,F_f)$
      For $i$=1 to $f$ do
       Set the contents of $N_i$ to $D_i$ ,where $D_i$ is all instances
in $N$
  Match $F_i$
          Call BuildTree($N_i$)
      End for
  End if
End
```

Fig. 4.9: Algorithm for RF

4.1.2 Deep Learning

4.1.2.1 Convolutional Neural Networks (CNN)

- CNN takes an input image, assigns importance to various aspects in the image and does classification.
- Convolutional Layer – Kernel/Filter bank convolve with input image and produce the output images.
- Pooling Layer – Replaces the output by deriving a summary statistic of the nearby outputs. e.g., max, avg, global pooling

- Non-Linearity - As images are not linear, non-linearity layers are introduced in CNN. ReLU is widely used non-linearity while other functions include sigmoid, tanh etc.
- At the end, output is flattened and given to Fully Connected Layer which helps to map the representation between the input and the output.

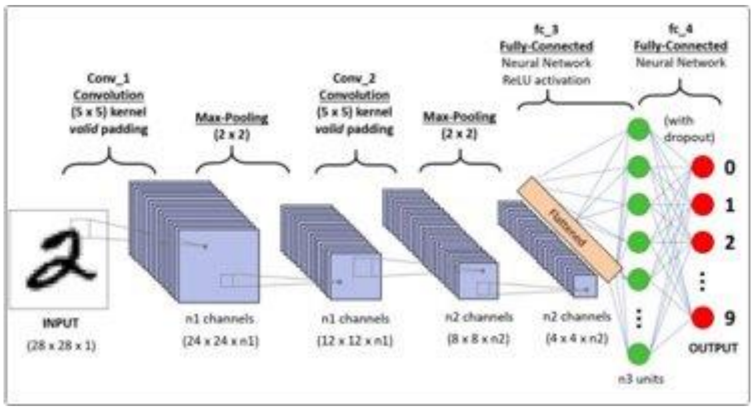
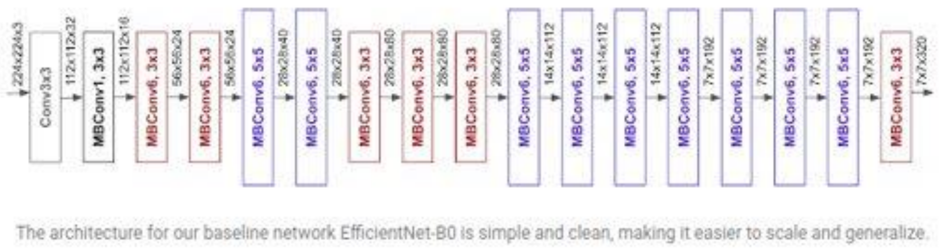


Fig. 4.10: Representation of layers of CNN

Architechture:

- 1408 images of mel spectrograms of 5 bird species were divided into training and validation dataset with split of 75:25
- Pretrained CNN Architecture – EfficientNetB3 was used
- EfficientNet is very famous for its ability to scale up the network from its base network EfficientNetB0 without affecting the accuracy.
- It has got very good accuracy of 95.6% on ImageNet and has around 11M parameters
- It takes an input image of size 224x224x3, uses swish activation function f(x) = x*sigmoid(x).



The architecture for our baseline network EfficientNet-B0 is simple and clean, making it easier to scale and generalize.

Fig. 4.11: Architecture of EfficientNet

Training:

- ImageDataGenerator() from keras was used to generate the batches of images after preprocessing and augmenting (zoom, width shift, height shift) the images

- Images were transformed to shape 224x224x3 and batch size was 16
- To train the EfficientNetB3 initialized with the weights from 'imagenet', 30 epochs were used with step size per epoch = 60
- Optimizer used – Adam
- Loss Function – Categorical Cross Entropy
- Metric – Accuracy
-

4.1.2.2 Taguchi Method

- Taguchi method is statistical method, developed by Genichi Taguchi to improve the performance of the process.
- The Taguchi method uses an orthogonal array (OA), consisting of factors and levels, to classify the results to optimize process parameters. In the Taguchi method, the experimental design using OA tables enables the optimization process. The results are converted to a signal-to-noise (S/N) ratio to determine the combination of control factor levels to improve system stability and reduce quality loss.
- Thus, it can also be used to tune the hyperparameters of the model. If there are more than N hyperparameters, with each parameter having L values to be tested upon, then there are total L^N possible combinations to do the experiments and find the best set of parameters. Taguchi method decreases this number of experiments by the use of Orthogonal Arrays and we can find the best parameters by performing significantly less number of experiments.
- Depending on the number of parameters and number of levels, the proper orthogonal array is chosen. E.g. For 5 parameters with 4 levels each, L16 OA is used which designs 16 experiments to be performed, which is very less when compared to 4^5 number of experiments.
- To determine the effect each variable has on the output, the signal-to-noise ratio, or the SN number, needs to be calculated for each experiment conducted. There are 3 ways of calculating this SN number: larger-is-better, smaller-is-better, nominal-is-better.
- Larger-is-better

$$SN_i = -10\log\left[\frac{1}{N_i}\sum_{u=1}^{N_i}\frac{1}{y_u^2}\right]$$

Eq. 4.5: Larger-is-better

- Smaller-is-better

$$SN_i = -10 \log \left(\sum_{u=1}^{N_i} \frac{y_u^2}{N_i} \right)$$

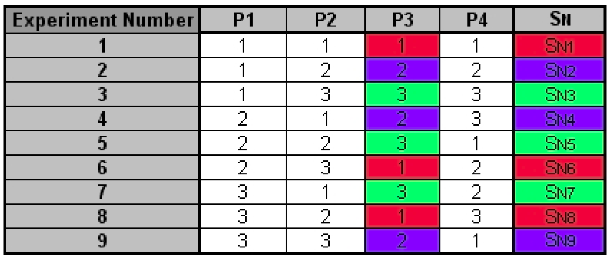
Eq. 4.6: Smaller-is-better

- Nominal-is-better

$$SN_i = 10 \log \frac{\bar{y}_i^2}{s_i^2}$$

Eq. 4.7: Nominal-is-better

- After calculating the SN ratio for each experiment, the average SN value is calculated for each factor and level. This is done as shown below for Parameter 3 (P3) in the array



Experiment Number	P1	P2	P3	P4	SN
1	1	1	1	1	SN1
2	1	2	2	2	SN2
3	1	3	3	3	SN3
4	2	1	2	3	SN4
5	2	2	3	1	SN5
6	2	3	1	2	SN6
7	3	1	3	2	SN7
8	3	2	1	3	SN8
9	3	3	2	1	SN9

Fig. 4.12: SN ratios

$$SN_{P3,1} = \frac{(S_{N1} + S_{N6} + S_{N8})}{3}$$

$$SN_{P3,2} = \frac{(S_{N2} + S_{N4} + S_{N9})}{3}$$

$$SN_{P3,3} = \frac{(S_{N3} + S_{N5} + S_{N7})}{3}$$

Eq. 4.8: Pairwise SN ratios

- Once these SN ratio values are calculated for each factor and level, they are tabulated and the range R (R = high SN - low SN) of the SN for each parameter is calculated and entered into the table. The larger the R value for a parameter, the larger the effect the variable has on the process. This is because the same change in signal causes a larger effect on the output variable being measured.

4.2 Unsupervised Learning

4.2.1 K-Means

- Our goal is to assign each point to a cluster, or group. To do this, we need to find out where the clusters are, and which points should belong to each one.
- The center of each cluster, known as centroid, is the mean of all the data points that belong to the cluster.
- Each data point belongs to the cluster with the nearest centroid.
- We can use several distance metrics, such as Manhattan and Chebyshev, but we will stick to the more familiar Euclidian.

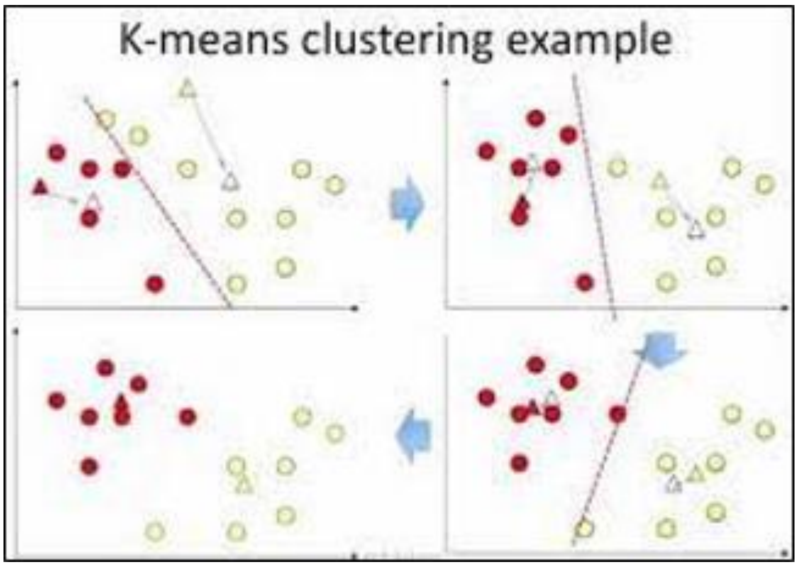
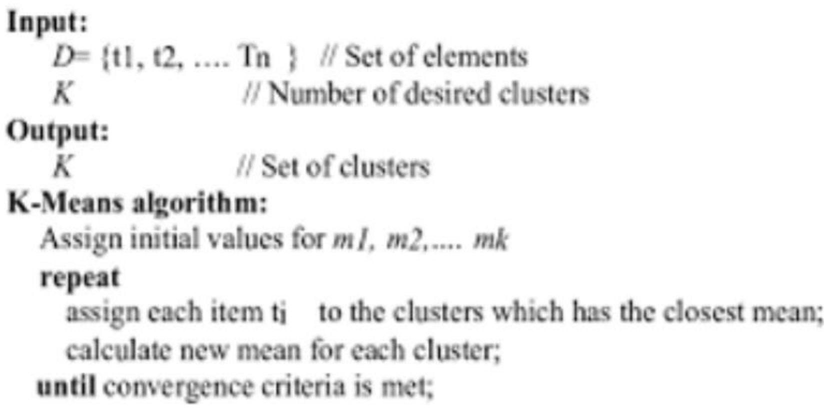


Fig. 4.13: K means clustering example

Algorithm:

1. Initialize a mean for each cluster by randomly picking points from the dataset and using these as starting values for the means.
2. Assign each point to the nearest cluster using Euclidean distance.
3. Compute the means for each cluster as the mean (centroid) for all the points that belong to it.
4. Reiterate steps 2 and 3 either a pre-specified number of times, or until convergence.



```
Input:
    D= {t1, t2, .... Tn }   // Set of elements
    K                       // Number of desired clusters
Output:
    K                       // Set of clusters
K-Means algorithm:
  Assign initial values for m1, m2,.... mk
  repeat
    assign each item ti   to the clusters which has the closest mean;
    calculate new mean for each cluster;
  until convergence criteria is met;
```

Fig. 4.14: Algorithm for K-Means clustering

4.2.2 Elbow Method

- If an appropriate value of k is not known from prior knowledge, it must be chosen somehow.
- The elbow method looks at the percentage of variance explained as a function of the number of clusters: One should choose the number of clusters so that adding another cluster doesn't give much better modeling of the data.
- If we plot the percentage of variance explained by the clusters against the number of clusters, the first clusters will add much information, but at some point, the marginal gain will drop, giving an angle in the graph.
- The number of clusters is chosen at this point. Although, this "elbow" cannot always be unambiguously identified.

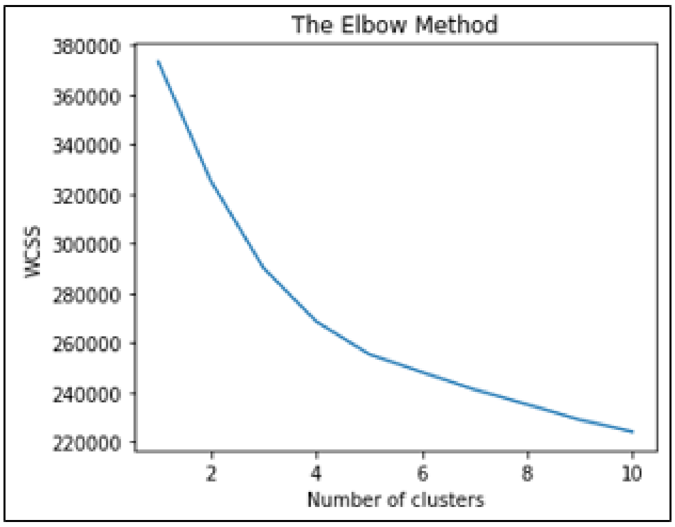


Fig. 4.15: Graph for Elbow Method

4.2.3 Silhouette Score

- The average silhouette of the data is another useful criterion to determine k.
- The silhouette of a data instance is a measure of how closely it is matched to data within its cluster and how loosely it is matched to data of the nearest neighboring cluster, i.e., the cluster whose average distance from the datum is lowest.
- A silhouette close to 1 implies the datum is in an appropriate cluster, while a silhouette close to −1 implies the datum is in the wrong cluster. A score of 0 means that the clusters are overlapping.

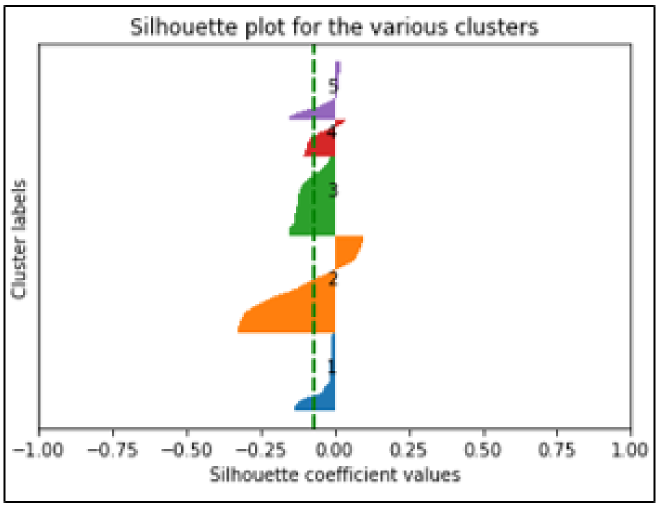


Fig. 4.16: Silhouette plot for various clusters

4.2.4 Gaussian Mixture Models (GMM)

- As the name implies, a Gaussian mixture model involves the mixture (i.e. superposition) of multiple Gaussian distributions.
- Here rather than identifying clusters by “nearest” centroids, we fit a set of k gaussians to the data.

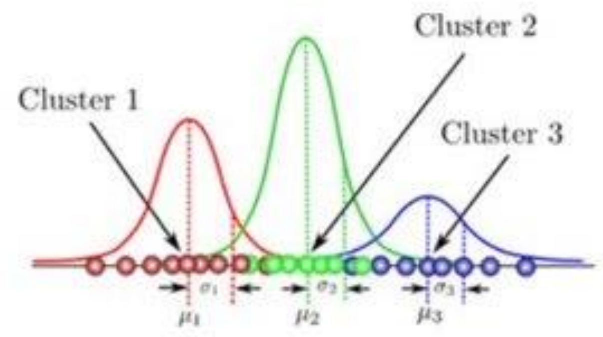


Fig. 4.17: GMM plot for various clusters

- And we estimate gaussian distribution parameters such as mean and Variance for each cluster and weight of a cluster.

Algorithm:

1. Initialize the mean *μk,* The covariance matrix Σ*k* and the mixing coefficients *πk* by some random values (or other values).
2. Compute the *Ck* values for all k.
3. Again, estimate all the parameters using the current C_k values.
4. Compute log-likelihood function.
5. Put some convergence criterion
6. If the log-likelihood value converges to some value (or if all the parameters converge to some values) then stop, else return to Step 2.

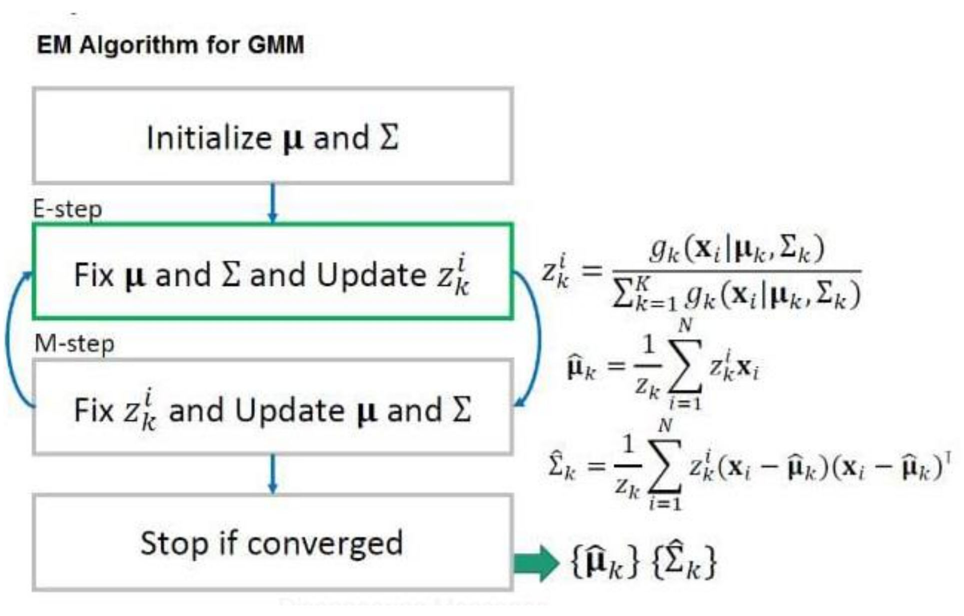

EM Algorithm for GMM
Initialize $\boldsymbol{\mu}$ and Σ
E-step
Fix $\boldsymbol{\mu}$ and Σ and Update z_k^i
M-step
Fix z_k^i and Update $\boldsymbol{\mu}$ and Σ
Stop if converged
$z_k^i = \frac{g_k(\mathbf{x}_i|\boldsymbol{\mu}_k,\Sigma_k)}{\sum_{k=1}^K g_k(\mathbf{x}_i|\boldsymbol{\mu}_k,\Sigma_k)}$
$\hat{\boldsymbol{\mu}}_k = \frac{1}{Z_k}\sum_{i=1}^N z_k^i \mathbf{x}_i$
$\hat{\Sigma}_k = \frac{1}{Z_k}\sum_{i=1}^N z_k^i (\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)(\mathbf{x}_i - \hat{\boldsymbol{\mu}}_k)^T$
$\{\hat{\boldsymbol{\mu}}_k\}\ \{\hat{\Sigma}_k\}$

Fig. 4.18: Algorithm for GMM clustering

4.3 Model performance parameters

We have used the following parameters to evaluate the performance of our model:

PRECISION: Precision gives the probability of true positive results among all positive results.

RECALL: Recall gives the probability of true positive results among all expected positive results.

F1 SCORE: A metric that combines precision and recall as the harmonic mean of precision and recall is known as the traditional F-measure or balanced F-score.

ACCURACY: Accuracy is also used as a statistical measure of how well a classification test correctly identifies a class. The accuracy is the probability of correct results (true positives and true negatives) among all examined samples.

Table 4.1: Confusion Matrix

		Predicted	
		Negative	Positive
Actual	Negative	True Negative (TN)	False Positive (FP)
	Positive	False Nagative (FN)	True Positive (TP)

$$precision = \frac{TP}{TP + FP}$$

Eq. 4.9: Precision

$$recall = \frac{TP}{TP + FN}$$

Eq. 4.10: Recall

$$F1 = \frac{2 \times precision \times recall}{precision + recall}$$

Eq. 4.11: F1 - Score

$$accuracy = \frac{TP + TN}{TP + FN + TN + FP}$$

Eq. 4.12: Accuracy

The various classifiers in Supervised and Unsupervised Learning were studied in depth along with parameters that would help us in judging the performance of our model. Optimization through Taguchi methods was also studied to be employed for the training of CNNs, in order to yield better results more efficiently. The efficacy and theory behind Unsupervised Learning models was also studied in detail.

5. Implementation

5.1 Methodology

The general structure of the designed system is given in Figure 1.

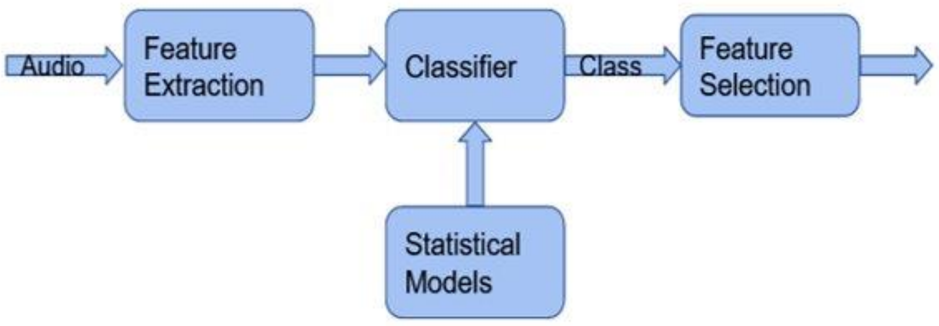


Fig. 5.1: Methodology of classification

FEATURE EXTRACTION:

The primary goal of this step is the conversion of audio signals into a set of numeric values which represent the signal in a very unique and compact way conveying only relevant information. As shown in the figure1 several audio signal features are extracted and by selecting the best features we classify them using classifying models which are explained later.

- The temporal features (time domain features) like the energy of signal, zero crossing rate, maximum amplitude, minimum energy, etc.
- The spectral features (frequency domain features) like MFCC, LPC, fundamental frequency, frequency components, spectral flux, spectral density, spectral centroid, spectral roll-off, etc.

5.2 Case Study 1 – Child Cry Classification

5.2.1 Resource Requirements

Software requirements:

Language – Python 3

Table 5.1: Software libraries' description

Sr. No.	Library	Description
1.	librosa	For audio analysis – used for features
2.	sklearn	For Machine learning models
3.	spafe	For GFCC extraction
4.	numpy	For working with arrays
5.	pandas	For data manipulation and analysis

5.2.2 Dataset and Procedure

In order to build and train a classifier it was imperative to obtain an appropriate dataset. We used Donate-A-Cry corpus dataset, [29] which consists of 457 child cry samples of different classes such as "hungry", "belly pain", "discomfort", "burping".

Features used : spectral flatness, spectral centroid, spectral bandwidth, spectral flux, spectral roll-off, ZCR, MFCC, LPC, STACF.

The procedure followed by us consists of the following two steps. First, we implemented signal processing to extract distinct acoustic features listed above characteristic to the audio files in the used dataset. Around 20 coefficients of both GFCC and MFCC were extracted from each audio file. Secondly, we used a classifier to assign each cry to one of labelled reasons for crying babies as given in the dataset. Then we trained and tested real cries recorded for this purpose in Android and iOS phones. First the feasibility of all acoustic features was tested individually, and based on that implementation, features giving the best results were grouped together. The models were again trained using these best features collectively and the results were recorded.

Models used:

1. Support Vector Machine (SVM)

2. Multiclass Logistic Regression (One vs All) (LR)

3. K Nearest Neighbour Classifier (KNN)

4. Random Forest Classifier (RF)

The dataset used in this case was rather skewed and had majority of samples belonging to the “hungry” category. Due to this, the feasibility of Deep Learning models was not explored for this case study.

5.3 Case Study 2 – Bird Classification

5.3.1 Resource Requirements

Software requirements:

Language – Python 3

Table 5.2: Software libraries’ description

Sr. No.	Library	Description
1.	librosa	For audio analysis – used for features
2.	sklearn	For Machine learning models
3.	spafe	For GFCC extraction
4.	keras	For developing deep learning model
5.	tensorflow	For training and inference of neural networks
6.	numpy	For working with arrays
7.	pandas	For data manipulation and analysis
8.	opencv	For image processing

5.3.2 Datasets and Procedure

Dataset 0:

Dataset: Labelled dataset of bird sounds from country 'Poland' was used. It consisted of total 28 bird species with 1405 audios of bird sounds, each audio of variable duration which ranges from 8 seconds to 10 minutes.

Link: https://www.xeno-canto.org/

Features used : spectral flatness, spectral centroid, spectral bandwidth, spectral flux, spectral roll-off, ZCR, MFCC, LPC, STACF.

METHOD 1 - ML Models on whole dataset

Models used:

1. Support Vector Machine (SVM)

2. Multiclass Logistic Regression (One vs All) (LR)

3. K Nearest Neighbour Classifier (KNN)

4. Random Forest Classifier (RF)

METHOD 2 – CNN

5 bird species ('Alaudaarvensis', 'Chlorischloris', 'Emberizacitrinella', 'Erithacusrubecula', 'Fringillacoelebs') were used from the same dataset

Silence part from each audio was removed.

Resultant audio was split in the audios of 5 seconds each.

After these preprocessing, we had total 1408 audios from all 5 classes with each class having roughly same number of audios making it 'balanced dataset'.

Mel spectrogram of each audio was used as an input image to CNN

Dataset 1:

Dataset: We used dataset from https://www.kaggle.com/vinayshanbhag/bird-song-data-set which consists of total 9107 audio files includes only "songs" from 5 species- Bewick's Wren (bewickii), Northern Cardinal (cardinalis), American Robin (migratorius), Song Sparrow (melodia), Northern Mockingbird (polyglottos). The audio files are of 6-11 minutes.

Features used: spectral flatness, spectral centroid, spectral bandwidth, spectral flux, spectral roll-off, ZCR, MFCC, LPC, STACF.

METHOD 1 - UNSUPERVISED LEARNING

Models used:

1. K-Means Clustering
2. GMM Clustering

METHOD 2 - SUPERVISED LEARNING

Models used:

1. Support Vector Machine (SVM)
2. Multiclass Logistic Regression (One vs All) (LR)
3. K Nearest Neighbour Classifier (KNN)
4. Random Forest Classifier (RF)

Dataset 2:

Dataset: This dataset was from https://www.kaggle.com/rtatman/british-birdsong-dataset and it comprised of around 270 files from 88 bird species. The audio files were around 3-4 minutes long. They were first converted to .wav format and then subsequently split. In the end, there were 2265 files each with a duration of 7 seconds.

Features used: spectral flatness, spectral centroid, spectral bandwidth, spectral flux, spectral roll-off, ZCR, MFCC, STACF.

METHOD 1 - UNSUPERVISED LEARNING

Models used:

1. K-Means Clustering
2. GMM Clustering

METHOD 2 - SUPERVISED LEARNING

PART 1 – MACHINE LEARNING

Models used:

1. Support Vector Machine (SVM)
2. Multiclass Logistic Regression (One vs All) (LR)
3. K Nearest Neighbour Classifier (KNN)
4. Random Forest Classifier (RF)

PART 2 – DEEP LEARNING

Silence part from each audio was removed via preprocessing.

Resultant audio was split in the audios of 7 seconds each.

After these preprocessing, we had total 2772 audios with each class having roughly same number of audios making it 'balanced dataset'. The data was split into training and validation dataset in a ratio of 3:1.(75% training, 25% validation)

Mel spectrogram of each audio was used as an input image to CNN.

EfficientNet (a pretrained network) was used for the classification, with the number of epochs being 30 and the number of validation steps being 60.

6. Results and Discussion

6.1 Case Study 1 – Child Cry Classification

6.1.1 Supervised Learning

Table 6.1: Results of individual features and best model

FEATURES	ACCURACY	F1-SCORE	RECALL	PRECISION	BEST MODEL
MFCC	**84%**	**76%**	**84%**	**70%**	**RF**
Sp. Flatness	82%	76%	80%	71%	RF
GTCC	**84%**	**80%**	**84%**	**76%**	**RF**
STACF	84%	76%	84%	70%	KNN
ZCR	**83%**	**76%**	**83%**	**70%**	**KNN**
LPC	83%	77%	83%	72%	KNN
Sp. Roll-off	82%	75%	82%	70%	RF
Sp. Centroid	80%	75%	80%	70%	KNN
Sp. Flux	79%	78%	79%	77%	RF
Bandwidth	84%	74%	81%	70%	KNN

Table 6.2: Results for best 3 features combined

MODEL	ACCURACY	F1 - SCORE	RECALL	PRECISION
RF	84 %	76 %	84 %	70 %
KNN	82 %	77 %	82 %	76 %
SVM	71 %	72 %	71 %	75 %
LR	42 %	53 %	42 %	74 %

Best features: GTCC, MFCC, ZCR (based on accuracy and F1 score)

We first extracted individual features of the audio signals and judged the performance of each model on the basis of each individual feature. All the performance parameters were calculated.

K nearest neighbors (KNN) and Random Forest provided promising results in classification. The accuracy all the features was around 80- 85%.

Taking an aggregate of other performance parameters, we shortlisted 3 main features, namely MFCC, GFCC and Zero crossing rate (ZCR) and then trained our models on these combined features. In this case, the most effective classification was provided by SVM as evident from the results.

6.2 Case Study 2 – Bird Song Classification

6.2.1 Supervised Learning (Dataset 0)

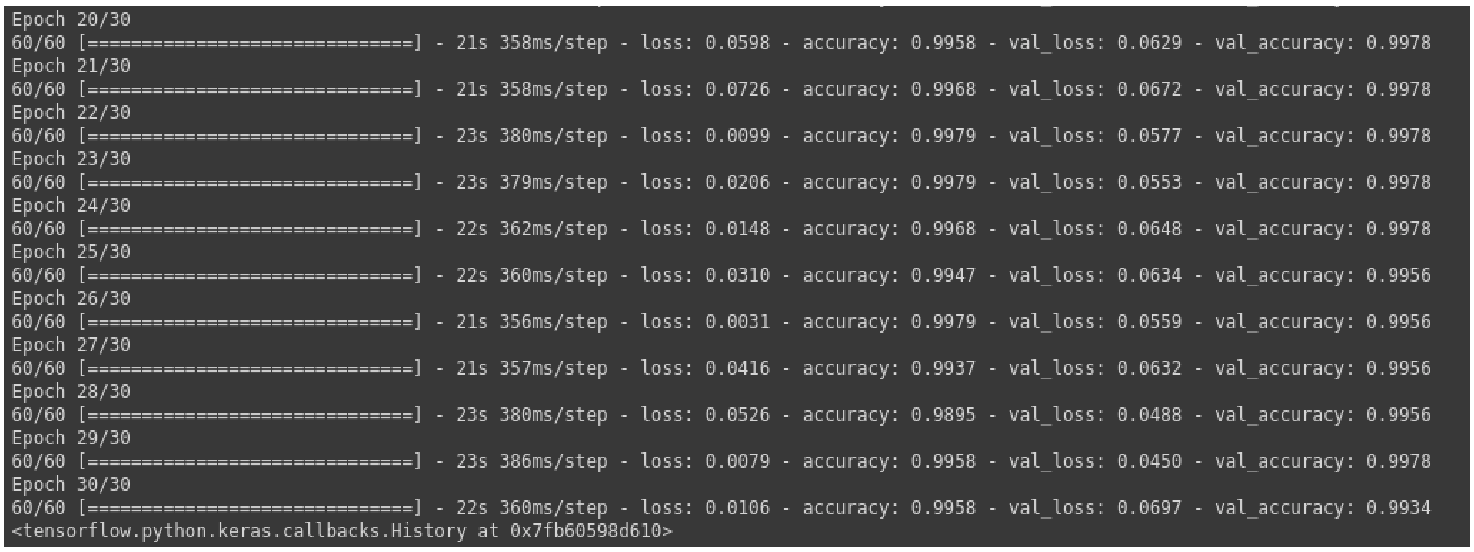
6.2.1.1 Machine Learning

Table 6.3: ML results for individual features of dataset 0

Features	LR	SVM	KNN	RF
Sp. Flatness	2 %	2 %	10 %	6 %
Sp. Centroid	2 %	2 %	7 %	8 %
Bandwidth	2 %	2 %	9 %	6 %
Sp. Flux	9 %	1 %	8 %	6 %
Sp. Roll-off (Mean)	2 %	5 %	10 %	5 %
Sp. Roll-off (Var)	1 %	2 %	7 %	7 %
ZCR (Mean)	2 %	2 %	7 %	6 %
ZCR (Variance)	4 %	2 %	8 %	6 %
STACF	0 %	3 %	9 %	2 %
MFCC	7 %	12 %	16 %	14 %
LPC (Mean)	3 %	2 %	11 %	10 %
LPC (Var)	2 %	1 %	10 %	12 %
All	13 %	12 %	17 %	20 %

6.2.1.2 Deep Learning

6.2.1.2.1 Convolutional Neural Networks (CNN)



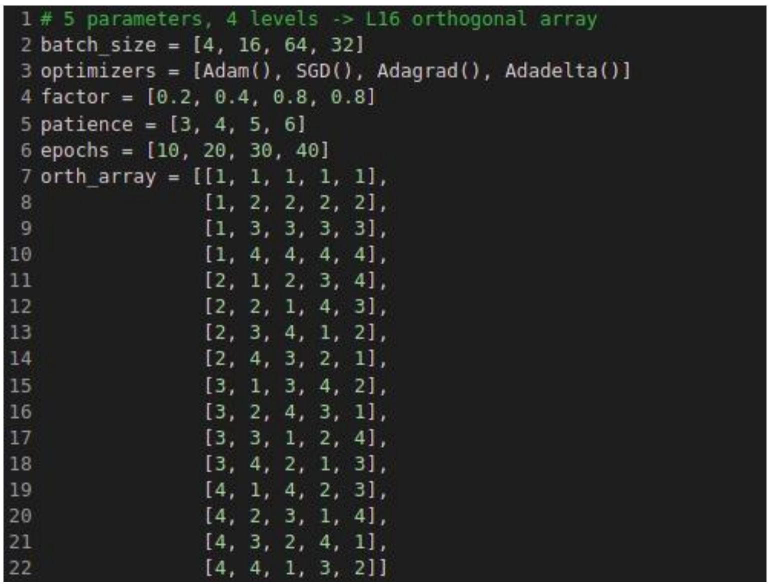
```
Epoch 20/30
60/60 [==============================] - 21s 358ms/step - loss: 0.0598 - accuracy: 0.9958 - val_loss: 0.0629 - val_accuracy: 0.9978
Epoch 21/30
60/60 [==============================] - 21s 358ms/step - loss: 0.0726 - accuracy: 0.9968 - val_loss: 0.0672 - val_accuracy: 0.9978
Epoch 22/30
60/60 [==============================] - 23s 380ms/step - loss: 0.0099 - accuracy: 0.9979 - val_loss: 0.0577 - val_accuracy: 0.9978
Epoch 23/30
60/60 [==============================] - 23s 379ms/step - loss: 0.0206 - accuracy: 0.9979 - val_loss: 0.0553 - val_accuracy: 0.9978
Epoch 24/30
60/60 [==============================] - 22s 362ms/step - loss: 0.0148 - accuracy: 0.9968 - val_loss: 0.0648 - val_accuracy: 0.9978
Epoch 25/30
60/60 [==============================] - 22s 360ms/step - loss: 0.0310 - accuracy: 0.9947 - val_loss: 0.0634 - val_accuracy: 0.9956
Epoch 26/30
60/60 [==============================] - 21s 356ms/step - loss: 0.0031 - accuracy: 0.9979 - val_loss: 0.0559 - val_accuracy: 0.9956
Epoch 27/30
60/60 [==============================] - 21s 357ms/step - loss: 0.0416 - accuracy: 0.9937 - val_loss: 0.0632 - val_accuracy: 0.9956
Epoch 28/30
60/60 [==============================] - 23s 380ms/step - loss: 0.0526 - accuracy: 0.9895 - val_loss: 0.0488 - val_accuracy: 0.9956
Epoch 29/30
60/60 [==============================] - 23s 386ms/step - loss: 0.0079 - accuracy: 0.9958 - val_loss: 0.0450 - val_accuracy: 0.9978
Epoch 30/30
60/60 [==============================] - 22s 360ms/step - loss: 0.0106 - accuracy: 0.9958 - val_loss: 0.0697 - val_accuracy: 0.9934
<tensorflow.python.keras.callbacks.History at 0x7fb60598d610>
```

Fig. 6.1: CNN results for dataset 0

- Initially after few epochs, the validation accuracy was around 78% and validation loss was around 6.5
- After 30 epochs, validation accuracy = 99.34% and validation loss = 0.0697

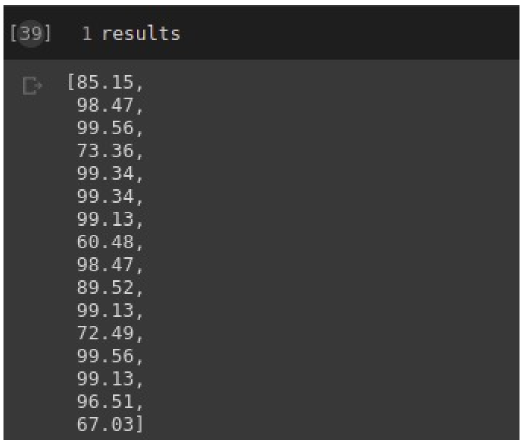
6.2.1.2.2 Results using Taguchi Method

Applying Taguchi-Method to tune the Hyperparameters:



```
# 5 parameters, 4 levels -> L16 orthogonal array
batch_size = [4, 16, 64, 32]
optimizers = [Adam(), SGD(), Adagrad(), Adadelta()]
factor = [0.2, 0.4, 0.8, 0.8]
patience = [3, 4, 5, 6]
epochs = [10, 20, 30, 40]
orth_array = [[1, 1, 1, 1, 1],
              [1, 2, 2, 2, 2],
              [1, 3, 3, 3, 3],
              [1, 4, 4, 4, 4],
              [2, 1, 2, 3, 4],
              [2, 2, 1, 4, 3],
              [2, 3, 4, 1, 2],
              [2, 4, 3, 2, 1],
              [3, 1, 3, 4, 2],
              [3, 2, 4, 3, 1],
              [3, 3, 1, 2, 4],
              [3, 4, 2, 1, 3],
              [4, 1, 4, 2, 3],
              [4, 2, 3, 1, 4],
              [4, 3, 2, 4, 1],
              [4, 4, 1, 3, 2]]
```

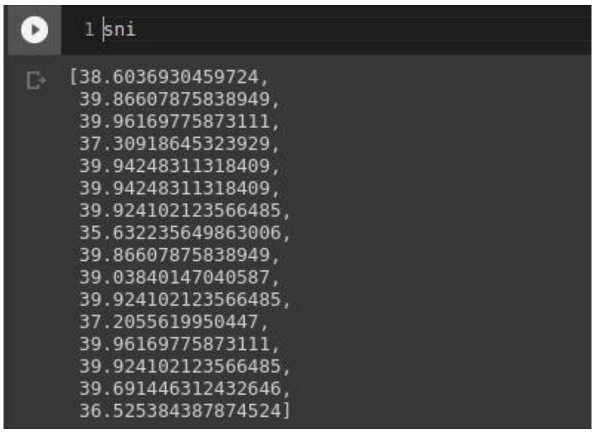
Fig. 6.2: Tuning 5 parameters using 4 levels for each parameter using L16 orthogonal array by Taguchi method



```
[39]  1 results

[85.15,
 98.47,
 99.56,
 73.36,
 99.34,
 99.34,
 99.13,
 60.48,
 98.47,
 89.52,
 99.13,
 72.49,
 99.56,
 99.13,
 96.51,
 67.03]
```

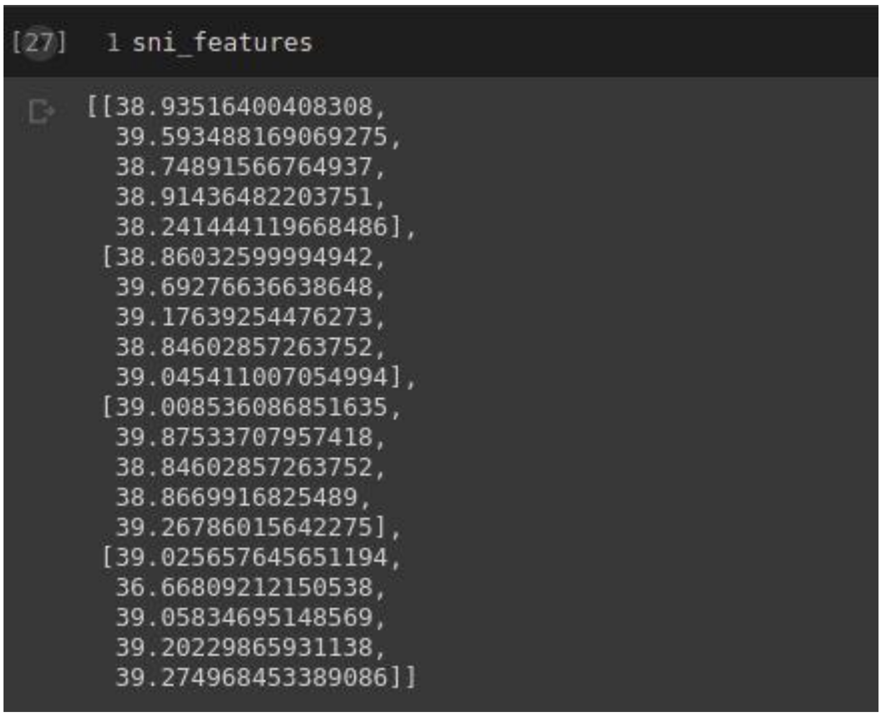
Fig. 6.3: Results of 16 experiments performed using Taguchi L16 orthogonal array



```
1 sni

[38.6036930459724,
 39.86607875838949,
 39.96169775873111,
 37.30918645323929,
 39.94248311318409,
 39.94248311318409,
 39.924102123566485,
 35.632235649863006,
 39.86607875838949,
 39.03840147040587,
 39.924102123566485,
 37.2055619950447,
 39.96169775873111,
 39.924102123566485,
 39.691446312432646,
 36.525384387874524]
```

Fig. 6.4: Larger-the-better signal to noise ratio for 16 experiments

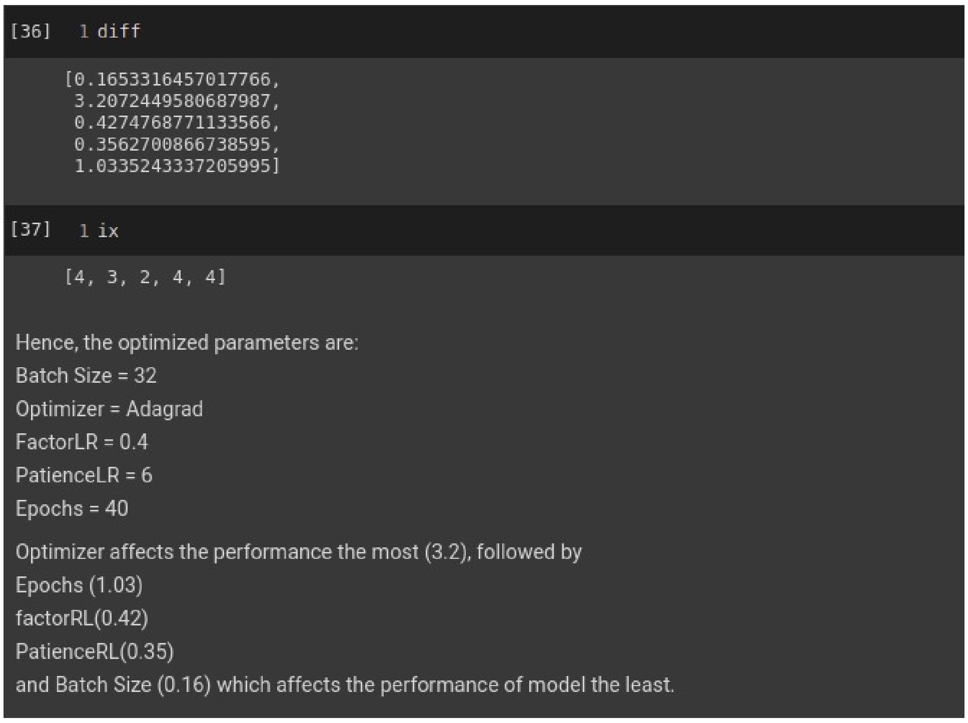


```
[27] 1 sni_features

[[38.93516400408308,
  39.593488169069275,
  38.74891566764937,
  38.91436482203751,
  38.241444119668486],
 [38.86032599994942,
  39.69276636638648,
  39.17639254476273,
  38.84602857263752,
  39.045411007054994],
 [39.008536086851635,
  39.87533707957418,
  38.84602857263752,
  38.8669916825489,
  39.26786015642275],
 [39.025657645651194,
  36.66809212150538,
  39.05834695148569,
  39.20229865931138,
  39.274968453389086]]
```

Fig. 6.5: Average SN value for each parameter and each level. Parameters are in sequence – Batch Size, Optimizer, FactorLR-(reduces learning rate by the factor - ‘FactorLR’), PatienceLR-(reduces learning rate after no progress for ‘PatienceLR’ number of epochs), Epochs (number of epochs)

After analyzing the max SN value of each parameter, its optimized value is decided. (‘ix’ list gives the index of optimized value of each feature). After analyzing the difference between max and min SN value of each parameter (R value), the importance of the hyperparameter in affecting the performance of the model is decided. Larger is the difference, larger is the effect on changing the value of corresponding parameter. ('diff' list gives the R value of all 5 parameters)



```
[36]  diff

     [0.1653316457017766,
      3.2072449580687987,
      0.4274768771133566,
      0.3562700866738595,
      1.0335243337205995]

[37]  ix

     [4, 3, 2, 4, 4]
```

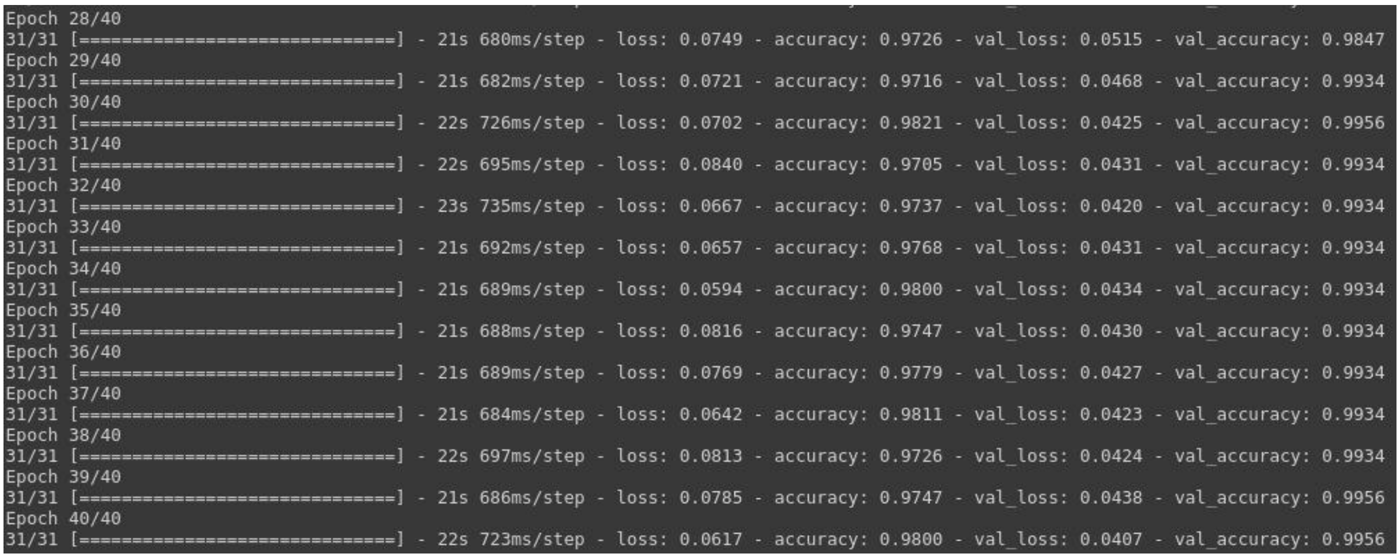
Hence, the optimized parameters are:
Batch Size = 32
Optimizer = Adagrad
FactorLR = 0.4
PatienceLR = 6
Epochs = 40

Optimizer affects the performance the most (3.2), followed by
Epochs (1.03)
factorRL(0.42)
PatienceRL(0.35)
and Batch Size (0.16) which affects the performance of model the least.

Fig. 6.6: Taguchi-Tuned Model Performance

Tuned values: Batch Size = 32, Optimizer = Adagrad, FactorLR = 0.4, PatienceLR = 6, Epochs = 40

Validation Accuracy: 99.56%, Validation Loss = 0.0407



```
Epoch 28/40
31/31 [==============================] - 21s 680ms/step - loss: 0.0749 - accuracy: 0.9726 - val_loss: 0.0515 - val_accuracy: 0.9847
Epoch 29/40
31/31 [==============================] - 21s 682ms/step - loss: 0.0721 - accuracy: 0.9716 - val_loss: 0.0468 - val_accuracy: 0.9934
Epoch 30/40
31/31 [==============================] - 22s 726ms/step - loss: 0.0702 - accuracy: 0.9821 - val_loss: 0.0425 - val_accuracy: 0.9956
Epoch 31/40
31/31 [==============================] - 22s 695ms/step - loss: 0.0840 - accuracy: 0.9705 - val_loss: 0.0431 - val_accuracy: 0.9934
Epoch 32/40
31/31 [==============================] - 23s 735ms/step - loss: 0.0667 - accuracy: 0.9737 - val_loss: 0.0420 - val_accuracy: 0.9934
Epoch 33/40
31/31 [==============================] - 21s 692ms/step - loss: 0.0657 - accuracy: 0.9768 - val_loss: 0.0431 - val_accuracy: 0.9934
Epoch 34/40
31/31 [==============================] - 21s 689ms/step - loss: 0.0594 - accuracy: 0.9800 - val_loss: 0.0434 - val_accuracy: 0.9934
Epoch 35/40
31/31 [==============================] - 21s 688ms/step - loss: 0.0816 - accuracy: 0.9747 - val_loss: 0.0430 - val_accuracy: 0.9934
Epoch 36/40
31/31 [==============================] - 21s 689ms/step - loss: 0.0769 - accuracy: 0.9779 - val_loss: 0.0427 - val_accuracy: 0.9934
Epoch 37/40
31/31 [==============================] - 21s 684ms/step - loss: 0.0642 - accuracy: 0.9811 - val_loss: 0.0423 - val_accuracy: 0.9934
Epoch 38/40
31/31 [==============================] - 22s 697ms/step - loss: 0.0813 - accuracy: 0.9726 - val_loss: 0.0424 - val_accuracy: 0.9934
Epoch 39/40
31/31 [==============================] - 21s 686ms/step - loss: 0.0785 - accuracy: 0.9747 - val_loss: 0.0438 - val_accuracy: 0.9956
Epoch 40/40
31/31 [==============================] - 22s 723ms/step - loss: 0.0617 - accuracy: 0.9800 - val_loss: 0.0407 - val_accuracy: 0.9956
```

Fig. 6.7: CNN results after tuning using Taguchi method

6.2.2 Supervised Learning (Dataset 1)

Table 6.4: ML results for individual features of dataset 1

Features	LR	SVM	KNN	RF
Sp. Flatness	24 %	24 %	25 %	24 %
Sp. Centroid	32 %	31 %	29 %	29 %
Bandwidth	28 %	29 %	23 %	22 %
Sp. Flux	28 %	30 %	28 %	27 %
Sp. Roll-off (Mean)	31 %	32 %	27 %	24 %
Sp. Roll-off (Var)	23 %	26 %	23 %	21 %
ZCR (Mean)	30 %	32 %	30 %	27 %
ZCR (Variance)	33 %	30 %	29 %	25 %
STACF	17 %	17 %	21 %	24 %
MFCC	61 %	87 %	93 %	93 %
LPC (Mean)	38 %	49 %	59 %	61 %
LPC (Var)	36 %	44 %	48 %	55 %
All	70 %	90 %	93 %	92 %

Table 6.5: ML results for combined features of dataset 1

MODEL	ACCURACY	F1 - SCORE	RECALL	PRECISION
RF	92 %	92 %	92 %	92 %
KNN	93 %	93%	93 %	93%
SVM	90 %	90 %	90 %	90 %
LR	70 %	70 %	70 %	70 %

6.2.3 Unsupervised Learning (Dataset 1)

6.2.3.1 K-Means

Table 6.5: Silhouette scores for k = 4 and k = 5 of dataset 1

Features	**Silhouette Score (k = 5)**	**Silhouette Score (k = 4)**
Sp. Flatness	**0.9862**	0.9825
Sp. Centroid	**0.5292**	0.5282
Bandwidth	**0.5270**	0.5256
Sp. Flux	**0.9216**	0.9105
Sp. Roll-off (Mean)	0.5234	**0.5293**
Sp. Roll-off (Var)	0.5461	**0.5625**
ZCR (Mean)	**0.97**	0.9642
ZCR (Variance)	**0.9983**	0.9978
STACF	0.9847	**0.988**
MFCC	0.0278	0.0714
LPC (Mean)	**0.5156**	0.4427
LPC (Var)	0.7404	**0.7591**

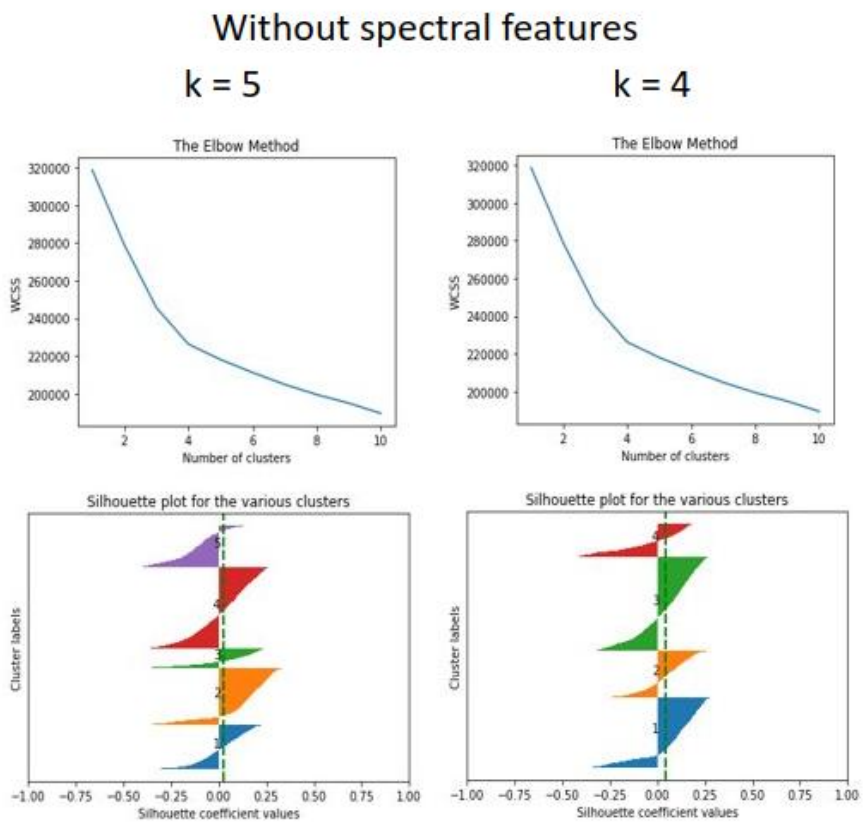


Fig. 6.8: Elbow Method and Silhouette Score plots for dataset 1 using K-Means clustering

- From the results obtained by clustering based on individual features, it is evident that k = 5 gives better results compared to k = 4.
- The features giving the most promising results are Spectral Flatness, Spectral Flux, ZCR (Mean), ZCR (Variance), STACF, LPC (Variance).
- Combining these features, we get a silhouette score of 0.4517 for k = 5, and 0.2592 for k = 4.
- K-Means gives better results for individual features and on combining the features, the accuracy worsens.

Using the best features, the results for k = 4 are worse when compared to k = 5, as expected.

6.2.3.2 Gaussian Mixture Models (GMM)

- Silhouette score checks how much the clusters are compact and well separated. The more the score is near to one, the better the clustering is!
- Using all features, we get a silhouette score of –0.23 where no. of clusters is 5 which is very poor.

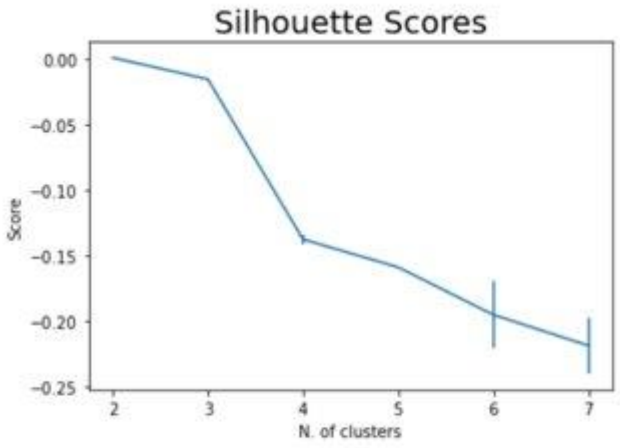


Fig. 6.9: Silhouette Scores for varying number of clusters of dataset 1

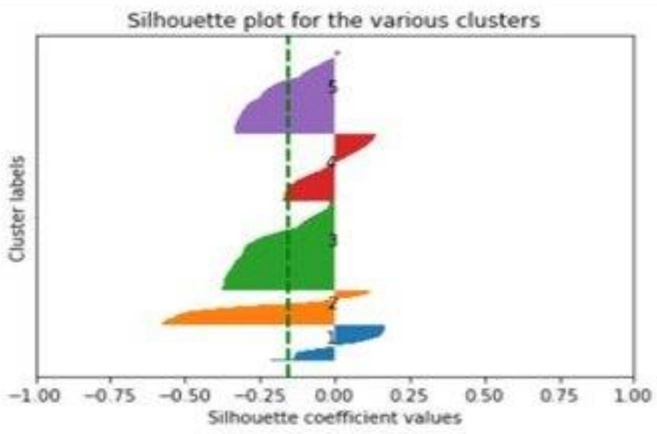


Fig. 6.10: Silhouette Score plots for dataset 1 using GMM clustering

6.2.4 Supervised Learning (Dataset 2)

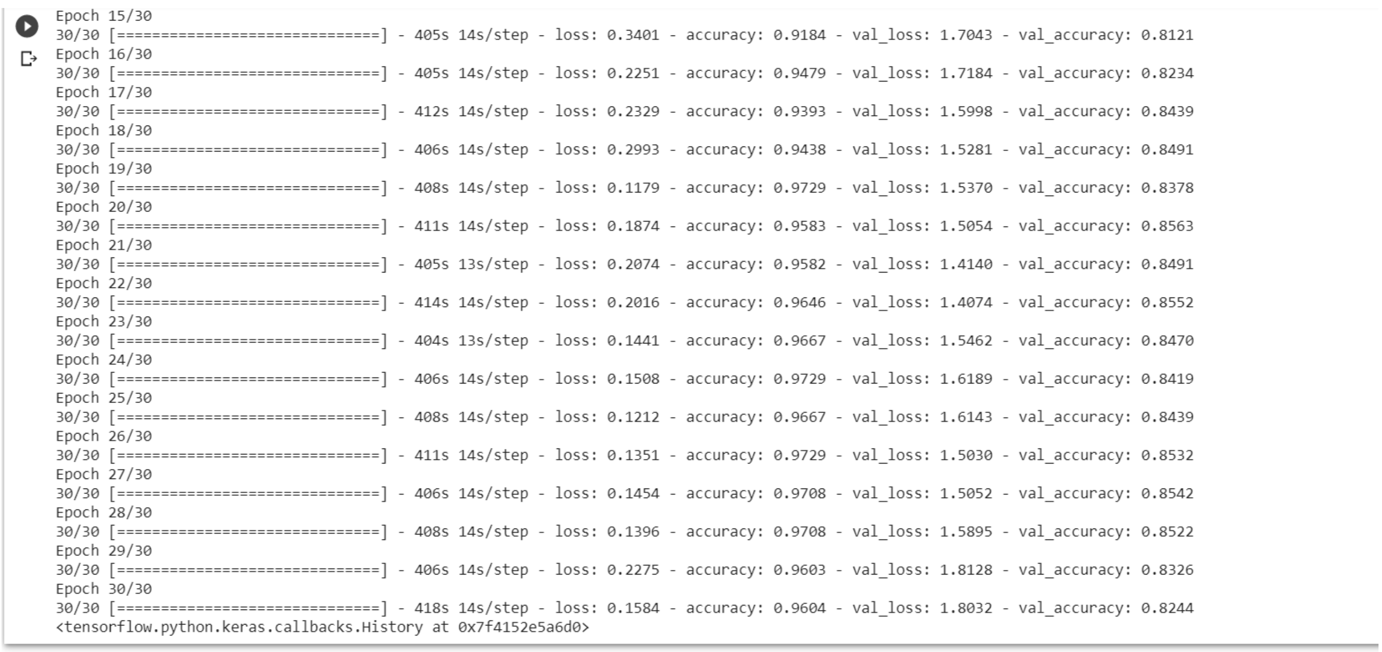
6.2.4.1 Machine Learning

Table 6.6: ML results for all features of dataset 2

MODEL	ACCURACY	F1-SCORE	RECALL	PRECISION
RF	96%	96%	96%	97%
KNN	90%	90%	90%	91%
SVM	94%	93%	94%	95%
LR	91%	91%	91%	92%

6.2.4.2 Deep Learning

6.2.4.2.1 Convolutional Neural Networks (CNN)



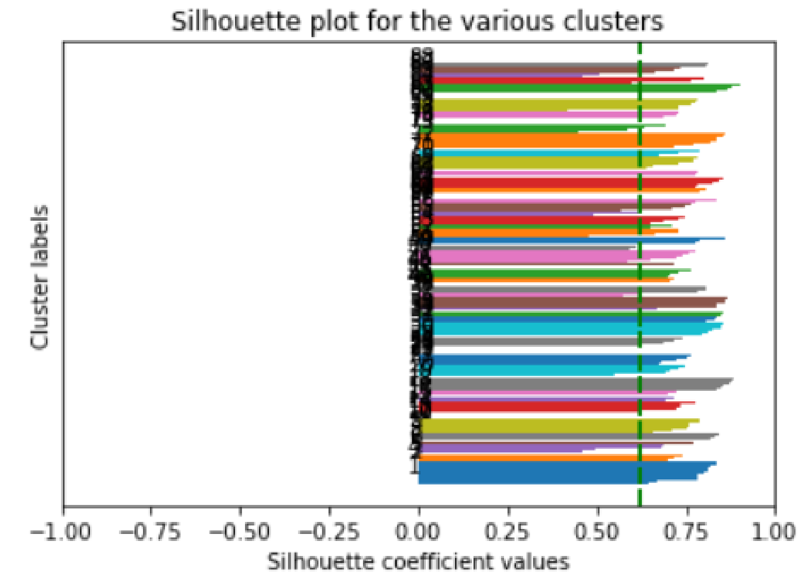
```
Epoch 15/30
30/30 [==============================] - 405s 14s/step - loss: 0.3401 - accuracy: 0.9184 - val_loss: 1.7043 - val_accuracy: 0.8121
Epoch 16/30
30/30 [==============================] - 405s 14s/step - loss: 0.2251 - accuracy: 0.9479 - val_loss: 1.7184 - val_accuracy: 0.8234
Epoch 17/30
30/30 [==============================] - 412s 14s/step - loss: 0.2329 - accuracy: 0.9393 - val_loss: 1.5998 - val_accuracy: 0.8439
Epoch 18/30
30/30 [==============================] - 406s 14s/step - loss: 0.2993 - accuracy: 0.9438 - val_loss: 1.5281 - val_accuracy: 0.8491
Epoch 19/30
30/30 [==============================] - 408s 14s/step - loss: 0.1179 - accuracy: 0.9729 - val_loss: 1.5370 - val_accuracy: 0.8378
Epoch 20/30
30/30 [==============================] - 411s 14s/step - loss: 0.1874 - accuracy: 0.9583 - val_loss: 1.5054 - val_accuracy: 0.8563
Epoch 21/30
30/30 [==============================] - 405s 13s/step - loss: 0.2074 - accuracy: 0.9582 - val_loss: 1.4140 - val_accuracy: 0.8491
Epoch 22/30
30/30 [==============================] - 414s 14s/step - loss: 0.2016 - accuracy: 0.9646 - val_loss: 1.4074 - val_accuracy: 0.8552
Epoch 23/30
30/30 [==============================] - 404s 13s/step - loss: 0.1441 - accuracy: 0.9667 - val_loss: 1.5462 - val_accuracy: 0.8470
Epoch 24/30
30/30 [==============================] - 406s 14s/step - loss: 0.1508 - accuracy: 0.9729 - val_loss: 1.6189 - val_accuracy: 0.8419
Epoch 25/30
30/30 [==============================] - 408s 14s/step - loss: 0.1212 - accuracy: 0.9667 - val_loss: 1.6143 - val_accuracy: 0.8439
Epoch 26/30
30/30 [==============================] - 411s 14s/step - loss: 0.1351 - accuracy: 0.9729 - val_loss: 1.5030 - val_accuracy: 0.8532
Epoch 27/30
30/30 [==============================] - 406s 14s/step - loss: 0.1454 - accuracy: 0.9708 - val_loss: 1.5052 - val_accuracy: 0.8542
Epoch 28/30
30/30 [==============================] - 408s 14s/step - loss: 0.1396 - accuracy: 0.9708 - val_loss: 1.5895 - val_accuracy: 0.8522
Epoch 29/30
30/30 [==============================] - 406s 14s/step - loss: 0.2275 - accuracy: 0.9603 - val_loss: 1.8128 - val_accuracy: 0.8326
Epoch 30/30
30/30 [==============================] - 418s 14s/step - loss: 0.1584 - accuracy: 0.9604 - val_loss: 1.8032 - val_accuracy: 0.8244
<tensorflow.python.keras.callbacks.History at 0x7f4152e5a6d0>
```

Fig. 6.11: CNN results for dataset 2

- Initially after eight epochs, the validation accuracy was around 70.63% and validation loss was around 2.267
- After 30 epochs, validation accuracy = 96.04% and validation loss = 0.1584

6.2.5 Unsupervised Learning (Dataset 2)

6.2.5.1 K-Means



```
0.6242404509686813
```

Fig. 6.12: Silhouette Score plots for dataset 2 using K-Means clustering

- The features giving the most promising results are Spectral Flatness, Spectral Flux, ZCR (Mean), ZCR (Variance), STACF, LPC (Variance).
- Combining these features, we get a silhouette score of 0.6242 for k = 88.

6.2.5.2 Gaussian Mixture Models (GMM)

- Silhouette score checks how much the clusters are compact and well separated. The more the score is near to one, the better the clustering is!
- Using all features, we get a silhouette score of 0.570 where no. of clusters are 88 which is not as good as we want.

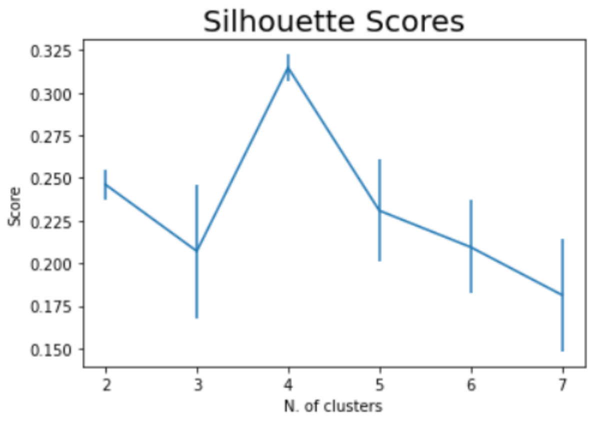


Fig. 6.13: Silhouette Scores for varying number of clusters of dataset 2

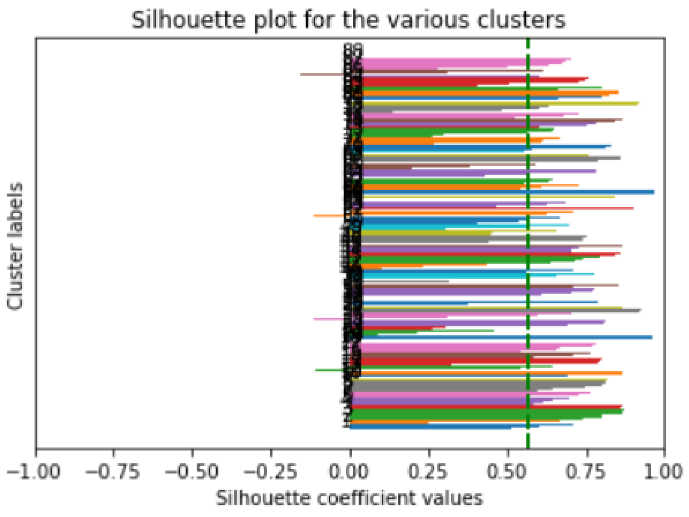


Fig. 6.14: Silhouette Score plots for dataset 2 using GMM clustering

7. Challenges

7.1 Case Study 1 – Child Cry Classification

Less number of datasets available

Less number of cry samples in available dataset

Available datasets are imbalanced

7.2 Case Study 2 – Bird Song Classification

Background Noise

Multi-label classification problem

Different types of bird calls

Inter-species variance

Dataset issues

8. Conclusion

8.1 Case Study 1 – Child Cry Classification

MFCC, GTCC and ZCR are the most efficient features for accurate classification.

GTCC is most important in classifying child cries audio signals.

Random Forest and K-Nearest Neighbors algorithms give the best results.

8.2 Case Study 2 – Bird Song Classification

Firstly, we trained ML models on the Xeno Canto dataset using individual features. The results were not satisfactory. So classification through neural networks was employed on the same dataset which yielded promising results.

We employed Taguchi method to tune the hyperparameters of CNN which resulted in parameter values such as batch size = 32, epochs = 40 etc. Using these parameters we get improved accuracy as well as lower validation loss.

A different dataset was chosen (consisting of 5 classes) and ML models were applied for combined and individual features. The results were poor for individual features, and were comparatively much better for combined features.

On the same dataset we applied K-Means clustering which gave promising results on few features such as ZCR, LPC variance, Spectral Flatness, Spectral Flux and STACF.

To check the usability of another dataset (consisting of 88 classes), we employed ML models here as well and got the best results.

Neural networks were also employed for classification on this dataset and the results were quite satisfactory in this case as well.

Using K-Means clustering we got average results. The efficacy of GMM was also tested on both datasets, however results were not satisfactory.

9. Future Work

9.1 Case Study 1 – Child Cry Classification

Concluding, the findings of our research show that certain ML models exist that perform well in classifying cries of infants. Such models could further assist in developing a screening instrument on the basis of auditory characteristics of cries. The development of such an instrument would help in detecting any pathological development earlier. These instruments would also be helpful for professions dealing with babies like babysitters, nurses, pediatricians and even parents of that very child.

9.2 Case Study 2 – Bird Song Classification

The audio files in the third dataset (consisting of 88 classes) were of real good quality with minimal silenced portion which makes this dataset a good option to work on in the future.

The suitability of other Deep Learning models in classification of bird songs will be explored.

Currently, we haven't explored the usefulness of 'Pitch' feature in classification of bird songs. This feature would be incorporated in the further training of our classification models.

10. References

1) The Study of Baby Crying Analysis Using MFCC and LFCC in Different Classification Methods S. P. Dewi, Anggunmeka Luhur Prasasti, Budhi Irawan Published 2019 Computer Science 2019 IEEE International Conference on Signals and Systems (ICSigSys)

2) L. Liu, W. Li, X. Wu and B. X. Zhou, "Infant cry language analysis and recognition: an experimental approach," in IEEE/CAA Journal of Automatica Sinica, vol. 6, no. 3, pp. 778-788, May 2019, DOI: 10.1109/JAS.2019.1911435.

3)Subramanian, Hariharan, P. Rao and D. Roy. “AUDIO SIGNAL CLASSIFICATION.” (2004).

4) Vibhute Anup. (2014). Feature Extraction Techniques in Speech Processing A Survey. International Journal of Computer Applications. 107. 1-8. 10.5120/18744-9997.

5) Jasleen and Dawood Dilber. “Feature Selection and Extraction of Audio Signal.” (2016).

6) http://www.ijirset.com/upload/2016/march/64_Feature.pdf

7) Bang, Arti & Rege, Priti. (2018). Automatic Recognition of Bird Species Using Human Factor Cepstral Coefficients.10.1007/978-981-10-5544-7_35.

8) Bang, Arti V. and P. Rege. “Classification of Bird Species based on Bioacoustics.” (2013).

9) Bang, Arti & Rege, Priti. (2017). Evaluation of various feature sets and feature selection towards automatic recognition of bird species. International Journal of Computer Applications in Technology. 56. 172. 10.1504/IJCAT.2017.088197

10) Bang, Arti & Rege, Priti. (2017). Recognition of Bird Species from their Sounds using Data Reduction Techniques.111-116. 10.1145/3154979.3155002.

11) Valero, Xavier & Alías, Francesc. (2012). Gammatone Cepstral Coefficients: Biologically Inspired Features for Non-Speech Audio Classification. Multimedia, IEEE Transactions on. 14. 1684-1689. 10.1109/TMM.2012.2199972.

12) Evaluating Gammatone Frequency Cepstral Coefficients with Neural Networks for Emotion Recognition from Speech,

13) Messaoud, Ali &Tadj, Chakib. (2010). A Cry-Based Babies Identification System. 6134. 192-199. 10.1007/978-3-642- 13681- 8_23.

14) Izmirli, Özgür. (2000). Using a Spectral Flatness Based Feature for Audio Segmentation and Retrieval.

15) Přibil, Jiří & Přibilová, Anna. (2009). Spectral Flatness Analysis for Emotional Speech Synthesis and Transformation. Lecture Notes in Computer Science. 5641. 106-115. 10.1007/978- 3-642-03320-9_11

16) https://link.springer.com/chapter/10.1007/978-3-642- 03320- 9_11

17) Auditory-model based robust feature selection for speech recognition .The Journal of the Acoustical Society of America 127, EL73 (2010), Christos Koniaris, Marcin Kuropatwinski, W. Bastiaan Kleijn.

18) https://towardsdatascience.com/featureselection-using- random-forest-26d7b747597f

19)https://www.researchgate.net/post/How_to_do_support_vector_machine_based_feature_variable_selection

20) Lin, Cheng-Jian & Shiou-Yun, Jeng & Chen, Mei-Kuei. (2020). Using 2D CNN with Taguchi Parametric Optimization for Lung Cancer Recognition from CT Images. Applied Sciences. 10. 2591. 10.3390/app10072591.

21)https://www.javatpoint.com/machine-learning-supportvector-machine-algorithm

22)
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_knn_algorithm_finding_nearest_neighbors.htm

23)
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_random_forest.htm

24)
https://www.tutorialspoint.com/machine_learning_with_python/machine_learning_with_python_classification_algorithms_logistic_regression.htm

25)Rashmika Patole and Priti Rege. Acoustic Classification of Bird Species

26) Fuhr, Tanja, H. Reetz and C. Wegener. "Comparison of Supervised-learning Models for Infant Cry Classification / Vergleich von Klassifikations modellenzur Säuglingsschrei analyse." International Journal of Health Professions 2 (2015): 15 – 4.

27) Cano O, Sergio & Escobedo Beceiro, Daniel &Ekkel, Taco. (2004). A Radial Basis Function Network Oriented for Infant Cry Classification. Progress in Pattern Recognition, Image Analysis and Applications. Vol. 3287 of Lecture Notes in Computer Science. 3287. 374-380. 10.1007/978-3-540- 30463-0_46.

28) https://musicinformationretrieval.com/spectral_features.html

29) https://github.com/gveres/donateacry-corpuS

30) https://www.kaggle.com/vinayshanbhag/bird-song-data-set

31) https://www.xeno-canto.org/

32) https://towardsdatascience.com/k-means-clustering-for-beginners-ea2256154109

33) https://en.wikipedia.org/wiki/Determining_the_number_of_clusters_in_a_data_set

34) https://towardsdatascience.com/how-to-evaluate-unsupervised-learning-models-3aa85bd98aa2

35) https://towardsdatascience.com/a-comprehensive-guide-to-convolutional-neural-networks-the-eli5- way-3bd2b1164a53

36) https://ai.googleblog.com/2019/05/efficientnet-improving-accuracy-and.html

37) https://towardsdatascience.com/sound-based-bird-classification-965d0ecacb2b

38) https://www.kaggle.com/rtatman/british-birdsong-dataset

PUBLICATIONS BY THE CANDIDATES:

Our team wrote a paper on Case Study 1 as demonstrated in this report under the guidance of Dr. Rege Ma'am. It was selected for presentation at the conference I2CT 2021 (6th International Conference for Convergence in Technology) held from 2 - 4 April 2021. Our paper is titled "Child Cry Classification - An analysis of features and models". The paper will soon be published by IEEE Xplore.

11.Appendix

11.1 Abbreviations

- WHO – World Health Organization
- AI – Artificial Intelligence
- ML – Machine Learning
- DL – Deep Learning
- LDA – Linear Discriminant Analysis
- ANN – Artificial Neural Networks
- PNN – Probabilistic Neural Networks
- HFCC – Human Factor Cepstral Coefficients
- LPF – Low Pass Filter
- RBF – Radial Basis Function
- ERB – Equivalent Rectangular Bandwidth
- Fc – Cutoff Frequency
- WPD – Wavelet Packet Decomposition
- DTW – Dynamic Time Warping
- PCA – Principal Component Analysis
- LBG – Linde-Buzo-Gray Algorithm
- MFCC – Mel Frequency Cepstral Coefficients
- SC – Spectral Centroid
- SF – Spectral Flux
- BW – Bandwidth
- SRF – Spectral Rolloff
- GTCC/ GFCC – Gammatone Frequency Cepstral Coefficients
- STACF – Short Time Auto-Correlation Coefficients
- ZCR – Zero Crossing Rate
- LPC – Linear Predictive Coefficients
- LR – Logistic Regression
- SVM – Support Vector Machine
- KNN – K-Nearest Neighbours Classifier
- RF – Random Forest Classifier
- TN – True Negative
- TP – True Positive

- FN – False Negative
- FP – False Positive
- CNN – Convolutional Neural Networks
- GMM – Gaussian Mixture Model

11.2 List of figures

11.3 List of equations

11.4 List of tables

11.5 Complete list of python libraries

absl-py==0.12.0

alabaster==0.7.12

albumentations==0.1.12

altair==4.1.0

anaconda-client==1.7.2

anaconda-navigator==1.10.0

anaconda-project==0.8.3

appdirs==1.4.4

argh==0.26.2

argon2-cffi==20.1.0

asn1crypto==1.3.0

astor==0.8.1

astroid==2.3.3

astropy==4.2.1

astunparse==1.6.3

async-generator==1.10

atari-py==0.2.6

atomicwrites==1.4.0

attrs==20.3.0

audioread==2.1.9

autograd==1.3

autopep8==1.4.4

Babel==2.9.0

backcall==0.2.0

backports.functools-lru-cache==1.6.1

backports.shutil-get-terminal-size==1.0.0

backports.tempfile==1.0

backports.weakref==1.0.post1

bcrypt

beautifulsoup4==4.8.2

bitarray==1.2.1

bkcharts==0.2

bleach==3.3.0

blis==0.4.1

bokeh==2.3.1

boto==2.49.0

Bottleneck==1.3.2

```
branca==0.4.2
brotlipy==0.7.0
bs4==0.0.1
CacheControl==0.12.6
cachetools==4.2.2
catalogue==1.0.0
certifi==2020.12.5
cffi==1.14.5
chainer==7.4.0
chardet==3.0.4
click==7.1.2
cloudpickle==1.3.0
clyent==1.2.2
cmake==3.12.0
cmdstanpy==0.9.5
colorama==0.4.3
colorcet==2.0.6
colorlover==0.3.0
community==1.0.0b1
comtypes==1.1.7
conda-build==3.20.5
conda-package-handling==1.6.0
conda-verify==3.4.2
conda==4.10.1
contextlib2==0.6.0
convertdate==2.3.2
corner==2.2.1
coverage==3.7.1
coveralls==0.5
crcmod==1.7
cryptography==2.8
cufflinks==0.17.3
cupy-cuda101==7.4.0
cvxopt==1.2.6
cvxpy==1.0.31
cycler==0.10.0
cymem==2.0.5
Cython==0.29.22
cytoolz==0.11.0
daft==0.0.4
dask==2.12.0
datascience==0.10.6
debugpy==1.0.0
decorator==4.4.2
defusedxml==0.7.1
descartes==1.1.0
diff-match-patch==20181111
dill==0.3.3
distlib==0.3.0
distributed==2.11.0
distro==1.4.0
dlib==19.18.0
dm-tree==0.1.6
docopt==0.6.2
docutils==0.17
dopamine-rl==1.0.5
earthengine-api==0.1.260
easydict==1.9
ecos==2.0.7.post1
editdistance==0.5.3
efficientnet==1.0.0
en-core-web-sm==2.2.5
entrypoints==0.3
ephem==3.7.7.1
et-xmlfile==1.0.1
fa2==0.3.5
fancyimpute==0.4.3
fastai==1.0.61
fastcache==1.1.0
fastdtw==0.3.4
fastprogress==1.0.0
fastrlock==0.6
fbprophet==0.7.1
feather-format==0.4.1
ffprobe==0.5
filelock==3.0.12
firebase-admin==4.4.0
fix-yahoo-finance==0.0.22
flake8==3.7.9
Flask==1.1.2
flatbuffers==1.12
folium==0.8.3
fsspec==0.6.2
future==0.18.2
gast==0.3.3
GDAL==2.2.2
gdown==3.6.4
gensim==3.6.0
geographiclib==1.50
geopy==1.17.0
gevent==1.4.0
gin-config==0.4.0
glob2==0.7
gmpy2==2.0.8
google-api-core==1.26.3
google-api-python-client==1.12.8
google-auth-httplib2==0.0.4
```

```
google-auth-oauthlib==0.4.4
google-auth==1.30.0
google-cloud-bigquery-storage==1.1.0
google-cloud-bigquery==1.21.0
google-cloud-core==1.0.3
google-cloud-datastore==1.8.0
google-cloud-firestore==1.7.0
google-cloud-language==1.2.0
google-cloud-storage==1.18.1
google-cloud-translate==1.5.0
google-colab==1.0.0
google-pasta==0.2.0
google-resumable-media==0.4.1
google==2.0.3
googleapis-common-protos==1.53.0
googledrivedownloader==0.4
graphviz==0.10.1
greenlet==1.0.0
grpcio==1.32.0
gspread-dataframe==3.0.8
gspread==3.0.1
gym==0.17.3
h5py==2.10.0
HeapDict==1.0.1
hijri-converter==2.1.1
holidays==0.10.5.2
holoviews==1.14.3
html5lib==1.0.1
httpimport==0.5.18
httplib2==0.17.4
httplib2shim==0.0.3
humanize==0.5.1
hyperopt==0.1.2
hypothesis==5.5.4
ideep4py==2.0.0.post3
idna==2.10
image-classifiers==1.0.0
imageio==2.6.1
imagesize==1.2.0
imbalanced-learn==0.4.3
imblearn==0.0
imgaug==0.2.9
importlib-metadata==3.10.1
importlib-resources==5.1.2
imutils==0.5.4
inflect==2.1.0
iniconfig==1.1.1
intel-openmp==2021.2.0
intervaltree==3.0.2
ipaddr==2.2.0
ipykernel==5.4.3
ipython
ipython-genutils==0.2.0
ipython-sql==0.3.9
ipython==7.19.0
ipywidgets==7.6.3
isort==4.3.21
itsdangerous==1.1.0
jax==0.2.12
jaxlib==0.1.65+cuda110
jdcal==1.4.1
jedi==0.18.0
jeepney==0.4.2
jieba==0.42.1
Jinja2==2.11.3
joblib==1.0.1
jpeg4py==0.1.4
json5==0.9.5
jsonschema==3.2.0
jupyter-client==6.1.11
jupyter-console==6.2.0
jupyter-core==4.7.1
jupyter==1.0.0
jupyterlab-pygments==0.1.2
jupyterlab-server==1.0.6
jupyterlab-widgets==1.0.0
jupyterlab==2.2.6
kaggle==1.5.12
kapre==0.1.3.1
Keras-Applications==1.0.8
Keras-Preprocessing==1.1.2
keras-vis==0.4.1
Keras==2.3.1
keyring==21.1.0
kiwisolver==1.3.1
knnimpute==0.1.0
korean-lunar-calendar==0.2.1
lazy-object-proxy==1.4.3
libarchive-c==2.9
librosa==0.8.0
lief==0.9.0
lightgbm==2.2.3
llvmlite==0.35.0
lmdb==0.99
locket==0.2.0
```

```
lockfile==0.12.2
LunarCalendar==0.0.9
lxml==4.5.0
Markdown==3.3.4
MarkupSafe==1.1.1
matplotlib-inline==0.1.2
matplotlib-venn==0.11.6
matplotlib==3.3.4
mccabe==0.6.1
menuinst==1.4.16
missingno==0.4.2
mistune==0.8.4
mizani==0.6.0
mkl-fft==1.2.0
mkl-random==1.1.1
mkl-service==2.3.0
mkl==2019.0
mlxtend==0.18.0
mock==4.0.2
more-itertools==8.7.0
moviepy==0.2.3.5
mpmath==1.2.1
msgpack==1.0.2
multipledispatch==0.6.0
multiprocess==0.70.11.1
multitasking==0.0.9
murmurhash==1.0.5
music21==5.5.0
natsort==5.5.0
navigator-updater==0.2.1
nbclient==0.5.3
nbconvert==6.0.7
nbformat==5.1.3
nest-asyncio==1.5.1
networkx==2.5.1
nibabel==3.0.2
nltk==3.4.5
nose==1.3.7
notebook==6.2.0
np-utils==0.5.12.1
numba==0.52.0
numexpr==2.7.3
numpy==1.19.5
numpydoc==0.9.2
nvidia-ml-py3==7.352.0
oauth2client==4.1.3
oauthlib==3.1.0
okgrade==0.4.3
olefile==0.46
opencv-contrib-
python==4.1.2.30
opencv-python==4.1.2.30
openpyxl==3.0.7
opt-einsum==3.3.0
osqp==0.6.2.post0
packaging==20.9
palettable==3.3.0
pandas-datareader==0.9.0
pandas-gbq==0.13.3
pandas-profiling==1.4.1
pandas==1.2.1
pandocfilters==1.4.3
panel==0.11.2
param==1.10.1
paramiko
partd==1.1.0
path==13.1.0
pathlib==1.0.1
pathlib2==2.3.5
pathtools==0.1.2
patsy==0.5.1
pep517==0.8.2
pep8==1.7.1
pexpect==4.8.0
pickleshare==0.7.5
Pillow==8.1.0
pip-autoremove==0.9.1
pip-tools==4.5.1
pkginfo==1.6.1
plac==1.1.3
plotly==4.4.1
plotnine==0.6.0
pluggy==0.13.1
ply==3.11
pooch==1.3.0
portpicker==1.3.1
prefetch-generator==1.0.1
preshed==3.0.5
prettytable==2.1.0
progress==1.5
progressbar2==3.38.0
prometheus-client==0.10.1
promise==2.3
prompt-toolkit==3.0.14
protobuf==3.16.0
psutil==5.6.7
psycopg2==2.7.6.1
ptyprocess==0.7.0
py==1.10.0
```

```
pyarrow==3.0.0
pyasn1-modules==0.2.8
pyasn1==0.4.8
pycocotools==2.0.2
pycodestyle==2.6.0
pycosat==0.6.3
pycparser==2.20
pycrypto==2.6.1
pyct==0.4.8
pycurl==7.43.0.6
pydata-google-auth==1.1.0
pydocstyle==4.0.1
pydot-ng==2.0.0
pydot==1.4.2
pydotplus==2.0.2
PyDrive==1.3.1
pydub==0.24.1
pyemd==0.5.1
pyerfa==1.7.2
pyflakes==2.2.0
pyglet==1.5.0
Pygments==2.7.4
pygobject==3.26.1
pylint==2.4.4
pymc3==3.7
PyMeeus==0.5.11
pymongo==3.11.3
pymystem3==0.2.0
PyNaCl
pyodbc===4.0.0-unsupported
PyOpenGL==3.1.5
pyOpenSSL==19.1.0
pyparsing==2.4.7
pyreadline==2.1
pyrsistent==0.17.3
pysndfile==1.3.8
PySocks==1.7.1
pystan==2.19.1.1
pytest-arraydiff==0.3
pytest-astropy-header==0.1.2
pytest-astropy==0.8.0
pytest-doctestplus==0.5.0
pytest-openfiles==0.4.0
pytest-remotedata==0.3.2
pytest==5.3.5
python-apt==0.0.0
python-chess==0.23.11
python-dateutil==2.8.1
python-jsonrpc-server==0.3.4
python-language-
server==0.31.7
python-louvain==0.15
python-slugify==4.0.1
python-utils==2.5.6
pytoml==0.1.21
pytz==2020.5
pyviz-comms==2.0.1
PyWavelets==1.1.1
pywin32-ctypes==0.2.0
pywin32==227
pywinpty==0.5.7
pyxdg==0.26
PyYAML==5.3.1
pyzmq==22.0.3
q==2.6
QDarkStyle==2.8.1
qdldl==0.1.5.post0
QtAwesome==0.6.1
qtconsole==5.1.0
QtPy==1.9.0
read==0.0.2
regex==2019.12.20
requests-oauthlib==1.3.0
requests==2.23.0
resampy==0.2.2
retrying==1.3.3
rope==0.16.0
rpy2==3.4.3
rsa==4.7.2
Rtree==0.9.4
ruamel-yaml==0.15.87
scikit-image==0.17.2
scikit-learn==0.24.1
scipy==1.6.0
screen-resolution-
extra==0.0.0
scs==2.1.3
seaborn==0.11.1
SecretStorage==3.1.2
segmentation-models==1.0.1
Send2Trash==1.5.0
setuptools-git==1.2
Shapely==1.7.1
simplegeneric==0.8.1
singledispatch==3.4.0.3
sip==4.19.13
six==1.15.0
sklearn-pandas==1.8.0
sklearn==0.0
```

```
smart-open==5.0.0
snowballstemmer==2.1.0
sortedcollections==1.2.1
sortedcontainers==2.3.0
SoundFile==0.10.3.post1
soupsieve==2.0.1
spacy==2.2.4
spafe==0.1.2
Sphinx==2.4.0
sphinxcontrib-applehelp==1.0.2
sphinxcontrib-devhelp==1.0.2
sphinxcontrib-htmlhelp==1.0.3
sphinxcontrib-jsmath==1.0.1
sphinxcontrib-qthelp==1.0.3
sphinxcontrib-serializinghtml==1.1.4
sphinxcontrib-websupport==1.2.4
spyder-kernels==1.8.1
spyder==4.0.1
SQLAlchemy==1.4.7
sqlparse==0.4.1
srsly==1.0.5
statsmodels==0.12.0
sympy==1.7.1
tables==3.6.1
tabulate==0.8.9
tblib==1.7.0
tensorboard-data-server==0.6.1
tensorboard-plugin-wit==1.8.0
tensorboard==2.5.0
tensorflow-datasets==4.0.1
tensorflow-estimator==2.4.0
tensorflow-gcs-config==2.4.0
tensorflow-hub==0.12.0
tensorflow-metadata==0.29.0
tensorflow-probability==0.12.1
tensorflow==2.1.0
termcolor==1.1.0
terminado==0.9.4
testpath==0.4.4
text-unidecode==1.3
textblob==0.15.3
textgenrnn==1.4.1
Theano==1.0.5
thinc==7.4.0
threadpoolctl==2.1.0
tifffile==2021.4.8
toml==0.10.2
toolz==0.11.1
torch==1.8.1+cu101
torchsummary==1.5.1
torchtext==0.9.1
torchvision==0.9.1
tornado==6.1
tqdm==4.42.1
traitlets==5.0.5
tweepy==3.10.0
typeguard==2.7.1
typing-extensions==3.7.4.3
tzlocal==1.5.1
ujson==1.35
unicodecsv==0.14.1
uritemplate==3.0.1
urllib3==1.25.8
vega-datasets==0.9.0
wasabi==0.8.2
watchdog==0.10.2
wavio==0.0.4
wcwidth==0.2.5
webencodings==0.5.1
Werkzeug==1.0.1
widgetsnbextension==3.5.1
win-inet-pton==1.1.0
win-unicode-console==0.5
wincertstore==0.2
wordcloud==1.5.0
wrapt==1.12.1
wurlitzer==2.0.0
xarray==0.1.1
xgboost==0.90
xkit==0.0.0
xlrd==2.0.1
XlsxWriter==1.2.7
xlwings==0.20.8
xlwt==1.3.0
xmltodict==0.12.0
yapf==0.28.0
yellowbrick==0.9.1
zict==2.0.0
zipp==3.4.1
zope.event==4.5.0
zope.interface
```