IHEAR – LIGHTWEIGHT MACHINE LEARNING ENGINE WITH CONTEXT AWARE AUDIO RECOGNITION MODEL

A THESIS IN
Computer Science

Presented to the Faculty of the University
Of Missouri-Kansas City in partial fulfillment
Of the requirements for the degree

MASTER OF SCIENCE

By
GURU TEJA MANNAVA

B.Tech, Jawaharlal Nehru Technological University – Hyderabad, India, 2013

Kansas City, Missouri
2016

IHEAR – LIGHTWEIGHT MACHINE LEARNING ENGINE WITH CONTEXT AWARE AUDIO RECOGNITION MODEL

Guru Teja Mannava, Candidate for the Master of Science Degree

University of Missouri-Kansas City, 2016

ABSTRACT

With the increasing popularity and affordability of smartphones, there is a high demand to add machine-learning engines to smartphones. However, Machine Learning with smartphones is typically not feasible due to the heavy loaded computation required for processing large-scale data with Machine Learning. The conventional Machine Learning systems do not naturally or efficiently support some very important features for large-scale stream data.

To overcome these limitations, we propose the iHear engine that aims to support lightweight Machine Learning through a collaboration between cloud and smartphones. The contributions of this thesis are summarized as follows:

1) The iHear system architecture for achieving high performance with parallel and distributed learning by separating cloud-based learning from smartphone-based recognition
2) The context-aware model for improvement of the accuracy and efficiency in audio recognition and sound enhancement
3) Audio recognition with real-time data preserving data consistency.
4) An intelligent hearing app for IOS devices developed for effective and dynamic audio recognition and enhancement depending upon users’ context for providing better hearing experiences.

The efficiency and effectiveness of the iHear engine in terms of its continuous learning capability were evaluated on an Apache Spark (MLlib) with audio recognition and filtering of streaming data.

We conducted experiments with multiple contexts of household traffic, offices, emergencies, and nature with real data collected from smartphones. Our experimental results show that the proposed framework for lightweight Machine Learning with the context aware model are very effective and efficient in terms of real time processing with a high accuracy rate of 90%, which is 20% higher than traditional approaches.

APPROVAL PAGE

The faculty listed below, appointed by the Dean of the School of Computing and Engineering, has examined a thesis titled "iHear – Lightweight Machine Learning Engine with Context Aware Audio Recognition Model" presented by Guru Teja Mannava, candidate for the Master of Science degree, and hereby certify that in their opinion, it is worthy of acceptance.

Supervisory Committee

Yugyung Lee, Ph.D., Committee Chair
Department of Computer Science & Electrical Engineering

Zhu Li, Ph.D.
Department of Computer Science & Electrical Engineering

Sejun Song, Ph.D.
Department of Computer Science & Electrical Engineering

Yongjie Zheng, Ph.D.
Department of Computer Science & Electrical Engineering

TABLE OF CONTENTS

ILLUSTRATIONS

ACKNOWLEDGEMENTS

I would like to express deepest gratitude to my advisor Dr. Yugyung Lee for her full support, expert guidance, understanding and encouragement throughout my study and research. Without her incredible patience and timely wisdom and counsel, my thesis work would have been an overwhelming pursuit. In addition, I express my application to Dr. Zhu Li, Dr. Sejun Song and Dr. Yongjie Zheng for having served on my committee. Their thoughtful questions and comments were valued greatly.

I would like to thank Seetha Rama Pradyumna Doddala for helping me to build a light weight engine with his SigSpace approach.

I would like to thank my data collection team Navya Redyy Sane, Vamsi, Pranusha, Padmini for helping me in collecting data for building models. Without their help my thesis would have took more time.

Thanks also go to my fellow graduate students at the University of Missouri Kansas City. Special thanks go to my numerous friends who helped me throughout this academic exploration.

Finally, I would like to thank my mother and sister for their unconditional love and support during the last two years; I would not have been able to complete this thesis without their continuous love and encouragement.

CHAPTER 1

INTRODUCTION

1.1 Motivation

Smart phones are the trend of wireless communication. It has become more prominent in daily life. As many smart phones strike an ideal balance of features and performance, it cannot replace Computers in terms of performance and compute power. Machine Learning requires building models on the data, which takes lots of computation power. Machine Learning on smart phones is typically not feasible due to the heavy loaded computation required for processing large-scale data. And the conventional Machine Learning systems do not naturally or efficiently support some very important features of large-scale streaming data. So, this formed the base for iHear engine which aimed to achieve lightweight Machine Learning through a collaboration between cloud and smartphones.

Our other motivation came from a hearing dog. Hearing specially trained dog for specially-abled people. It was trained to identify general household sounds like an alarm, doorbell, siren, washing machine, oven, bike, car, honks, blender, boiling water, cough, dishwasher, vacuums, shower, toothbrush, dryer machine, flush, clock etc.

It was trained to help deaf people by alerting them on hearing the owner's name or any of the above-mentioned sounds. Our project does the same job as hearing dog, which recognizes sounds, but has extra functionality to hearing dog, which is; it can differentiate the sounds and notify the user. The main motto of this project is to help those with hearing problems by assisting them in differentiating between the sounds. This is achieved by applying dynamic filters to the audio on a fly. In addition, we are motivated to provide an advanced feature found in Beats Headphones 'Noise Cancellation'. Our application does Noise Reduction is using filters.

Another motivation for this project is considered a problem we face in day-to-day life. It often occurs as a difficulty for people with hearing and communicating with the opposite person when in public places like restaurants, concerts, shows and railway stations due to the unavoidable surrounding noises. Using this app shall favor them in reducing the unwanted/unnecessary background noises and helps them concentrate only on the sounds they want to hear. They can easily reduce the background noise by applying filters and clearly hear what the opposite person is saying or has to say. This is another authentic way of using this app.

Another best place this app can find as a use is when people would like to study or work on the go. Most of the people make time in the travel to manage their office/personal work. It can be some work they are supposed to finish on their laptop or a journal to read or a conference call to attend. Turning on this app will automatically reduce the background noises and favors them in providing a better surrounding to work just like the beats headphones helps in cancelling the background noise and enhancing the music we want to hear.

1.2 Problem Statement

As many smart phones strike an ideal balance of features and performance, it cannot replace Computers in terms of performance and compute power. Machine Learning requires building models on the data, which takes lots of computation power. Machine Learning on smart phones is typically not feasible due to the heavy loaded computation required for processing large-scale data with Machine Learning for building models. Even making Machine Learning on the server side for building models recognize audio and sending to Client takes will have some network issues. And the conventional Machine Learning systems do not naturally or efficiently support some very important features of large-scale streaming data.

Another problem is building models with static data (data samples collected from open source), these models have a very less accuracy on smartphones, as we are dealing with real time data with some disturbances. Also, smartphone applications should be intelligent enough to know user activities and should automate it flow.

In today's life, everything is digitalized and made available within the reach of hands. The other alternatives to this app that are available to the common man are costly. Also, other alternative resources available in the market come with limited access, which has only some functionality. Taking this into consideration, we started to develop an app, which is handy, accessible and cost-efficient to the common man. Our application is a one-stop solution, which is having all necessary functionalities needed for better hearing aid.

1.3 Proposed Solution

Our machine provides an intelligent hearing aid, which will achieve Light weight Machine Learning engine with the collaboration between server and Cloud. Machine Learning for building models are done on the server side, these models are later saved on Cloud. Which are downloaded on client for audio recognition.

To achieve good accuracy on smartphones for audio recognition. We came up with a solution, rather than building models using static data, we are collecting real time data from the client, this data is saved on a cloud which are later used on server for building models. This model achieves better accuracy.

On the server side we are reducing the computation power by clustering the input data by using Self Organizing Map (SOM) and building models on it. We have even simplified the technique used for building model on the server side by considering the Context based approach that is based on the location's we are building separate models, which decreases data for building model on the server side. These Context based models are used on the Client side based on the location of client. There is so much research going on in the area of hearing aid and many devices has come up solving some of the major problems with amplification, noise cancellation etc.

CHAPTER 2

BACKGROUND AND RELATED WORK

2.1 Terminology

In this chapter, the key terms related to Machine Learning, Classification algorithms are introduced and their interpretations in the context of this paper are defined.

Machine Learning represents a type of artificial intelligence (AI) that provides computers with the ability to learn without being explicitly programmed. Machine Learning focuses on the development of computer programs that can teach them to grow and change when exposed to new data. The process of Machine Learning is similar to that of data mining. Both systems search through data to look for patterns. However, instead of extracting data for human comprehension - as is the case in data mining applications -- Machine Learning uses that data to detect patterns in data and adjust program actions accordingly. Machine Learning algorithms are often categorized as being supervised or unsupervised. Supervised algorithms can apply what has been learned in the past to new data. Unsupervised algorithms can draw inferences from datasets.

Classification Algorithm is a procedure for selecting a hypothesis from a set of alternatives that best fits a set of observations. In Machine Learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. In the terminology of Machine Learning, classification is considered an instance of supervised learning, i.e. learning where a training set of correctly identified observations is available. The corresponding unsupervised procedure is known as clustering, and involves grouping data into categories based on some measure of inherent similarity or distance.

The Audio signal is a representation of sound, typically as an electrical voltage. Audio signals have frequencies in the audio frequency range of roughly 20 to 20,000 Hz (the limits of human hearing). Audio signals may be synthesized directly, or may originate at a transducer such as a microphone, musical instrument pickup, phonograph cartridge, or tape head. Loudspeakers or headphones convert an electrical audio signal into sound. Digital representations of audio signals exist in a variety of formats.

2.2 Related Work

Automatic Sound Recognition for the Hearing Impaired [1]:

They present a wearable sound recognition system to assist the hearing impaired. Traditionally, hearing aid dogs are specially trained to facilitate the daily life of the hearing impaired. However, since training hearing aid dogs is costly and time-consuming, it would be desirable to substitute them with an automatic sound recognition system using speech recognition technologies. As the sound recognition system will be used in home environments where background noises and reverberations are high, conventional speech recognition techniques are not directly applicable, since their performance drops off rapidly in these environments. In this paper, we introduce a new sound recognition algorithm which is optimized for mechanical sounds such as doorbells. The new algorithm uses a new distance measure called the normalized peak domination ratio (NPDR) that is based on the characteristic spectral peaks of these sounds. The proposed algorithm showed a sound recognition accuracy of 99.7%, and noise rejection accuracy of 99.7%

Environmental Audio Scene and Activity Recognition through Mobile-based Crowdsourcing [2]:

Environmental audio recognition through mobile devices is difficult because of background noise, unseen audio events, and changes in audio channel characteristics due to the phone's context, e.g., whether the phone is in the user's pocket or in his hand. We propose a crowdsourcing framework that models the combination of scene, event, and phone context to overcome these issues. The framework gathers audio data from many people and share user-generated models through a cloud server to accurately classify unseen audio data. A Gaussian histogram is used to represent an audio dip with a small number of parameters, and a k-nearest classifier allows the easy incorporation of new training data into the system. Using the Kullback-Leibler divergence between two Gaussian histograms as the distance measure, we find that audio scenes, events, and phone context are classified with 85.2%, 77.6%, and 88.9% accuracy, respectively.

An Open and Portable Software Development Kit for Handheld Devices with Proprietary Operating Systems [3]:

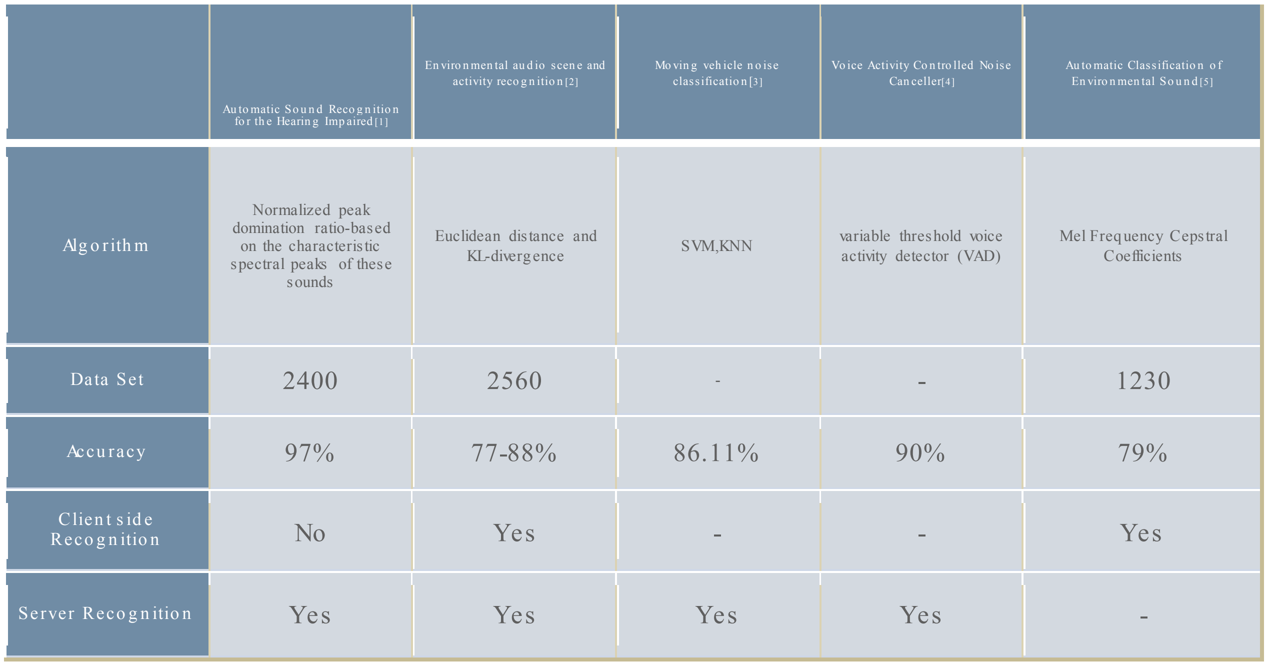
The availability of modern handheld devices equipped with powerful CPUs, large main and secondary memory, webcam and wireless connection suggests that those devices might also be used to develop sophisticated applications based on computationally intensive algorithms, including for example face recognition, object recognition, character recognition or speech recognition applications. A wide choice of software implementations exists for each of the categories listed above, but almost all of them are designed and developed for the PC platform, and porting complex software from PC to the most common handheld platforms can be very time consuming and often inefficient. In this paper, we present Nano desktop, a portable software development kit created to make easier and more efficient the development and the porting of complex software to handheld platforms, in particular those with proprietary operating systems such as the Sony Playstation Portable. Practical examples and real use cases are presented and evaluated. Nano desktop is an open source software and can be freely downloaded for testing and evaluation.

Smartphone Application for Automatic Classification of Environmental Sound [4]:

Sounds are an important source for events in the environment. They convey information about events, even when these are not in line of sight. This information can warn a person that a danger is occurring. E.g. a pedestrian can estimate the distance of a vehicle and if the vehicle is approaching or departing. In this contribution a mobile sound classification system based on a smartphone running the Android operating system is presented. The software was developed in Java and applies pattern recognition methods to recognize environmental sounds. It extracts thirteen Mel Frequency Cepstral Coefficients (MFCC) from the data collected by the microphone and classifies the sound using the neural network. The use of pattern recognition methods makes the approach easily adaptable to different sounds. As application example the presented system is trained to recognize sounds of emergency vehicles in the road traffic.

Computational Diagnosis of Parkinson's Disease Directly from Natural Speech Using Machine Learning Techniques [5]:

The human voice signal carries much information in addition to direct linguistic semantic information. This information can be perceived by computational systems. In this work, we show that early diagnosis of Parkinson's disease is possible solely from the voice signal. This is in contrast to earlier work in which we showed that this can be done using hand-calculated features of the speech (such as formants) as annotated by professional speech therapists. In this paper, we review that work and show that a differential diagnosis can be produced directly from the analog speech signal itself. In addition, differentiation can be made between seven different degrees of progression of the disease (including healthy). Such a system can act as an additional stage (or another building block) in a bigger system of natural speech processing. For Example, it could be used in automatic speech recognition systems that are used as personal assistants (such as IPhones Siri, Google Voice), or as natural man-machine interfaces. We also conjecture that such systems can be extended to monitor and classifying additional neurological diseases and speech pathologies. The methods presented here using a combination of signal processing features and Machine Learning techniques.



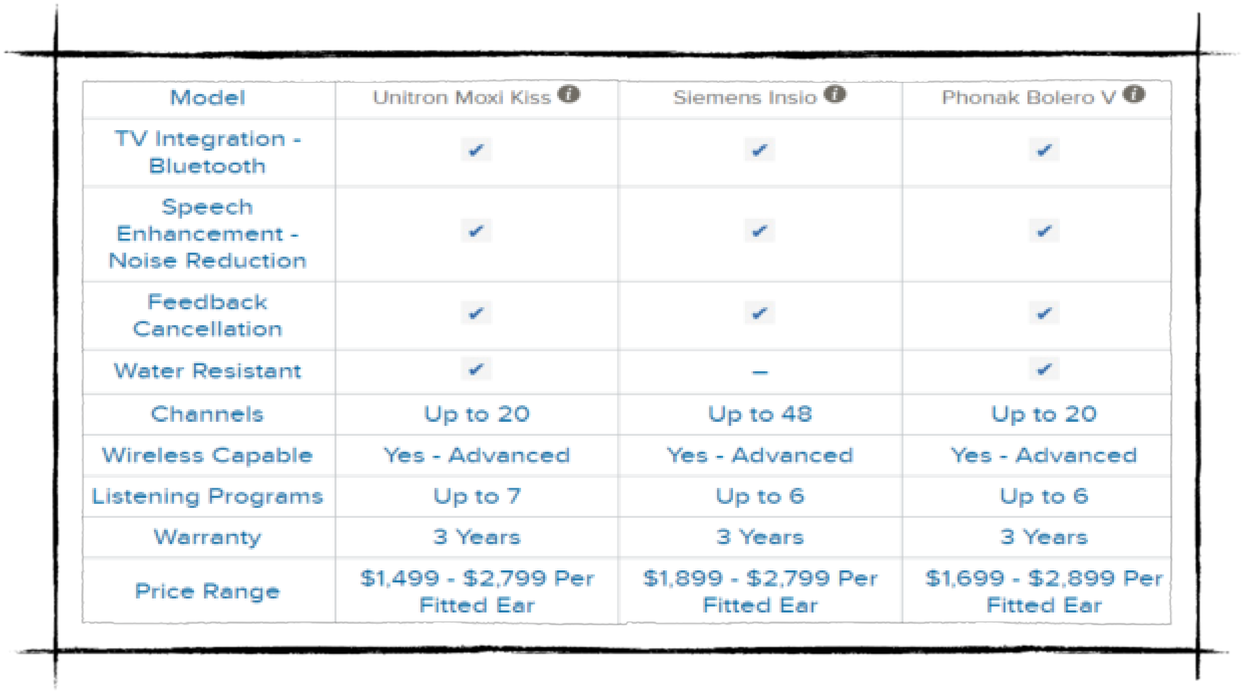
	Automatic Sound Recognition for the Hearing Impaired[1]	Environmental audio scene and activity recognition[2]	Moving vehicle noise classification[3]	Voice Activity Controlled Noise Canceller[4]	Automatic Classification of Environmental Sound[5]
Algorithm	Normalized peak domination ratio-based on the characteristic spectral peaks of these sounds	Euclidean distance and KL-divergence	SVM,KNN	variable threshold voice activity detector (VAD)	Mel Frequency Cepstral Coefficients
Data Set	2400	2560	-	-	1230
Accuracy	97%	77-88%	86.11%	90%	79%
Client side Recognition	No	Yes	-	-	Yes
Server Recognition	Yes	Yes	Yes	Yes	-

Figure 1: Related Work

Figure 1, is the comparison chart for all the related work specified. Each related work used a limited amount of data set. Some have client side recognition or has server side recognition or both. And has accuracy from 97% to 77%.

Libxtract [6]: LibXtract is a simple, portable, lightweight library of audio feature extraction functions. The purpose of the library is to provide a relatively exhaustive set of feature extraction primitives that are designed to be 'cascaded' to create an extraction hierarchy. For example, 'variance', 'average deviation', 'skewness' and 'kurtosis', all require the 'mean' of the input vector to be precomputed. However, rather than compute the 'mean' 'inside' each function, it is expected that the 'mean' will be passed in as an argument. This means that if the user wishes to use all of these features, the mean is calculated only once, and then passed to any functions that require it.

This philosophy of 'cascading' features is followed throughout the library, for example, with features that operate in the magnitude spectrum of a signal vector (e.g. 'irregularity'), the magnitude spectrum is not calculated 'inside' the respective function, instead, a pointer to the first element in an array containing the magnitude spectrum is passed in as an argument.



Model	Unitron Moxi Kiss	Siemens Insio	Phonak Bolero V
TV Integration - Bluetooth	✓	✓	✓
Speech Enhancement - Noise Reduction	✓	✓	✓
Feedback Cancellation	✓	✓	✓
Water Resistant	✓	–	✓
Channels	Up to 20	Up to 48	Up to 20
Wireless Capable	Yes - Advanced	Yes - Advanced	Yes - Advanced
Listening Programs	Up to 7	Up to 6	Up to 6
Warranty	3 Years	3 Years	3 Years
Price Range	$1,499 - $2,799 Per Fitted Ear	$1,899 - $2,799 Per Fitted Ear	$1,699 - $2,899 Per Fitted Ear

Figure 2: Existing Applications: Hearing Aid

Figure 2 shows the existing applications for hearing aids and their available features like TV Integration - Bluetooth, Speech Enhancement-Noise reduction, Feedback cancellation, Water resistant, Number of channels, Wireless capable, Listening programs, warranty and price Range. Each has a price range from $ 1,499 to $ 2,899.

Existing Approaches: Music Recognition [7]

Shazam [7] - With unlimited tagging and loads of features, the original music ID app is still the best. Shazam identifies songs quickly and accurately, and the app is fully integrated with Facebook, Twitter, Spotify, and Pandora. It has also scrapped most of the limitations they used to have on the software, meaning you can now tag as many songs as you want.

SoundHound[7]: Have a song stuck in your head, but can’t think of the name? Soundhound can help you with those songs that are on the tip of your tongue, or ear rather. This app can identify a song for you, just by humming the melody or singing a few lyrics, in addition to providing Sham-style tagging. You have to be pretty accurate with your humming for it to guess the song, because it can’t read minds (yet). Tagging a song as its playing lets you see the lyrics as they’re sung, just like karaoke. You can also check out what’s hot to see what other people are tagging.

MusiXmatch [7]: MusiXmatch has powerful lyric recognition and a large catalog of lyrics to accurately find what you’re looking for, based on song lyrics. You can tag and save lyrics, share them, and even browse lyrics while offline. The app is compatible with many third-party music players to give you lyrics as you’re listening.

CHAPTER 3

PROPOSED FRAMEWORK

3.1 Overview

Machine Learning requires building models on the data, which takes lots of computation power. Machine Learning on smart phones is typically not feasible due to the heavy loaded computation required for processing large-scale data. And the conventional Machine Learning systems do not naturally or efficiently support some very important features of large-scale streaming data. After analyzing the problems and objectives, we came up with a set of features and framework in our application. Our framework goal is to achieve an engine which aims for a lightweight Machine Learning engine with collaboration between server and cloud environment. We are testing and building models on the server and these models are saved in cloud environment which are later downloaded on smart phones for recognition.

We have reduced space complexity by reducing the number of data points used for building models. We have achieved this by clustering the data through a self-organized map. By which we have achieved less build time and space complexity of the server.

Another objective is to increase the model accuracy, using real time data. Rather than using static data we are using real time data for testing and training models. Despite of finding audio samples from open source we tried collecting audio samples from the client(smartphones) and made use of it as the models built with static data doesn't give accurate result when compared with models build with real time data.

Rather than building models for complete data we have built models in different contexts like household, emergency, office, traffic, animals and nature which shows great improved accuracy on smartphones when compared to models built by using the complete data(no context). Our application keeps track of user information like Geo location and time and this is used to fetch that particular context and is downloaded from cloud and used.

The User can apply dynamic filters on dynamic data coming from receiver i.e., mike of a smartphone. The three types of filters used are hi-pass, low-pass, and band-pass. Our application achieves noise reduction on dynamic data by applying low-pass filter at frequency 1012 Hz. Another small feature is voice changer where you can record a sound and change pitch, tempo of the recorded sound to produce a new sound.

3.2 Collaboration: Cloud and Smartphone

Our proposed framework is achieved by collaboration between cloud and smartphones. Figure 3 shows the overall flow between Smartphones and Cloud and Server. Real time data from the client is used as testing and training the classification models on the server. The Server then saves the model to Cloud, which are later downloaded on smartphones for audio recognition. In the first step, as real time audio data are collected through the receiver of smartphones, the audio features are extracted from each audio data. These features are then saved on a cloud with class name like ‘dog’, ‘door bell’ etc. On server these class names are used to distinguish data. In step two, Audio features are then saved to cloud i.e. MongoDB. There features are collected on the server in step three. When the data collection is done for a particular context (for example, if we are building a model for House hold context, we collect door bell, dog, Vacuum and Baby cry feature data from MongoDB). In step 5, as the data collection part is completed, this data is sent to Apache spark for building model for all contexts. These models are saved on MongoDB in step 6. Late in step 7 and 8, based on user location and time, Smartphones download respective model from MongoDB for audio recognition process.

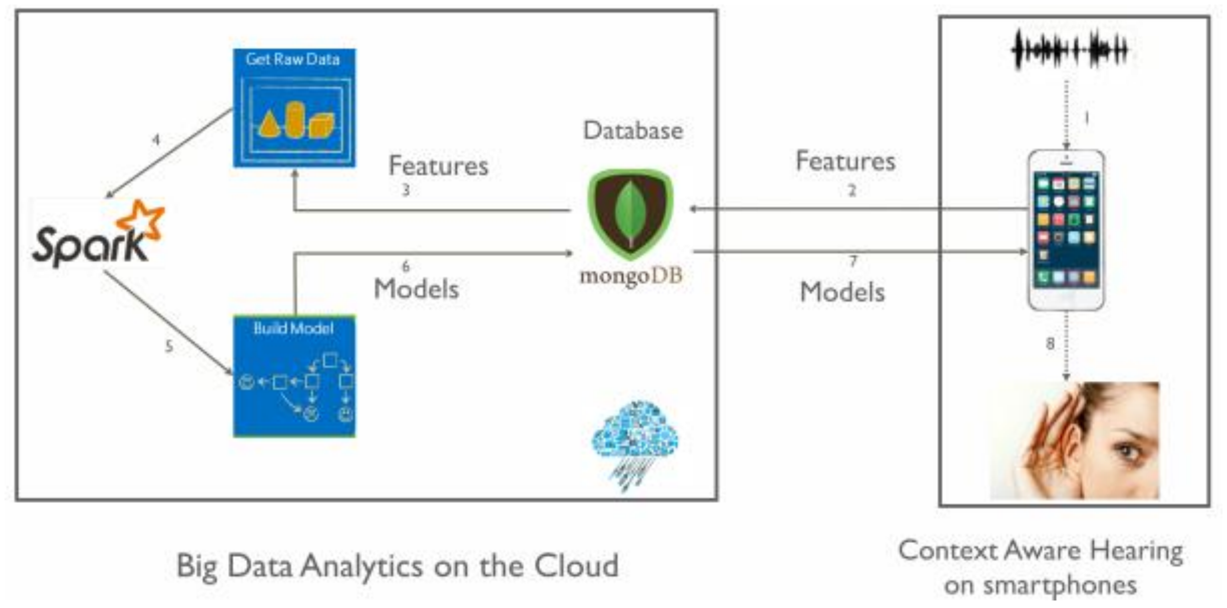


Figure 3: Collaboration: Cloud and Smartphones

3.3 Architecture

Our proposed architecture in figure 4 includes three major parts- a) Client b) Cloud and c) Server.

a) Client: Client side process has two modules, First module is for collecting data on client for model building. In the second module audio data is collected for audio recognition. In First Module, as soon as the audio data is received through the receiver of smartphone in step 1, data send to LIbXtract library for feature extraction in step 2. We are extracting 11 features from the library, namely Mean, Standard Deviation, Variance, Spectral Centroid, Skewness, Kurtosis, ZCR, irregularity_J, irregularity_K, Rolloff, ZCR, we also normalizing the feature set to fall in a range from [11-13] each. For testing models feature data is appended to the respective context class i.e. door bell, dog bark, etc. As soon as the feature data is normalized it is saved to MongoDB in step 4.

In Second Module step 1, 2 and 3 are common. At step 9 based on user location and time,the application decides user context as an office or House or traffic and downloads respective context model from MongoDB, which are used for audio recognition at step 10. We have five different set of audio context. Each context has four classes. For example, in Office context, it has four classes, namely Door knock, phone call, Fax machine, Printer.

b) Cloud: We are using MongoDB as our cloud environment, it acts as a mediator between Client and Server. Further data are saved to the cloud by the client, this feature data is collected on the server. Later models are saved on cloud, which are downloaded on client for audio recognition.

C) Server: Machine Learning is completely done on server side. While Building models for each context - Household, traffic, nature, animal and emergency, we collect data from MongoDB as training data for Spark at step 5. For example, if we are building a model for Traffic context. We collect data for Door knock, Fax machine, Printer and phone ring from MongoDB (We can identity context data on MongoDB with the prefix name appended to feature data). This complete training data for traffic context is passed through SOM (self-organized map) in step 6 for getting clustered featured data points. Which are passed to Spark MLlib for building a traffic model. This traffic model is then saved on MongoDB for future use. This whole process is repeated for building models for all contexts and saving it in MongoDB.

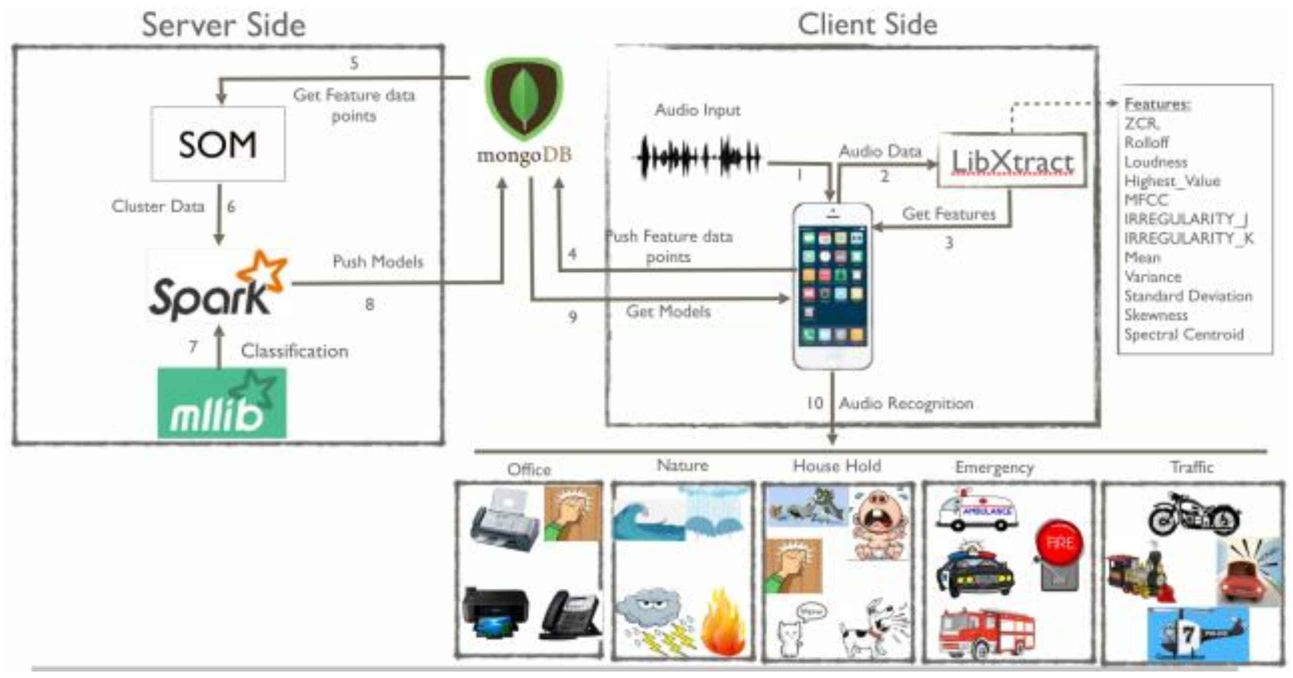


Figure4: Architecture

3.4 Phase I: Feature Extraction and Selection (Smartphones)

LibXtract library is used for feature extraction and Feature Selection is done recursively .

3.4.1 Workflow:

In the feature extraction and selection workflow, as real time audio data are collected on iHear application, it is sent to LibXtract for feature extraction. These features are then passed to two approaches ‘Removing features with Low variance’ and ‘Recursive Feature Elimination’ for getting set of features which are more accurate than using all features together. The two approaches leads to this set of features - Mean, Standard Deviation, Variance, Spectral Centroid, Skewness, Kurtosis, ZCR, irregularity_J, irregularity_K, Rolloff, ZCR.

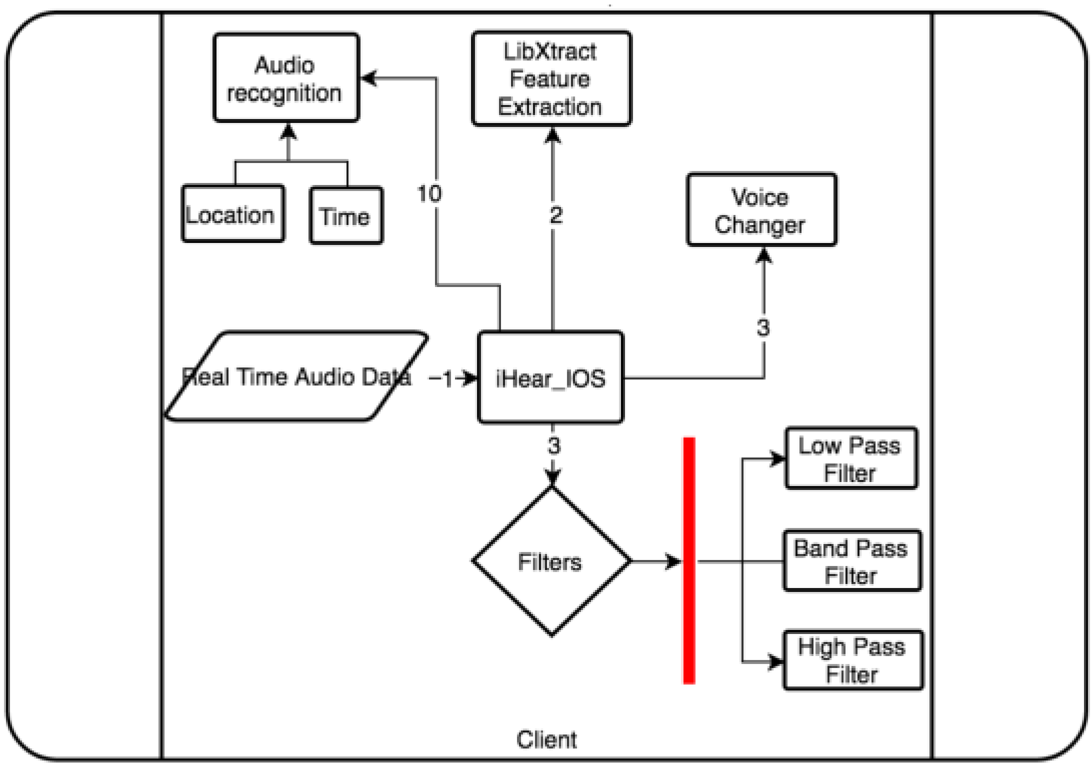


Figure 5: Workflow - Feature Extraction

3.4.2 Feature Extraction: Time Domain Features

Different types of features can be extracted from an audio signal. Here we will discuss more about Time domain features, which are dependent on time and amplitude.

Pitch: Pitch is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. Pitch is a term used to describe how high or low a note a being played by a musical instrument or song seems to be. The pitch of a note depends on the frequency of the source of the sound. Frequency is measured in Hertz (Hz), with one vibration per second being equal to one hertz (1 Hz) [10].

A high frequency produces a high pitched note and a low frequency produces a low pitched note. Figure 5 shows how lower pitch and higher pitch looks.

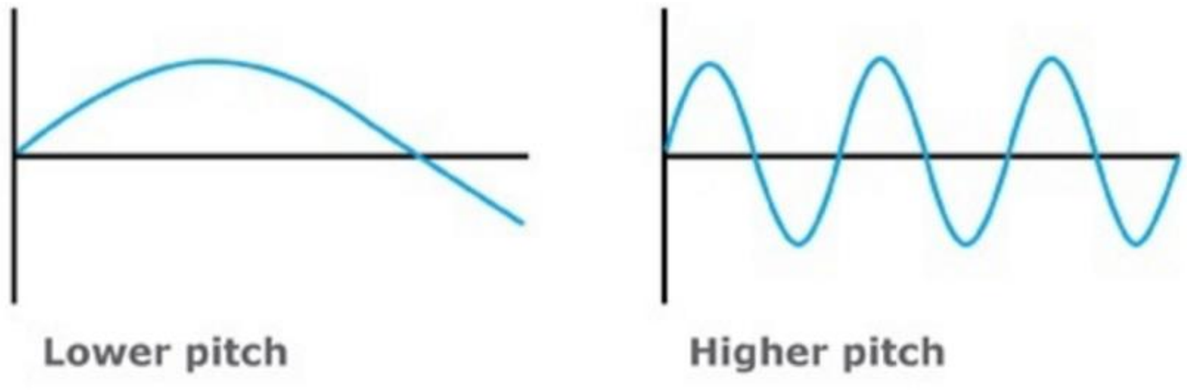


Figure 6: Pitch

Loudness: The amplitude of a sound wave determines its loudness or volume. A larger amplitude means a louder sound, and a smaller amplitude means a softer sound. In Figure 6 right sound is louder than left sound. The vibration of a source sets the amplitude of a wave. It transmits energy into the medium through its vibration. More energetic vibration corresponds to larger amplitude. The molecules move back and forth more vigorously.

The loudness of a sound is also determined by the sensitivity of the ear. The human ear is more sensitive to some frequencies than to others. The volume we receive thus depends on both the amplitude of a sound wave and whether its frequency lies in a region where the ear is more or less sensitive [10].

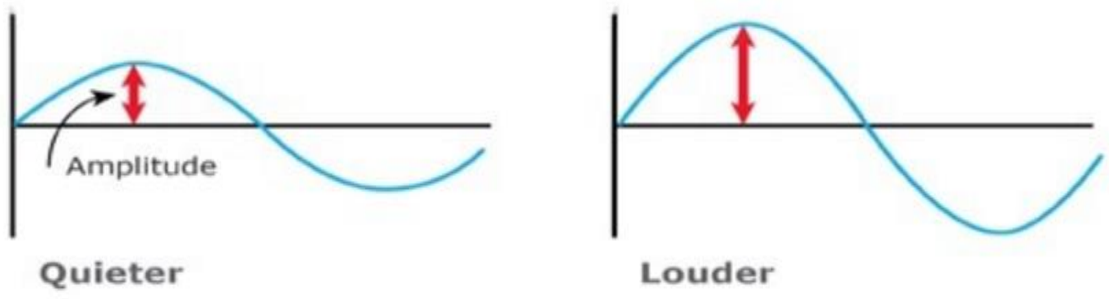


Figure7: Loudness

Bandwidth: Bandwidth is the difference between the upper and lower frequencies in a continuous set of frequencies. It is typically measured in hertz, and may sometimes refer to passband bandwidth, sometimes to baseband bandwidth, depending on context. Passband bandwidth is the difference between the upper and lower cutoff frequencies of, for example, a band-pass filter, a communication channel, or a signal spectrum. In the case of a low-pass filter or baseband signal, the bandwidth is equal to its upper cutoff frequency[10].

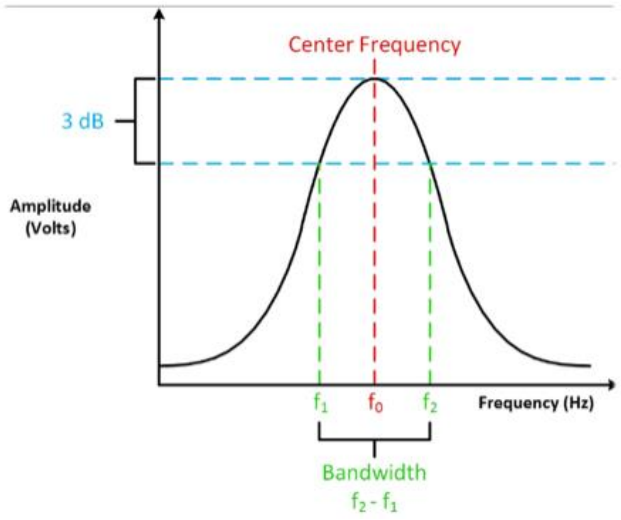


Figure8: Bandwidth

Zero Crossing: Zero Crossing is the rate of sign-changes along a signal or the number of times that the sound signal crosses the x-axis. Zero-crossing is a point where the sign of a mathematical function change (e.g. from positive to negative), represented by a crossing of the axis (zero value) in the graph of the function. It is a commonly used term in electronics, mathematics, sound, and image processing. In alternating current, the zero-crossing is the instantaneous point at which there is no voltage present. In a sine wave or other simple waveform, this normally occurs twice during each cycle [10].

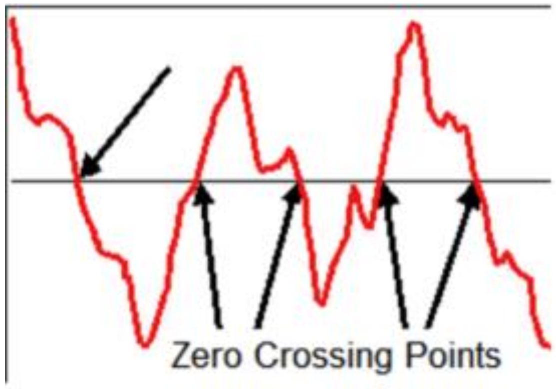


Figure 9: Zero Crossing Rate

Short Time Energy: A measure for the energy of the frames of a sound signal. This measure gives insight into the intensity and the variations in intensity of a sound signal. The amplitude of the speech signal varies appreciably with time. In particular, the amplitude of unvoiced segment is generally much lower than the amplitude of voice segments. S.T energy provides a convenient representation that reflects these amplitude variations [10].

$$E_n = \sum_m [\, s(m) w(n-m) \,]^2$$

Where s(m) is a short time speech segment obtained by passing the speech signal x(n) through window w(n):

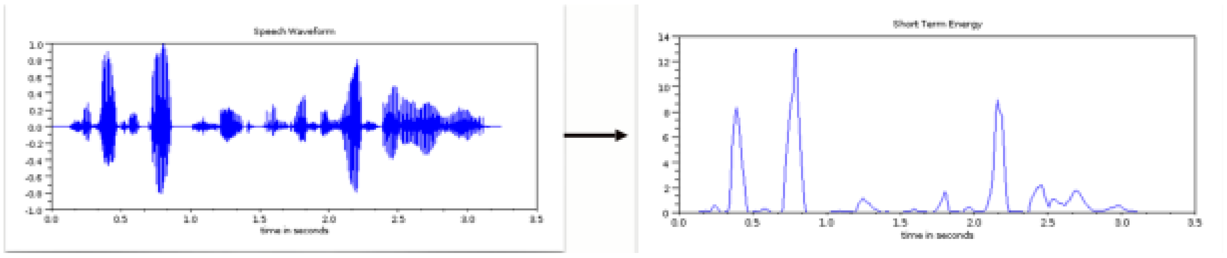


Figure 10: Short Time Energy

3.4.3 Feature Extraction: Frequency Domain Features

Skewness: In probability theory and statistics, skewness is a measure of the asymmetry of the probability distribution of a real-valued random variable about its mean. The skewness value can be positive or negative, or even undefined. The qualitative interpretation of the skew is complicated. For a unimodal distribution, negative skew indicates that the tail on the left side of the probability density function is longer or faster than the right side – it does not distinguish these shapes. Conversely, positive skew indicates that the tail on the right side is longer or fatter than the left side. In cases where one tail is long but the other tail is fat, skewness does not obey a simple rule. From figure 10, we can say that wave form which are longer on the left are said to be Positive Skewness and wave forms which are longer on the right side are said to be Negative Skewness[10].

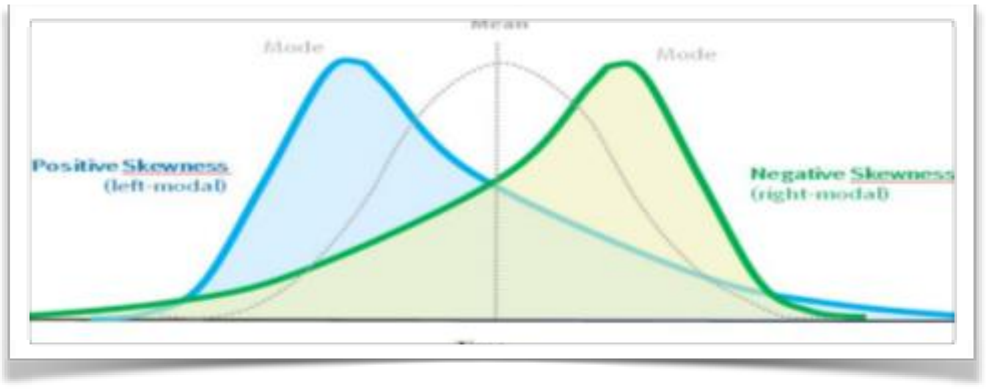


Figure11: Skewness

Kurtosis: is the measure of the peak of a distribution, and indicates how high the distribution is around the mean. In probability theory and statistics, kurtosis is a measure of the "tailedness" of the probability distribution of a real-valued random variable. In a similar way to the concept of skewness, kurtosis is a descriptor of the shape of a probability distribution and, just as for skewness, there are different ways of quantifying it for a theoretical distribution and the corresponding ways of estimating it from a sample from a population. Depending on the particular measure of kurtosis that is used, there are various interpretations of kurtosis, and of how particular measures should be interpreted [10].

From figure 11, we can say that wave forms with less amplitude are said to be Negative Kurtosis and wave from with high amplitude are said to be Positive Kurtosis.

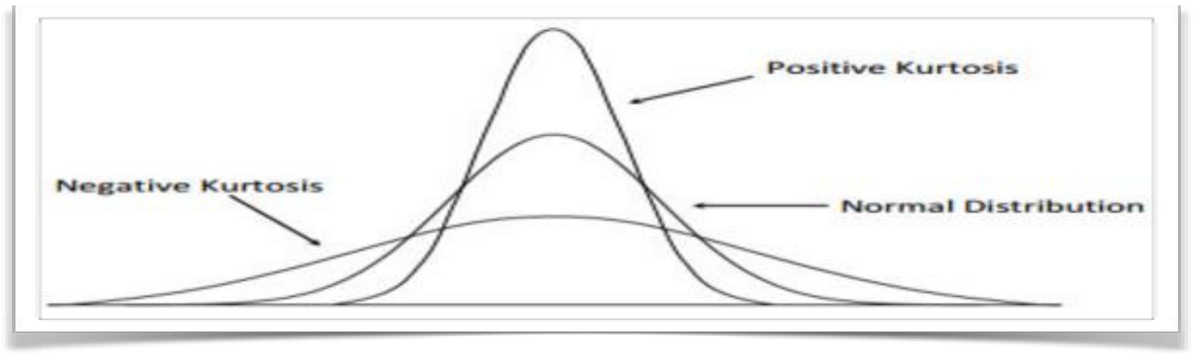


Figure 12: Kurtosis

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. Perceptually, it has a robust connection with the impression of "brightness" of a sound.

It is calculated as the weighted mean of the frequencies present in the signal, determined using a Fourier transform, with their magnitudes as the weights:

$$Centroid = \frac{\sum_{n=0}^{N-1} f(n)\, x(n)}{\sum_{n=0}^{N-1} x(n)}$$

where x(n) represents the weighted frequency value, or magnitude, of bin number n, and f(n) represents the center frequency of that bin.

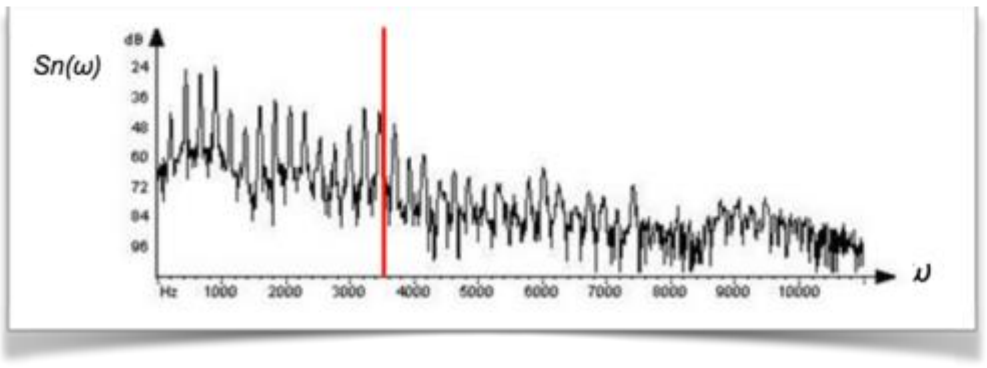


Figure13: Spectral Centroid

Spectral Irregularity: Spectral irregularity represents jaggedness of spectral envelope. Following figure 13, illustrate how Spectral irregularity looks like when it is represented in spectral envelope.

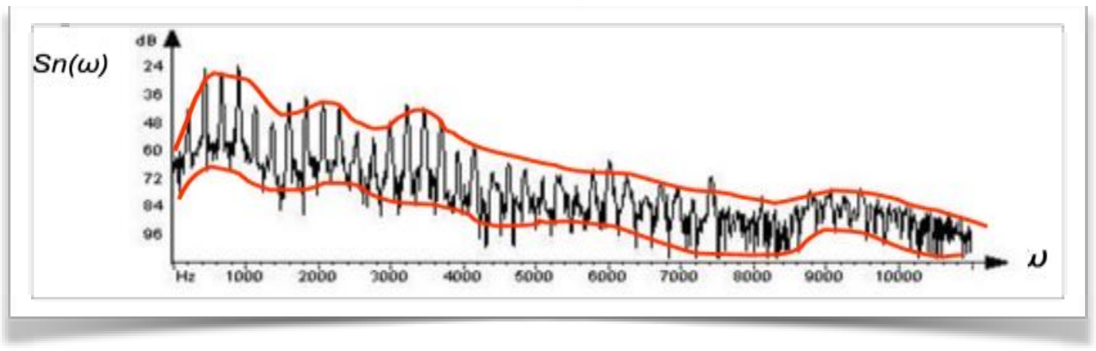


Figure14: Spectral Irregularity

Spectral Rolloff Point: spectral roll-off point is the frequency below which 95th percentile of the power in the spectrum resides [30]. This is a measure of the "skewness" of the spectral shape - the value is higher for right-skewed distributions [30]. SR can tells voiced speech from unvoiced speech as unvoiced speech has a high proportion of energy contained in the high-frequency range of the spectrum, while most of the energy for unvoiced speech and music is contained in lower bands of spectrum. Figure 14 shows the energy spectrum (cumulative energy) along frequency with 95% spectral roll-off frequency [8]. The spectral roll-off value for a frame is computed as follows:

$$SR = K \text{ , where } \sum_{f=0}^{K} E[f] = 0.95 \sum_{f=o}^{fMAX} E[f]$$

E[f] is the energy of the signal at the frequency *f*. *fMAX* is the maximal frequency in the spectrum.

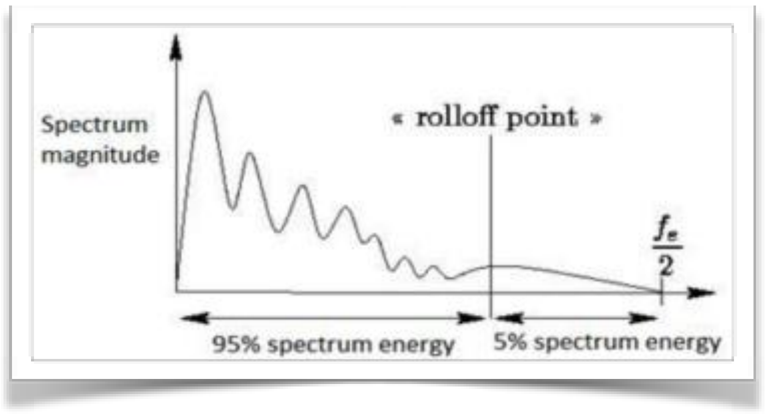


Figure15: Spectral Rolloff

Spectral Centroid: the "balancing point" of the spectral power distribution. Many types of music involve percussive sounds which push the spectral mean higher by including high-frequency noise [30]. The spectral centroid for a frame is computed as follows:

$$SC = \frac{\sum_{k} k \cdot X[k]}{\sum_{k} X[k]}$$

Where: *k* is an index corresponding to a frequency, or small band of frequencies within the overall measured spectrum, and *X*[*k*] is the power of the signal at the corresponding frequency band. Table 3.5 shows the twelve statistics of SC in our work.

Spectrogram: A spectrogram is a visual representation of the spectrum of frequencies in a sound or other signal as they vary with time or some other variable. Spectrograms are sometimes called spectral waterfalls, voiceprints, or voice grams.

Spectrograms can be used to identify spoken words phonetically, and to analyze the various calls of animals. They are used extensively in the development of the fields of music, sonar, radar, and speech processing, seismology [1].

The instrument that generates a spectrogram is called a spectrograph.

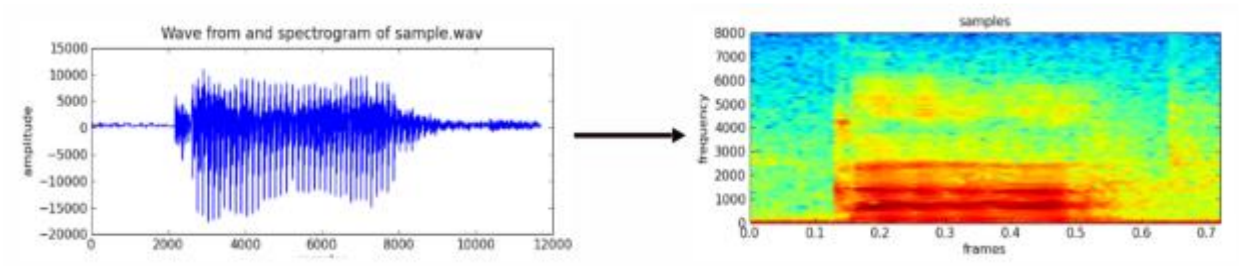


Figure 16: Spectrogram

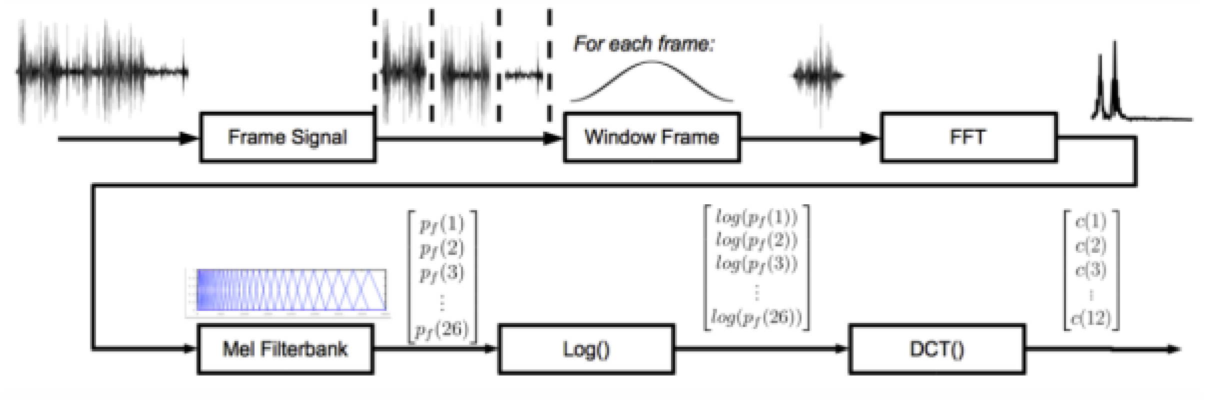
Mel Frequency Cepstral Coefficients (MFCCs): MFCCs is the feature popularly used for speech recognition and audio classification due to its effectiveness in representing the spectral variations of audio. The MFCCs stands in the shape of the spectrum with few coefficients. The cepstrum is the Fourier Transform (or Discrete Cosine Transform DCT) of the logarithm of the spectrum. Instead of the Fourier spectrum, the Mel-cepstrum is the cepstrum computed on the Mel-bands. By using of the mel scale, the mid-frequency part of the signal is taken better taken into account. The MFCCs are the coefficients of the Mel cepstrum.

MFCCs are commonly derived as follows: This is shown in figure 16.

1. Take the Fourier Transform of (a windowed excerpt of) a signal.
2. Map the log amplitudes of the spectrum obtained above onto the mel scale, using triangular overlapping windows.

3. Take the Discrete Cosine Transform of the list of mel log-amplitudes, as if it was a signal.

4. The MFCCs are the amplitudes of the resulting spectrum



For each frame:
Frame Signal
Window Frame
FFT
Mel Filterbank
$p_f(1)$ $p_f(2)$ $p_f(3)$ ⋮ $p_f(26)$
Log()
$log(p_f(1))$ $log(p_f(2))$ $log(p_f(3))$ ⋮ $log(p_f(26))$
DCT()
$c(1)$ $c(2)$ $c(3)$ ⋮ $c(12)$

Figure 17: MFCC

3.4.4 Feature Selection[13]:

In Machine Learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Feature selection techniques are used for three reasons:

- Simplification of models to make them easier to interpret by researchers/users.
- shorter training times,
- enhanced generalization by reducing overfitting

We are doing a feature selection by following two approaches:

Removing Features with Low Variance:

Variance Threshold is a simple baseline approach to feature selection. It removes all features whose variance don't meet some threshold. By default, it removes all zero-variance features, i.e. features that have the same value in all samples.

As an example, suppose that we have a dataset with Boolean features, and we want to remove all features that are either one or zero (on or off) in more than 80% of the samples. Boolean features are Bernoulli random variables, and the variance of such variables is given by

$$\mathrm{Var}[X] = p(1-p)$$

so we can select using the threshold .8 * (1 - .8).

Recursive Feature Elimination:

Given an external estimator that assigns weights to features (e.g., the coefficients of a linear model), recursive feature elimination (RFE) is to select features by recursively considering smaller and smaller sets of features. First, the estimator is trained on the initial set of features and weights are assigned to each one of them. Then, features whose absolute weights are the smallest are pruned from the current set features. That procedure is recursively repeated in the pruned set until the desired number of features to select is eventually reached.

3.4.5 Feature scaling & Data Normalization:

The result of standardization (or Z-score normalization) is that the features will be rescaled so that they'll have the properties of a standard normal distribution with μ=0 and σ=1, where μ is the mean (average) and σ is the standard deviation from the mean; standard scores (also called z scores) of the samples are calculated as follows:

$$z = \frac{x - \mu}{\sigma}$$

Feature scaling was used to standardize the range of independent variables or features of data. The simplest method is rescaling the range of features to scale the range in [0, 1] or [−1, 1]. Selecting the target range depends on the nature of the data. The general formula is given as:

$$x' = \frac{x - \min(x)}{\max(x) - \min(x)}$$

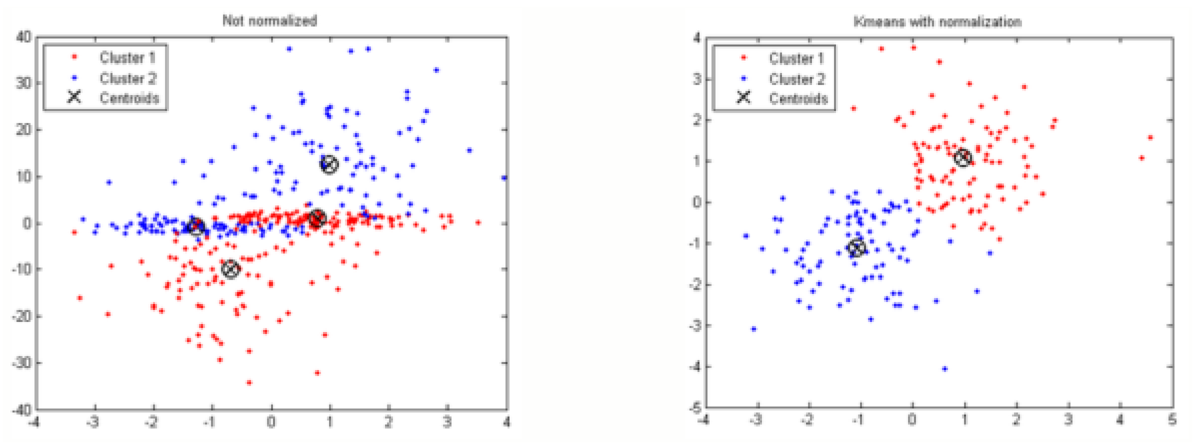
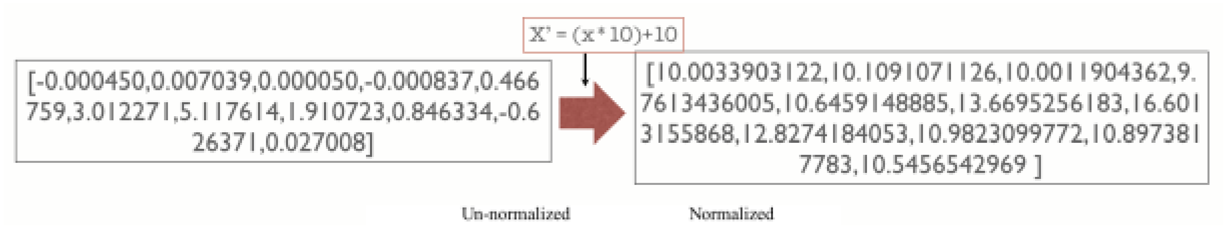


Figure18: Feature Scaling

Figure 18 shows the difference between k-means normalized data vs without normalized data. K-means cluster after normalizing the data look more organized data points on the clusters.



[-0.000450,0.007039,0.000050,-0.000837,0.466759,3.012271,5.117614,1.910723,0.846334,-0.626371,0.027008]

[10.0033903122,10.1091071126,10.0011904362,9.7613436005,10.6459148885,13.6695256183,16.6013155868,12.8274184053,10.9823099772,10.8973817783,10.5456542969]

Figure19: Normalized Real time data

Data normalization is computational data transformations intended to remove certain systematic biases from audio data. In figure 19, We have normalized real time feature data to normalized feature data with a range of [10-12] by applying some mathematical formula.

3.5 PHASE II: MODEL BUILDING for BIG DATA ANALYTICS ON CLOUD

Following are the steps involved in Phase II.

Phase II-I: Audio Class Signature Generation using Machine Learning Algorithm

Phase II-II: Audio Classification using Supervised Machine Learning Algorithms based on Audio Signature

3.5.1 Phase II: Model Building Workflow:

Machine Learning completely is done on the server side. While building models for a particular context, server gets all the data required for a particular context as a training set for building models. These data are sent to SOM (self-organized map) to get clustered data points. These clustered data points are sent to Spark MLlib for model building. Here we are building decision tree and random forest. These are saved to MongoDB for further use.

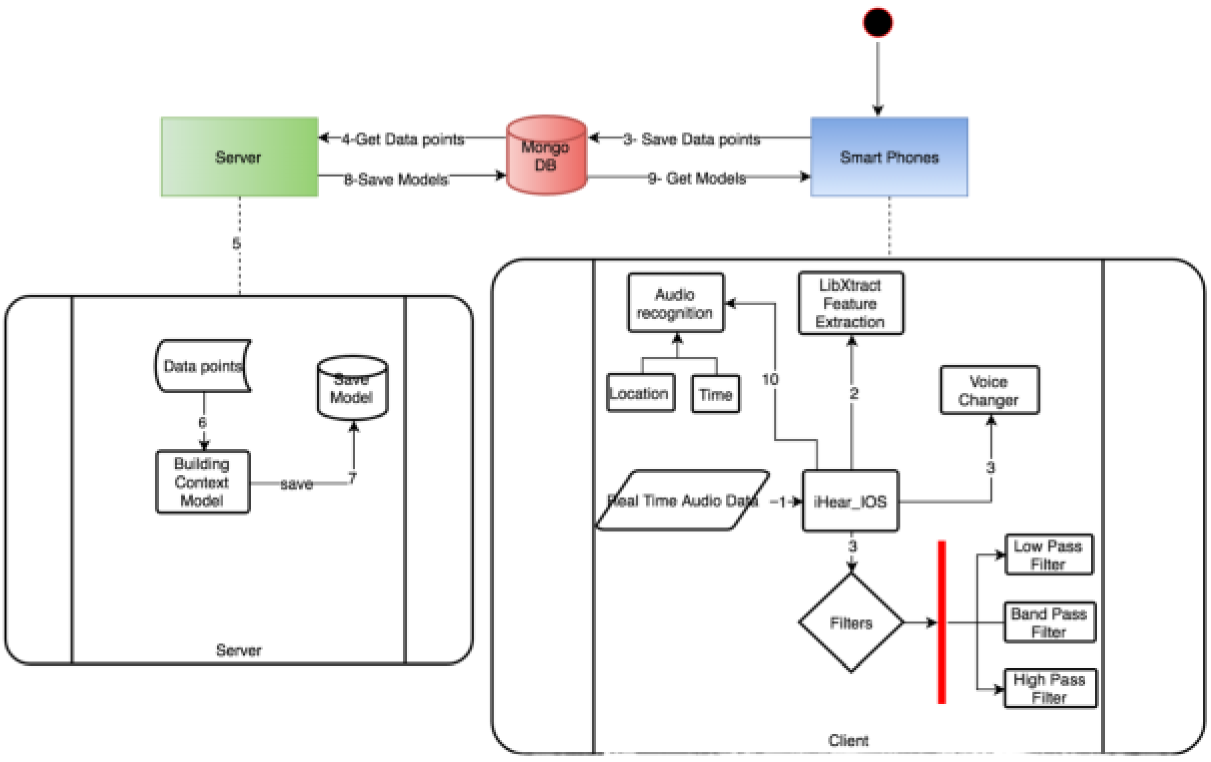


Figure 20: Workflow- Model Building

3.5.2 Phase II - I: Audio Signature Generation

Audio signature is generated for each audio class of a context using Self Organizing Maps (SOM). SOM is a type of artificial neural network (ANN) - Trained using unsupervised learning to produce a low-dimensional signature for given audio features. Discretized representation of the input space of the training samples, called a map.

From figure 21, consider House hold context, which has 5 classes, namely vacuum, Baby cry, dog bark, door knock, phone ring. Each class has its own feature data which is send to Self-organized map for clustered data. Feature data of the same type are clustered to one group. Resultant clusters data occupy less space.

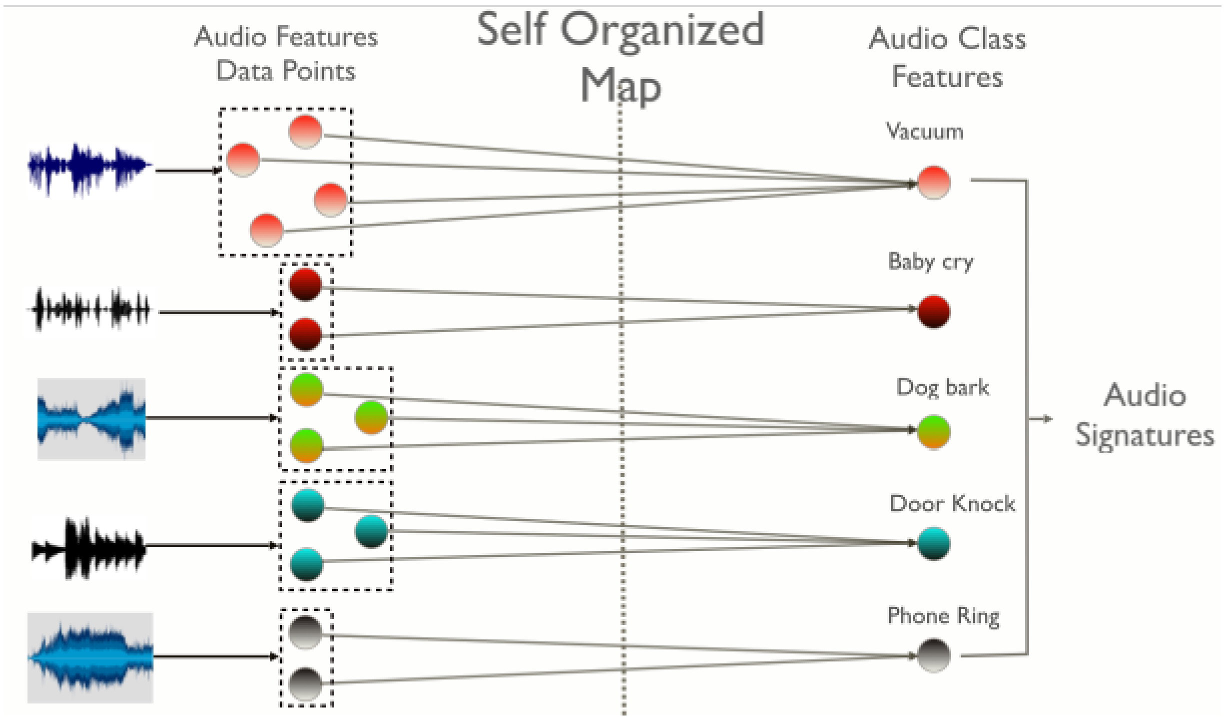


Figure21: SOM

3.5.3 Phase II - II: Audio Classification[9]

In Machine Learning and statistics, classification is the problem of identifying to which of a set of categories (sub-populations) a new observation belongs, on the basis of a training set of data containing observations (or instances) whose category membership is known. An example would be assigned a given email into "spam" or "non-spam" classes or assigning a diagnosis to a given patient as described by observed characteristics of the patient (gender, blood pressure, presence or absence of certain symptoms, etc.). Classification is an example of pattern recognition[13][9].

a. Naive Bayes:

The Naive Bayes technique is used to design the classifiers using the features and labels of the datasets. In general the Machine Learning algorithms must be trained in order to classify and predict. The Naive Bayes algorithm is one of the classification algorithm based on Bayes theorem to train the dataset. It generates the models to identify the relation between input data and the predictable data. These classifiers consider each attribute as independent and generates the features and using each feature it generate the probability of identifying.

b. Decision Tree:

A decision tree is a decision support tool that uses a treelike graph or model of decisions and their possible consequences, including chance event outcomes, resource costs, and utility. It is one way to display an algorithm. Decision trees are commonly used in operations research, specifically in decision analysis, to help identify a strategy most likely to reach a goal, but are also a popular tool in Machine Learning.

A decision tree is a flowchart-like structure in which each internal node represents a "test" on an attribute (e.g., whether a coin flip comes up heads or tails), each branch represents the outcome of the test and each leaf node represents a class label

Decision Tree is a supervise Machine Learning algorithm. It builds audio classifiers using audio signatures. Classification rules extracted from the decision tree classifiers are saved and reused for client side audio recognition.

Figure 22 shows how decision tree flow looks, and how it parses through the branches to reach the final results based on intermediate nodes that is SigFeat features. For example, for recognizing 'baby cry' sound it goes to 3 levels, in level one the root node checks if SigFeat0 is less than or equal to 9.997 if it is true, then it goes to next level node to check if SigFeat6 is less than or equal to 10.161. If this is true, then it goes to level three nodes and checks if SigFear1 is less or equal to 9.991. If this condition is true, it comes to conclusion that the sound is a 'Baby cry'.

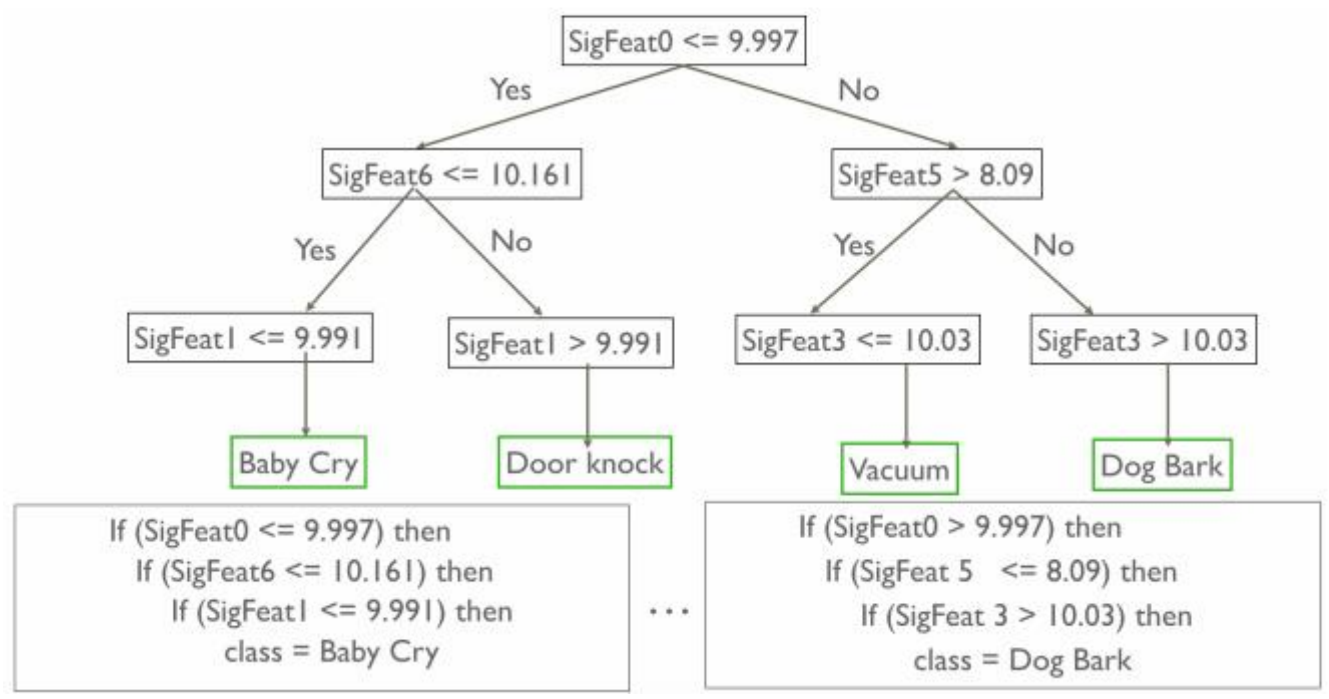


Figure 22: Decision Tree

3.6 PHASE III: CONTEXT AWARE AUDIO RECOGNITION (SMARTPHONES):

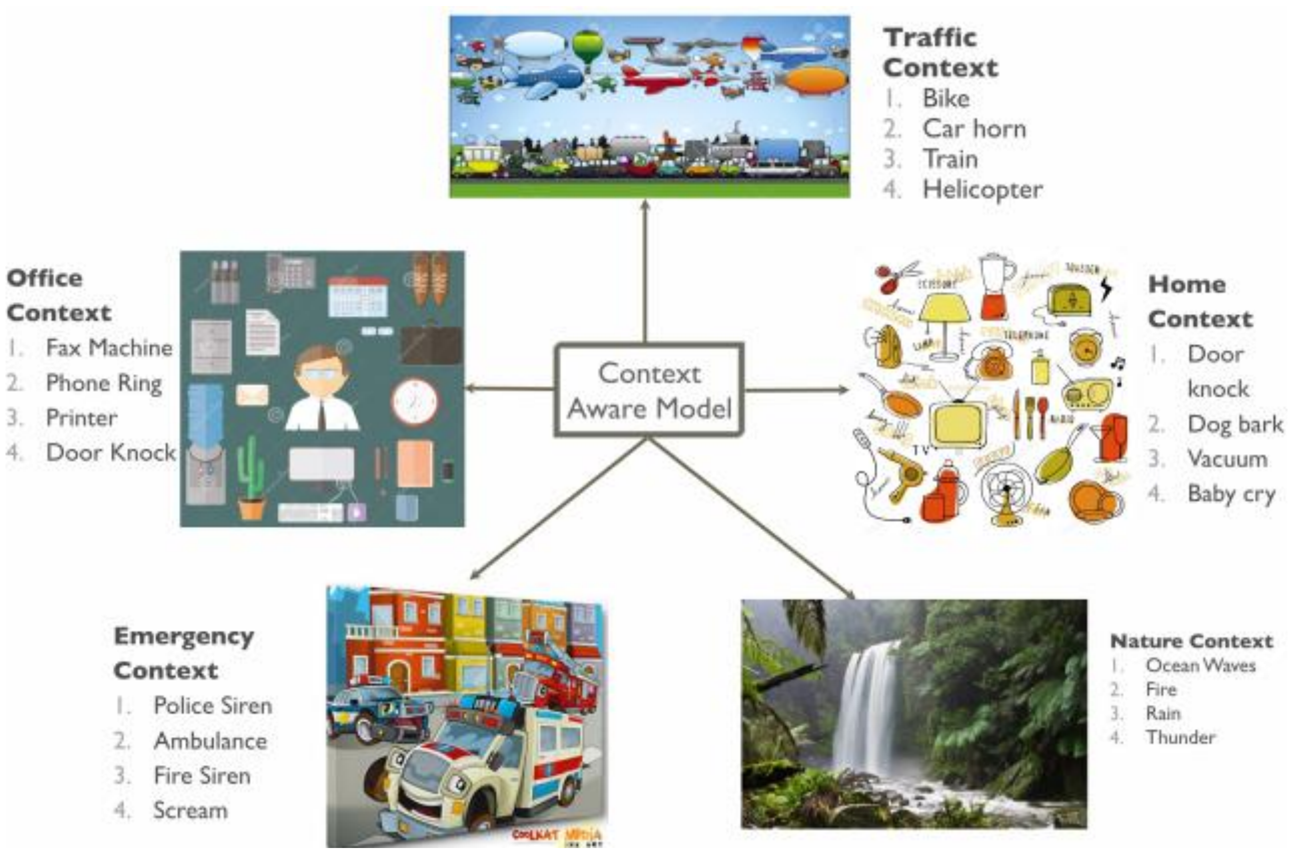


Figure23: Context Aware Models

Context aware systems are a component of ubiquitous computing or pervasive computing environments. CAAR makes the mobile platforms capable:

1. Sensing the users and their current state, exploiting context information to significantly reduce demands on human attention.
2. Minimizing noises and maximizing the recognition accuracy with context-awareness.
3. Scalable and efficient recognition with a minimum set of models for the lightweight Machine Learning engine.

Figure 23 shows, how we have divided our context aware model considering the user location and time.

In our application, we have modeled five important contexts based on the geographical prevalence and time domain of the user. Following are the five contexts we have considered:

Home Context:

This context has all basic household sounds a user can have. This context is chosen when the user is in their house. In our application, we have considered only four sounds of households. Following are the four classes/sounds we considered:

1. Door knock.
2. Vacuum cleaner.
3. Dog bark.
4. Baby cry.

Nature Context:

This context consists of all basic nature sounds. This context is chosen when the user is in their outdoor near beach or forest. In our application, we have considered only four sounds of Nature. Following are the four classes/sounds we considered:

1. Ocean waves.
2. Fire.
3. Rain.
4. Thunder.

Office Context:

This context consists of all basic Office sounds. This context is chosen when the user is in their office. In our application, we have considered only four sounds of office. Following are the four classes/sounds we considered:

1. Fax machines.
2. Phone rings.

3. Printer.

4. Door Knock.

Traffic Context:

This context has all basic traffic sounds we have. This context is applied when the user is moving in traffic. In our application, we have considered only four sounds of traffic. Following are the four classes/sounds we considered:

1. Car horn.

2. Bike exhaust.

3. Train.

4. Helicopter.

Emergency Context:

This context has all basic emergency sounds we have. This context is applied all the time, regardless of user location. In our application, we have considered only four sounds of emergency. Following are the four classes/sounds we considered:

1. Police siren.

2. Ambulance.

3. Fire alarm.

4. Scream.

CHAPTER 4

IMPLEMENTATION AND EXPERIMENT SETUP

4.1 Implementation

For implementation, we have used following

- Python, Scala: Python and Scala are used as programming languages to use Apache Spark MLlib and Self organized map.
- Apache Spark 1.5.2: Apache Spark is used for audio streaming and classification algorithms.
- Mongo DB: Mongo DB is used for storing feature data and user basic information. It acts like a mediator between server and client,

4.2 Experiment Setup

We have conducted and evaluated my experiments on three different machines. We have used machine 1 and machine 2 for Machine Learning i.e., training and testing models. Machine 3 is used for iHear application.

Machine 1:

- Ubuntu 14.04 – we have used Ubuntu 14.04 version. We have used OS X mac platform for building models.
- Memory: 15.6 GB – with 15.6 GB RAM size.
- Processor: Intel® Xeon(R) CPU E3-1230 V2 @ 3.30GHz × 8

Machine 2:

- OS X – we have used OS X mac platform for building models.
- Memory: 8 GB - with 8 GB RAM size.
- Processor: 2.2 GHz Intel Core i7

Machine 3:

- IOS - we have used IOS device for the application.
- Memory: 1 GB - with 1 GB RAM size.

- Processor: Dual-core 1.4 GHz Typhoon (ARM v8-based).

CHAPTER 5

EVALUATIONS

5.I Evaluation Plan

Following is the evaluation plan, we will go through step by step to justify our proposed solutions:

1. Noise Reduction:

In this evaluation we will justify how the noise reduction process works with iHear filters. We performed tests with noise data in this case, it's Vacuum sound. This noise sound is combined with a song, and we will see the resultant sound with reduced noise sound.

2. Feature Extraction:

In feature extraction phase, we will show how audio features look when the audio is captured dynamically from the client.

3. Feature Selection: Accuracy:

Feature selection techniques are used for two main reasons: simplification of models to make them easier to interpret by researchers/users, shorter training times. We will evaluate this section by taking a different feature set and evaluate which feature set is more accurate.

4. Feature Scaling and Normalization: Accuracy:

In this section we will evaluate with normal feature set and normalized feature set, and observe the accuracy for both of them in terms of building models.

5. Audio Classification:

In audio classification, we are evaluating our light weight engine with Context aware models and Signature to other regular approaches to see which one gives better results in build time, accuracy and space. Following is the evaluation for audio classifications.

1. context-aware audio classification:

Based on user location and time, we have differentiated five contexts namely office, house, traffic, nature and animals. For building models for each of the contexts, we have two approaches:

(A) Dynamic Learning/Dynamic Recognition:

In Dynamic Learning/Dynamic Recognition, for training the models we collected real time(Dynamic) data from smartphones and for testing the models we use same Dynamic data collected from smartphones for getting accuracy.

(B) Static Learning/ Dynamic Recognition:

In Static Learning/Dynamic Recognition, for training the models we collected static data which are audio smokes collected from the internet and for testing the models we used real time audio data (Dynamic data) collected from smartphones for getting accuracy.

2. without context audio classification:

Another approach is without using any context, we will evaluate all models together as one here. We have two approaches in building models:

(A) Dynamic Learning/Dynamic Recognition:

In Dynamic Learning/Dynamic Recognition, for training the models we collected real time (Dynamic) data from smartphones and for testing the models we use same Dynamic data collected from smartphones for getting accuracy.

(B) Static Learning/ Dynamic Recognition:

In Static Learning/Dynamic Recognition, for training the models we collected static data which are audio smokes collected from the internet and for testing the models we used real time audio data (Dynamic data) collected from smartphones for getting accuracy.

5.2 Noise Reduction

Figure 24, is the representation of a song data with time domain on x-axis and frequency on Y-axis. As we can see song data is ranging from {-0.2, 0.1}.

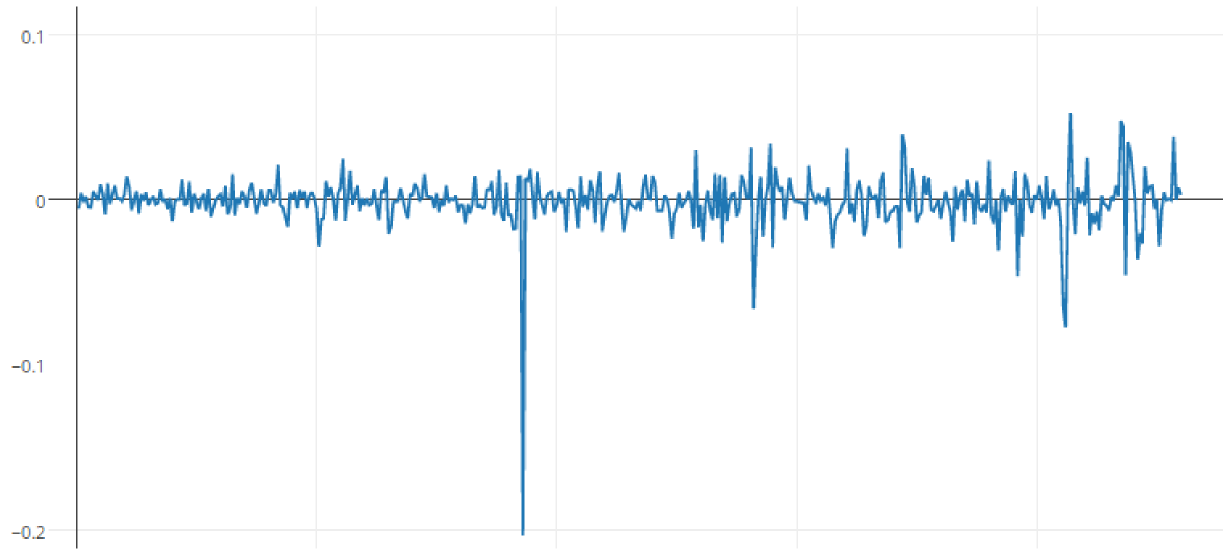


Figure 24: Song data

Figure 25, is the representation of noise data (vacuum sound) with time domain on x-axis and frequency on the Y- axis. From the wave form, we can observe that vacuum sound is ranging from {-0.05, 0.05}.

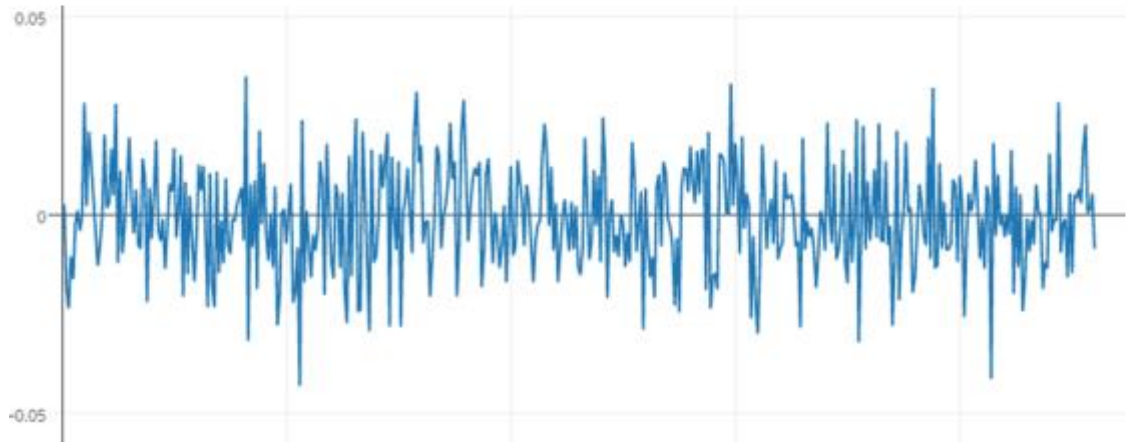


Figure 25: Vacuum sound data.

Figure26, represents noise data (vacuum sound) and song data together, with time domain on x-axis and frequency on the Y- axis. Form the below waveform, we can observe that Sound data with noise data is ranging from {-0.05, 0.1}

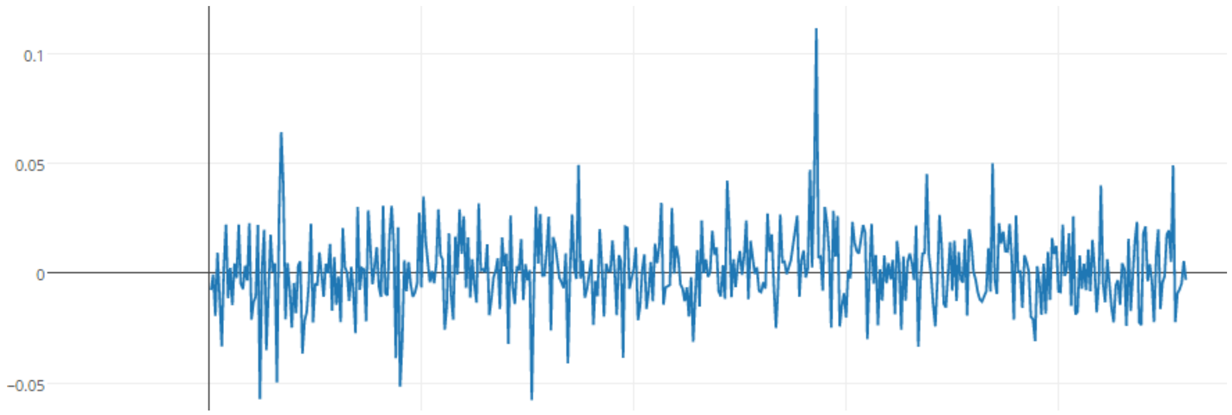


Figure 26: Sound + Vacuum data

Now by applying iHear dynamic Low pass filter at 1400 Hz frequency, we can reduce the noise data(Vacuum) by 60-70%. This can be shown in Figure27, where vacuum data ranged from {-0.05, 0.05} is neglected. Resulted sound ranged from {-0.02, 0.02}.

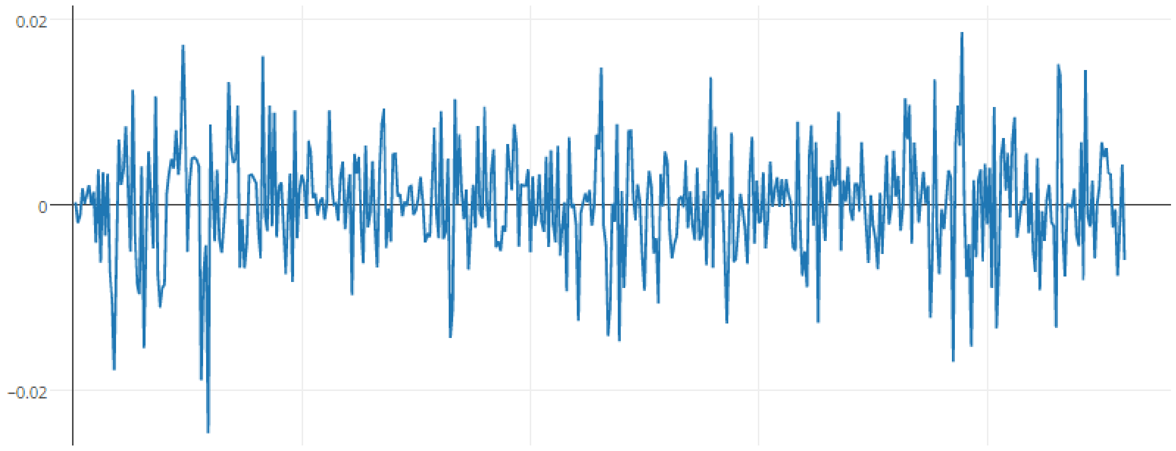


Figure 27: Noise Reduction.

5.3 Phase I - I: Feature Extraction

The Following figures show how each feature looks when plotted on a graph. This graph is the representation of the dynamic audio wave.

Mean:

Mean graph is plotted for three different sounds namely Door bell, police siren and vacuum. With Mean values on Y axis and time(milliseconds) on x axis.

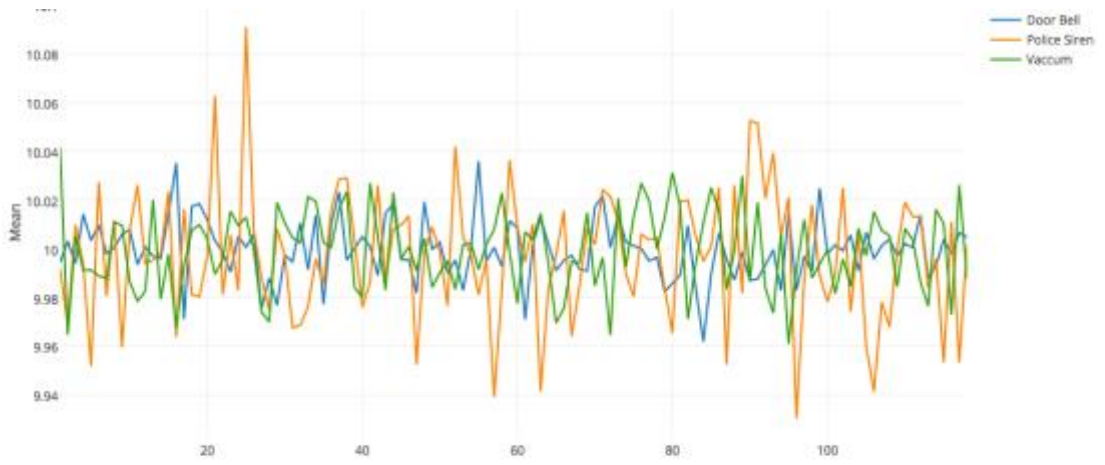


Figure 28: Feature -Mean

Variance: Variance graph is plotted for their sounds namely Door bell, police siren and Vacuum. With Mean values on Y axis and time (milliseconds) on x axis.

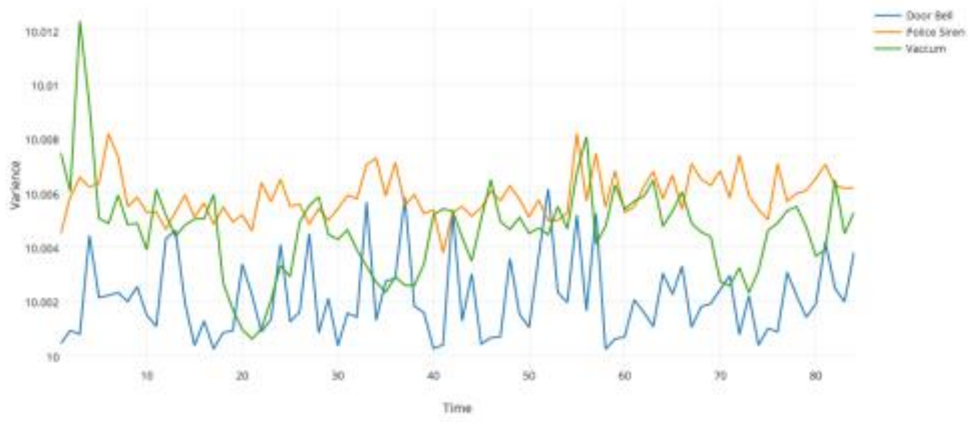


Figure 29: Feature - Variance

Spectral Centroid: Spectral centroid graph is plotted for their sounds namely Door bell, police siren and Vacuum. Spectral centroid values on Y axis and time (milliseconds) on x axis.

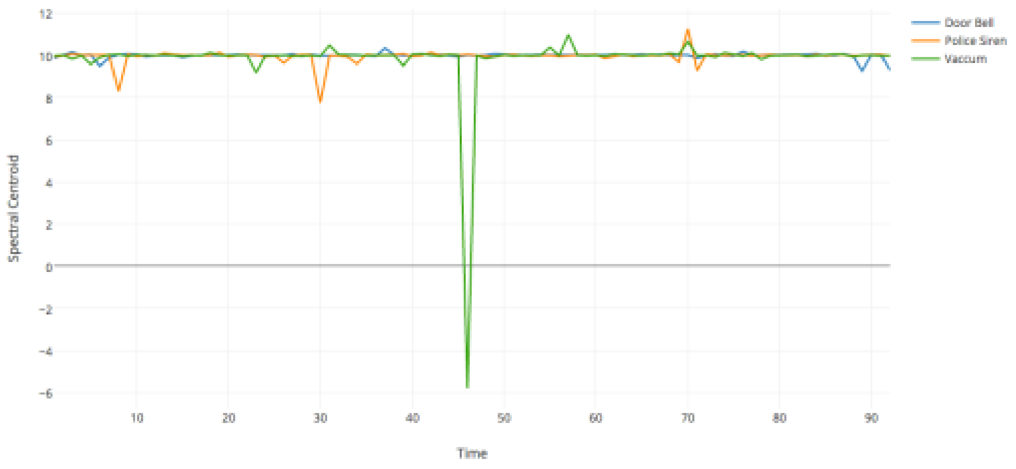


Figure 30: Feature - Spectral Centroid

Skewness: Skewness graph is plotted for their sounds namely Door bell, police siren and Vacuum. With Skewness values on Y axis and time (milliseconds) on x axis.

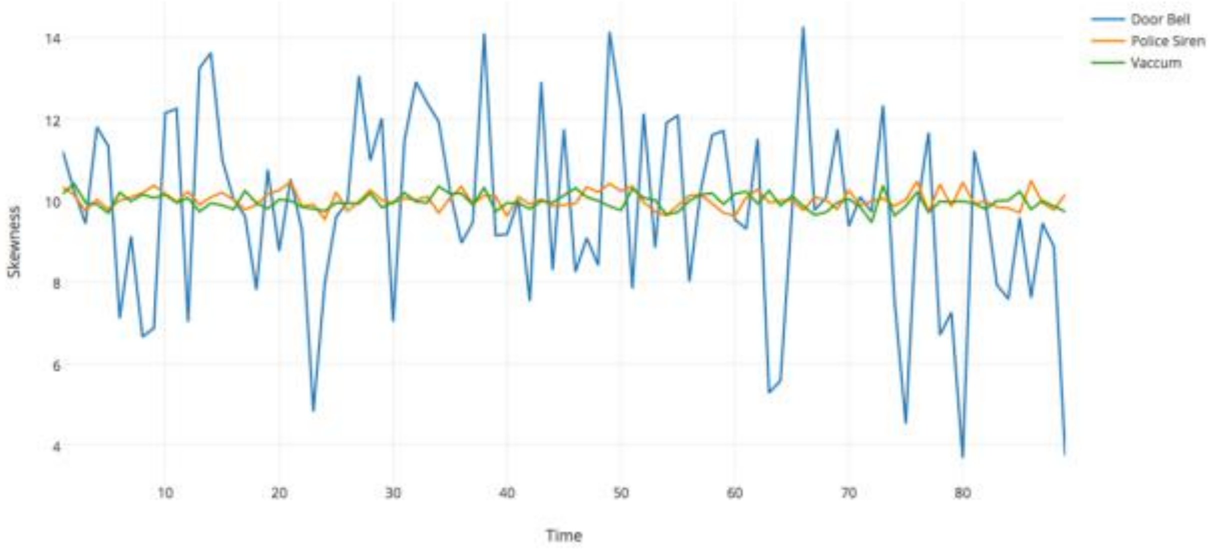


Figure 31: Feature - Skewness

ZCR: ZCR graph is plotted for their sounds namely Door bell, police siren and Vacuum. With ZCR values on Y axis and time (milliseconds) on x axis.

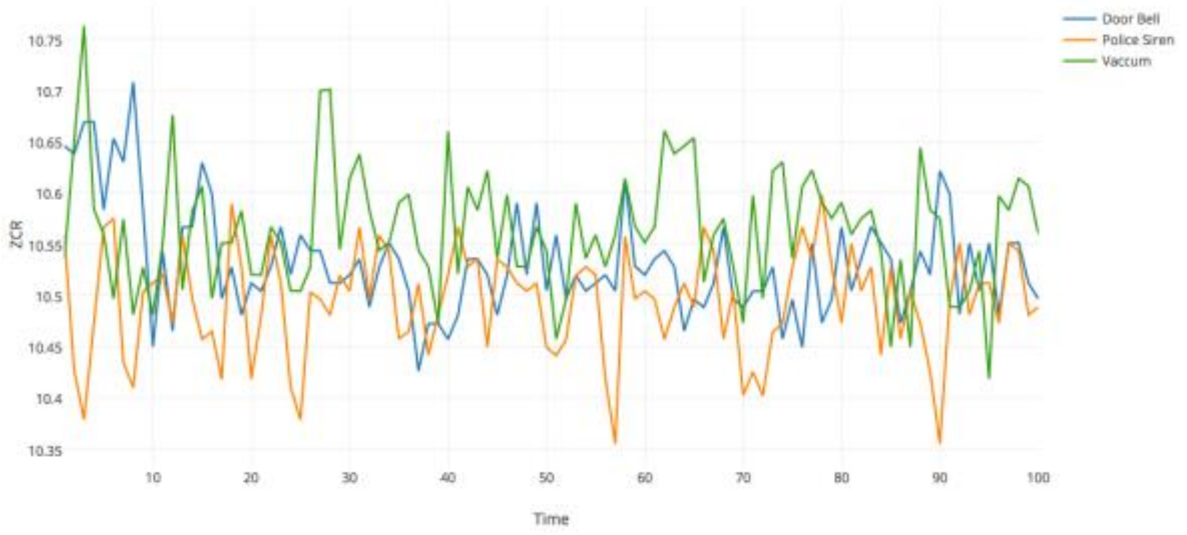


Figure 32: Feature - ZCR

Kurtosis: Kurtosis graph is plotted for their sounds namely Door bell, police siren and Vacuum. With Kurtosis values on Y axis and time (milliseconds) on x axis.

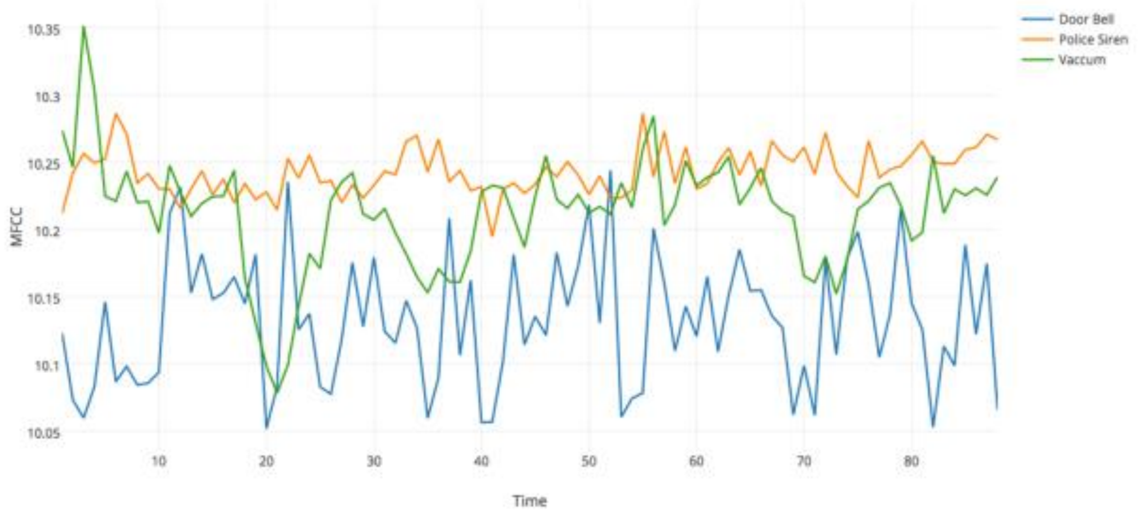


Figure 33: Feature – Kurtosis

Irregularity: Irregularity graph is plotted for their sounds namely Door bell, police siren and Vacuum. With Irregularity values on Y axis and time (milliseconds) on x axis.

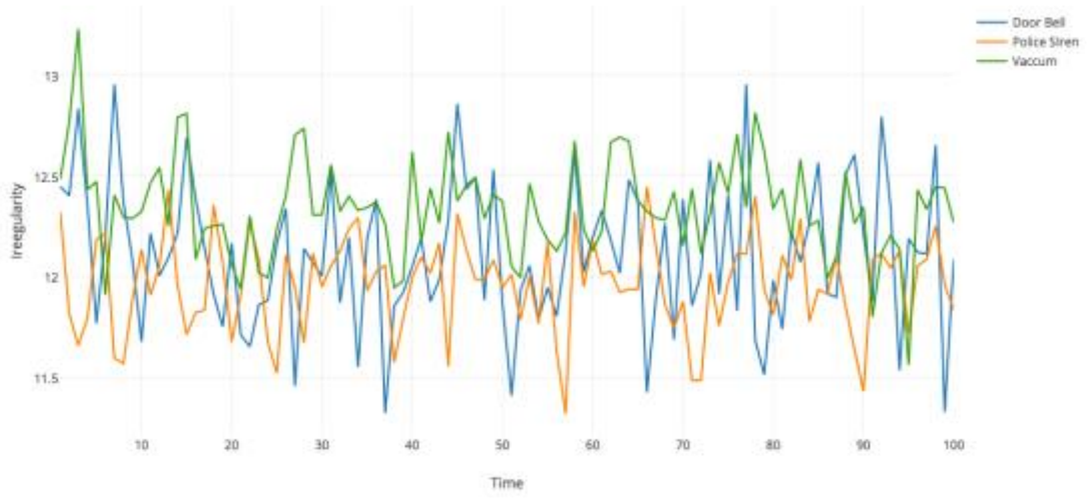


Figure 34: Feature - Irregularity

MFCC: MFCC graph is plotted for their sounds namely Door bell, police siren and Vacuum. With MFCC values on Y axis and time (milliseconds) on x axis.

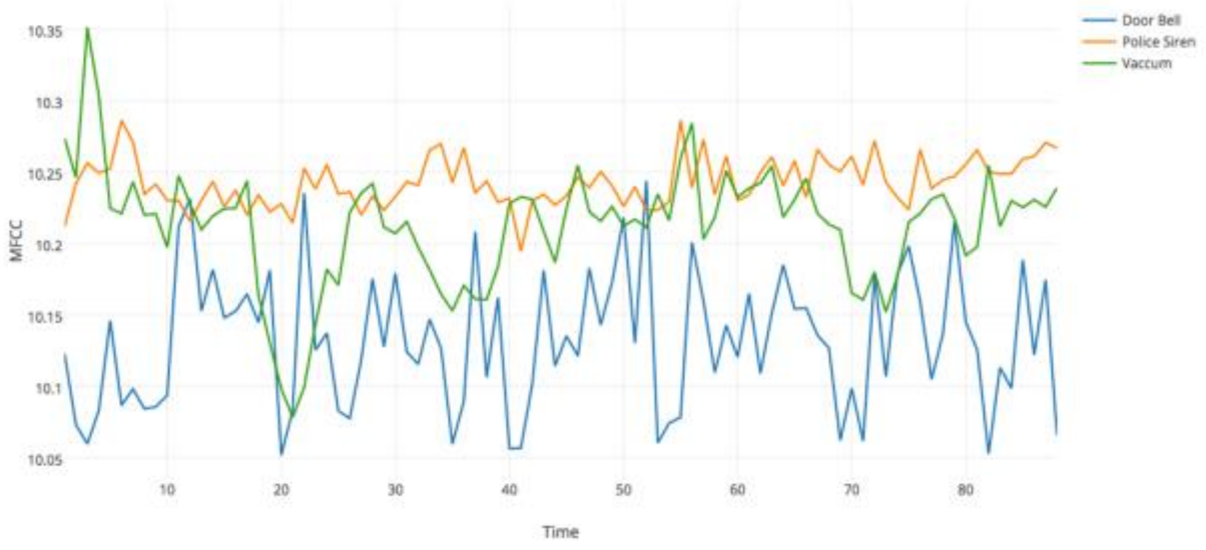


Figure 35: Feature - MFCC

5.4 Phase I - II: Feature Selection: Test Cases

In this evaluation we will find the best set of features for best accuracy. Following is the table with different test cases with different set of features. Test 1 has 27 features, test 2 has 16 features, test 3 has 12 features and test 4 has 9 features. In the next sections we will we discuss about accuracy for each test case chosen, and find the best fit.

Test 1	27	Mean, Standard Deviation, Variance, Skewness, Kurtosis, Spectral Mean, Spectral Variance, Spectral Standard Deviation, Spectral Skewness, Spectral Kurtosis, Spectral Centroid, MFCC, Irregularity_k, Irregularity_j, Tristimulus_1, Tristimulus_2, Tristimulus_3, Smoothness, Spread, ZCR, Rolloff, Loudness, Flatness, Tonality, Breast, Lowest_Value, Highest Value
Test 2	16	Mean, Standard Deviation, Variance, Spectral centroid, Spectral Skewness, ZCR, MFCC, Loudness, kurtosis , irregularity_k, irregularity_j, Tristimulus_2, skewness, highest_value, Rolloff, Tonality
Test 3	12	Mean, Standard Deviation, Variance, Spectral centroid, ZCR, Loudness, kurtosis , irregularity_k, irregularity_j, skewness, highest_value, MFCC
Test 4	9	Mean, Variance, Spectral centroid, Loudness, kurtosis , irregularity_j, skewness, highest_value, MFCC

5.4.1 Feature Selection: Accuracy

Figure 36, shows the accuracy for all four test cases choose in the feature selection evaluation process. From the results, we can see that test 3 with a feature set [Mean, Standard Deviation, Variance, Spectral centroid, ZCR, Loudness, kurtosis, irregularity_k, irregularity_j, skewness, highest_value, MFCC] has high accuracy 89% than remaining three test cases. Test case 1 has an accuracy of 52% for decision tree and 51% of random forest, test case 2 has an accuracy of 68% for decision tree and 70%

of random forest, test case 4 has an accuracy of 74% for decision tree and 69% of random forest. Which are less than test case 2 accuracy. Our engine chooses test case 3 feature set as for building models.

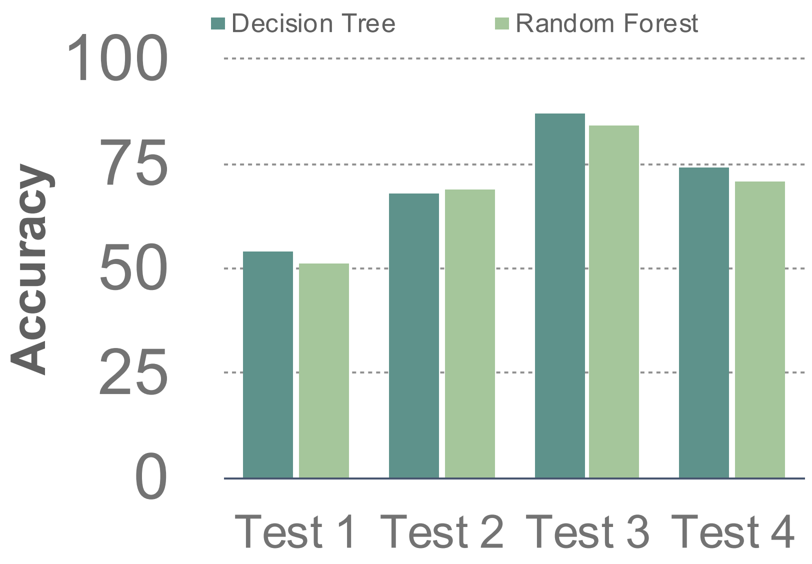


Figure 36: Feature Selection - accuracy

5.5 Phase 1-3: Data Normalization - Accuracy

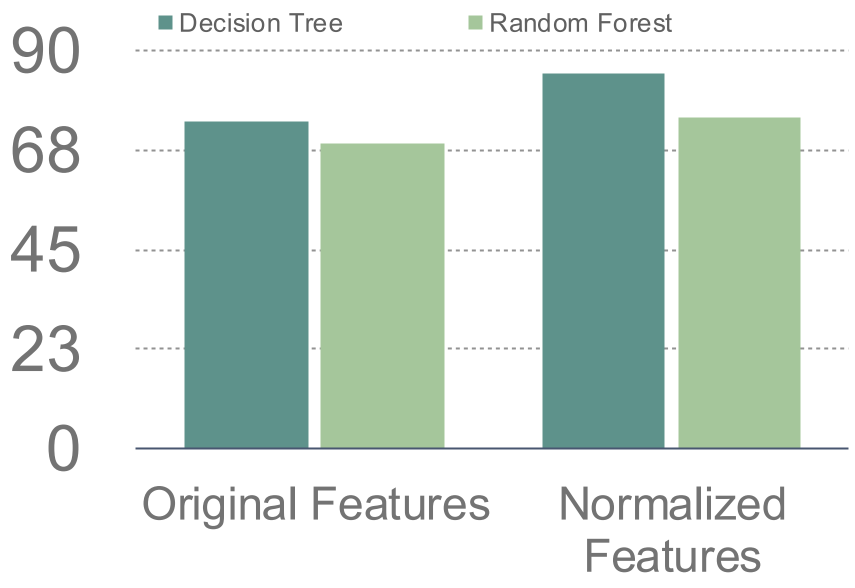


Figure 37: Data Normalization - accuracy

Figure 37, shows the accuracy of models for both original features and normalized features. With original feature data set, each features is ranged from 0 to 49999. In normalized feature data set each feature is ranged from 11 to 14.

5.6 Phase II-III: Audio Classification Evaluation

In this audio classification evaluation. We are considering two level hierarchy. The First level is done on the client, we have Learned with static data and learning with dynamic data. Each of these learnings has two approaches Feature Approach and Signature Based Approach. Each of these approaches has two process - 'Audio recognition with contexts' (AR) and 'Context Aware audio recognition'(CAAR). This complete hierarchy is showed in figure 38.

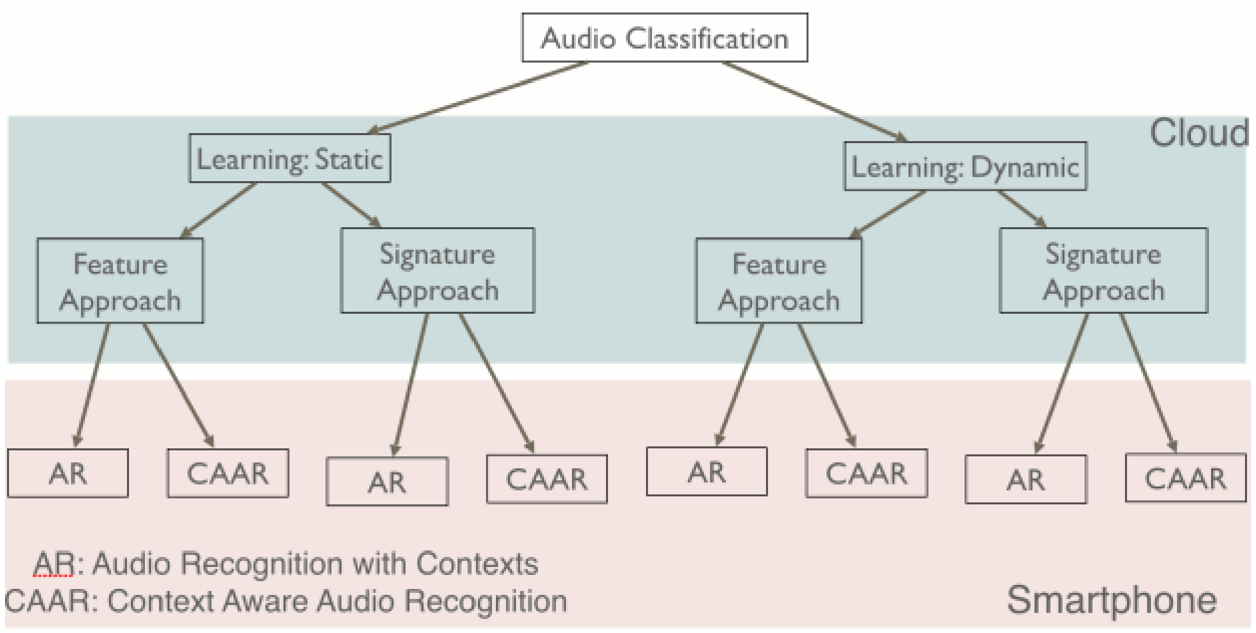


Figure 38: Audio Classification Hierarchy

5.7 Context Aware Audio Classification

5.7.1 Context Aware Audio Classification: Dynamic Learning/ Dynamic Recognition

Context Aware Audio Classification: Household

In figure 39, we are evaluating Household context model accuracy with all feature data set and clusters featured data set ranging from 650 to 100. For example, as house hold context has four classes - Door knock, baby cry, vacuum sound, Dog bark. Each having a feature data count of 1250. So total feature count of the Household context is 1250 * 4 = 6000 data points. Signature (cluster: 650) means data points for each class are clustered in 650 from 1250 original features. Now Signature (cluster: 650) has total data set of = 650 * 4 = 2600 data points. Signature (cluster: 400) means data points for each class are clustered to 400 from 1250 original features. Now Signature (cluster: 400) has total data set of = 400 * 4 = 1600 data points. Signature (cluster: 200) means data points for each class are clustered in to 2000 from 1250 original features. Now Signature (cluster: 200) has total data set of = 200 * 4 = 800 data points. Signature (cluster: 100) means data points for each class are clustered in to 100 from 1250 original features. Now Signature (cluster: 100) has a data set as = 100 * 4 = 400 data points.

From figure 39, We can observe that house hold context model with total feature has an accuracy of 82% for decision tree and 81 % for random forest. Whereas Signature (cluster: 650) features has an accuracy of 80% for decision tree and 78 % for random forest. Signature (cluster: 400) features has an accuracy of 80% for decision tree and 79 % for random forest. Signature (cluster: 200) features has an accuracy of 82% for decision tree and 74 % for random forest. Signature (cluster: 100) features has an accuracy of 73% for decision tree and 63 % for random forest.

From this evaluation we can say that, Signature(cluster: 200) has the same accuracy as that of total features. From figure 40, Signature(cluster: 200) has a build time of 53 sec and data points count as 800 where as Total feature has a build time of 70 sec and data points count as 6000. So by using signature approach we have reduced reduced the build time from 70 sec to 53 sec and data points count from 6000 to 800 which is 84% size drop.

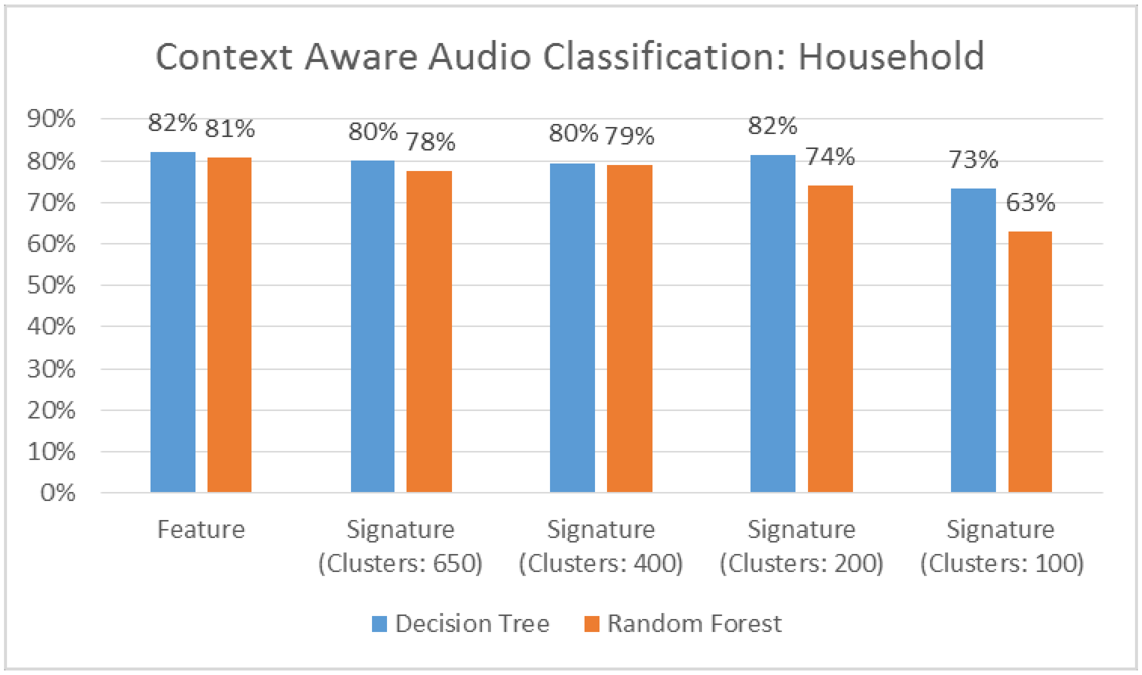


Figure 39: Context Aware Audio Classification Household-accuracy

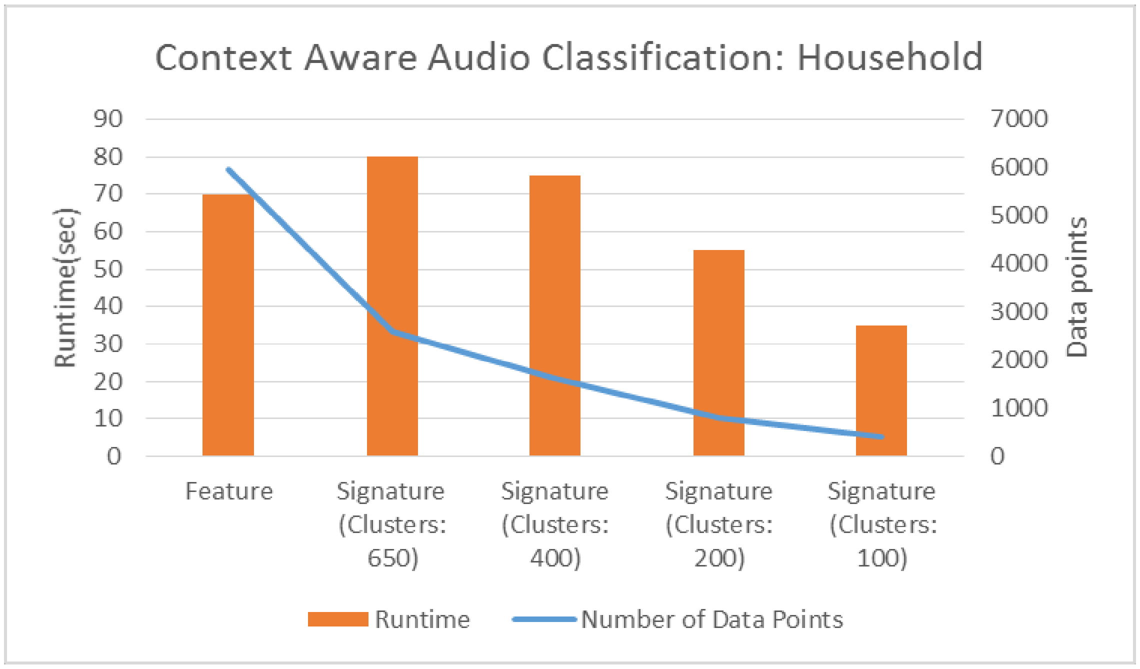


Figure 40: Context Aware Audio Classification Household-time/space

Context Aware Audio Classification: Nature

In figure 41, we are evaluating Nature context model accuracy with all feature data seta and clusters featured data set ranging from 650 to 100. From figure 41, we can observe that nature context model with total feature has accuracy of 81% for decision tree and 79% for random forest. Whereas Signature (cluster: 650) features has an accuracy of 81% for decision tree and 74% for random forest. Signature (cluster: 400) features has an accuracy of 80% for decision tree and 79% for random forest. Signature (cluster: 200) features has an accuracy of 76% for decision tree and 74% for random forest. Signature (cluster: 100) features has an accuracy of 76% for decision tree and 73% for random forest.

Form the figure 42, we can observe that total feature data set took build time as 45 sec and data points count as 5000. Signature (Cluster: 650) took build time as 55 sec and data points count as 2800. Signature (Cluster: 400) took build time as 50 sec and data points count as 1800. Signature (Cluster: 200) took build time as 35 sec and data points count as 800. Signature (Cluster: 100) took build time as 25 sec and data points count as 400.

Form this evaluation we have two observations, if we want less build time and less data points, we can choose Signature(Cluster: 200) or Signature(Cluster: 100) approach as both has less build time and less data size(20% of original feature data count). If we consider accuracy we can go for Signature (Cluster: 650) as it has same accuracy 81% compared with total feature data accuracy. It’s a tradeoff between accuracy, time/size in this scenario.

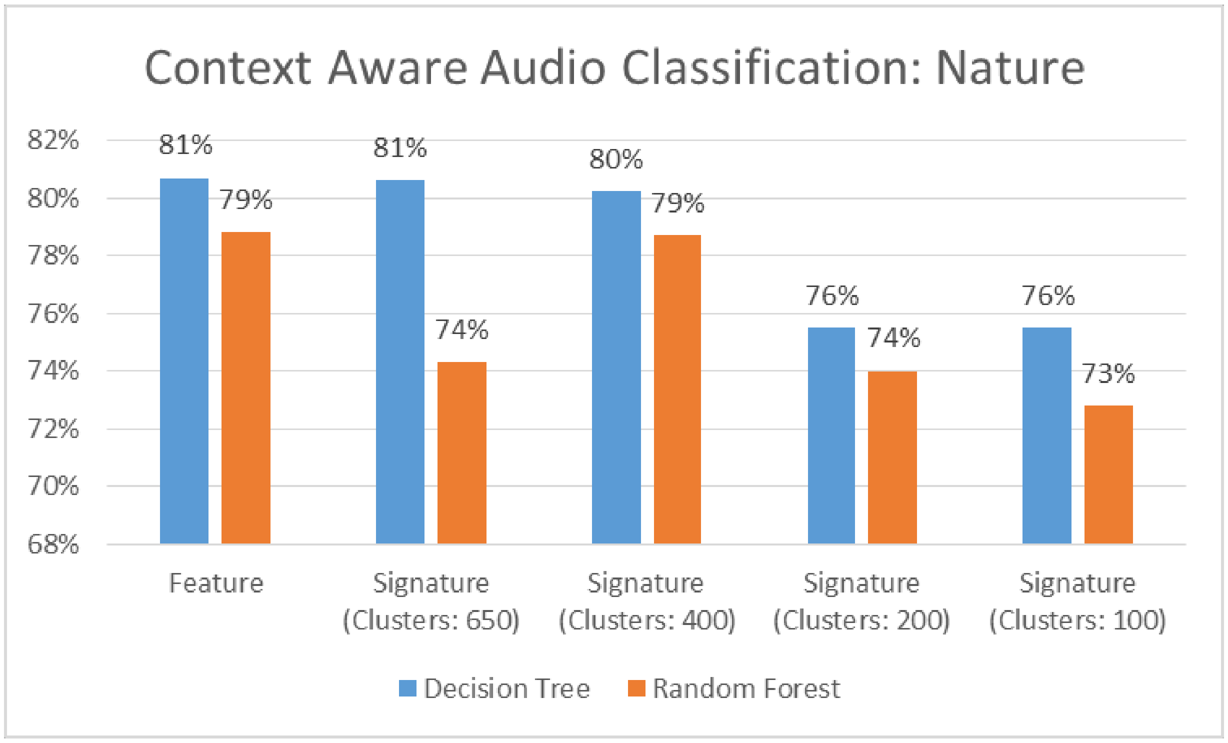


Figure 41: Context Aware Audio Classification Nature- Accuracy

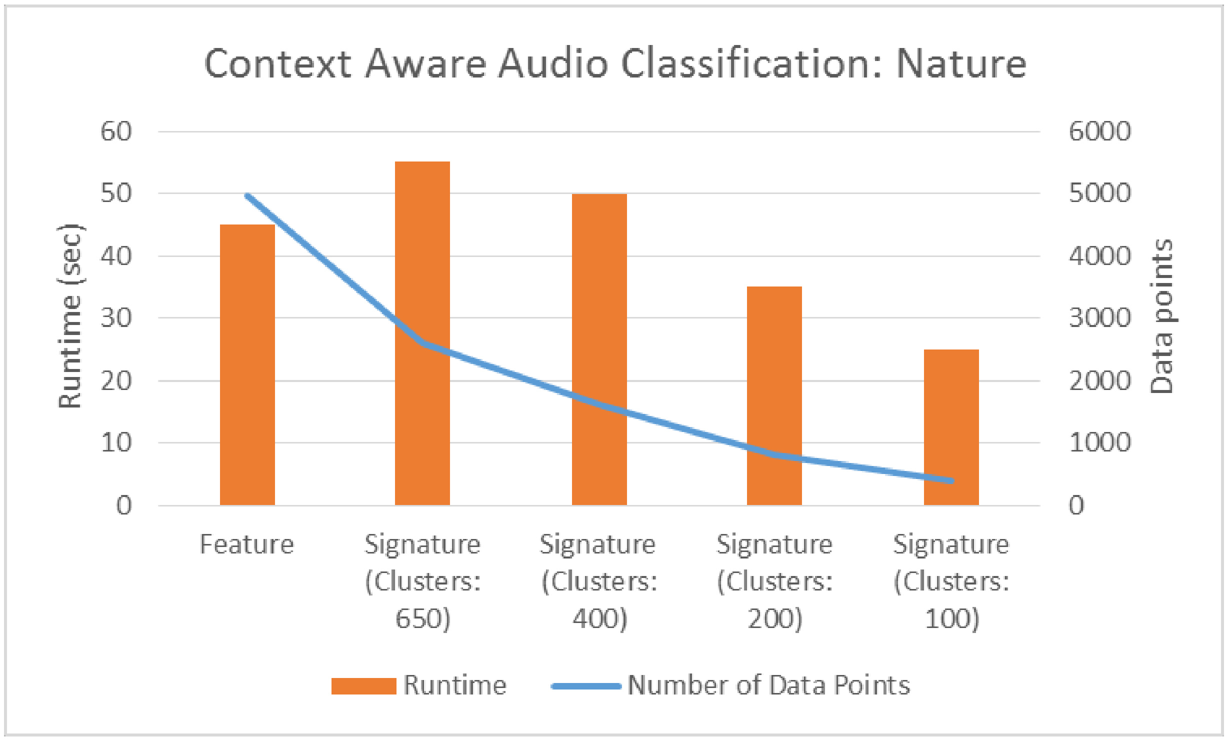


Figure 42: Context Aware Audio Classification Nature-time/space

Context Aware Audio Classification: Emergency

In figure 43, we are evaluating Emergency context model accuracy with all feature data seta and clusters featured data set ranging from 650 to 100. From figure 43, we can observe that Emergency context model with total feature has an accuracy of 90% for decision tree and 87% for random forest. Whereas Signature (cluster: 650) features has an accuracy of 90% for decision tree and 88% for random forest. Signature (cluster: 400) features has an accuracy of 91% for decision tree and 89% for random forest. Signature (cluster: 200) features has an accuracy of 90% for decision tree and 89% for random forest. Signature (cluster: 100) features has an accuracy of 84% for decision tree and 81% for random forest.

From figure 44, we can observe that total feature data set took build time as 50 sec and data points count as 6000. Signature (Cluster: 650) took build time as 55 sec and data points count as 2200. Signature (Cluster: 400) took build time as 48 sec and data points count as 1500. Signature (Cluster: 200) took build time as 34 sec and data points count as 800. Signature (Cluster: 100) took build time as 25 sec and data points count as 400.

From this evaluation we can say that, Signature (cluster: 400) has more accuracy that of total features accuracy. Even build time has reduce from 50 sec to 48 sec, and data point count has reduce from 6000 to 1500 (25% of total size).

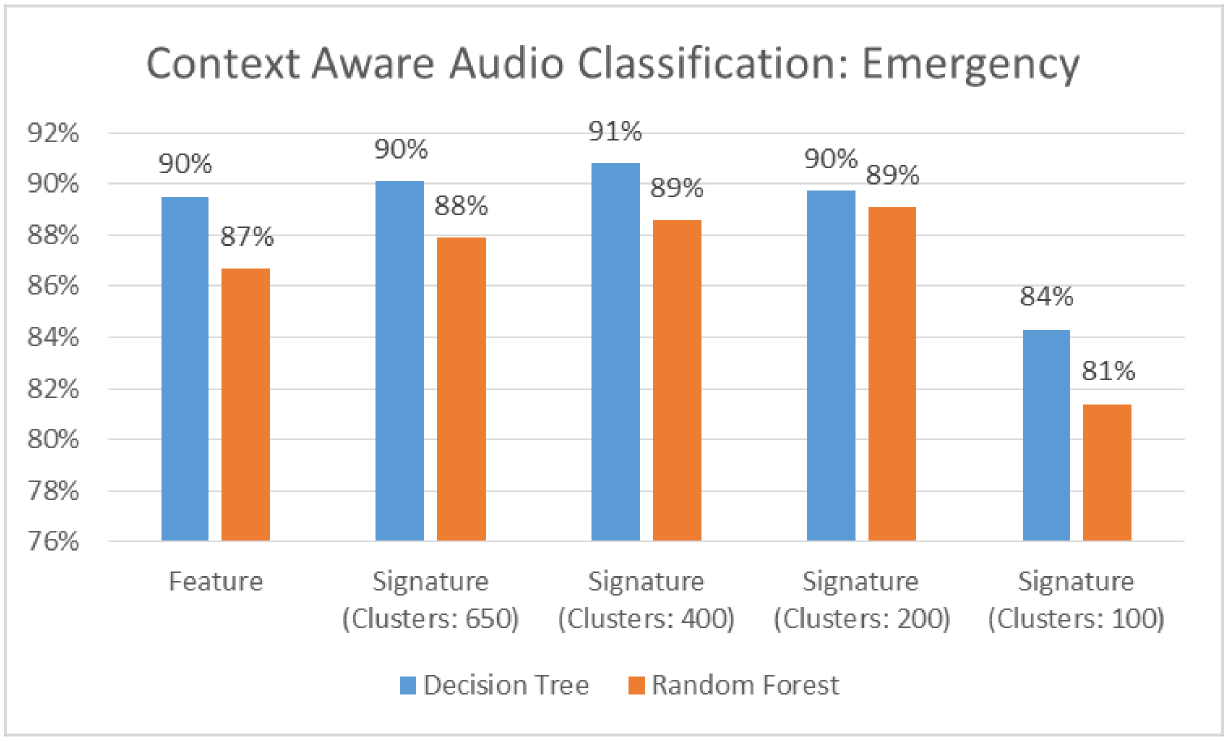


Figure 43: Context Aware Audio Classification Emergency- Accuracy

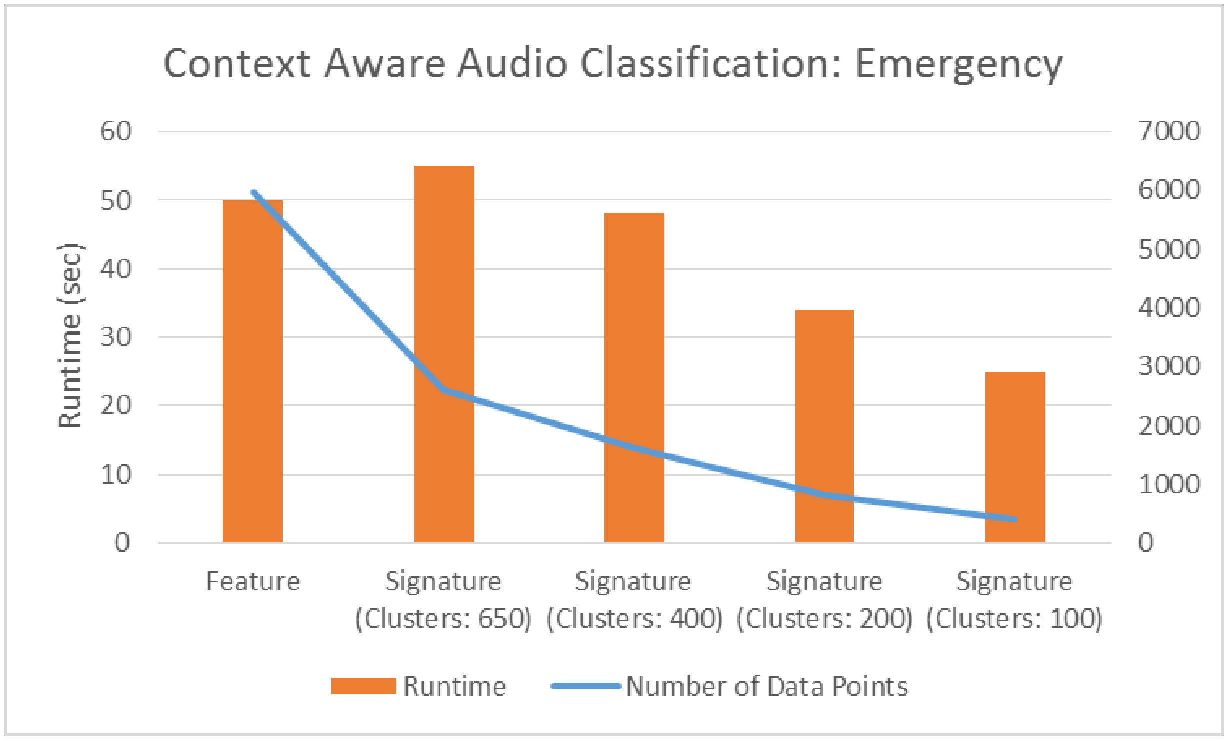


Figure 44: Context Aware Audio Classification Nature-time/space

Context Aware Audio Classification: Office

In figure 45, we are evaluating Office context model accuracy with all feature data seta and clusters featured data set ranging from 650 to 100. From figure 45, we can observe that Emergency context model with total feature has an accuracy of 88% for decision tree and 85% for random forest. Whereas Signature (cluster: 650) features has an accuracy of 89% for decision tree and 86% for random forest. Signature (cluster: 400) features has an accuracy of 85% for decision tree and 84% for random forest. Signature (cluster: 200) features has an accuracy of 82% for decision tree and 80% for random forest. Signature (cluster: 100) features has an accuracy of 74% for decision tree and 71% for random forest.

From figure 46, we can observe that total feature data set took build time as 45 sec and data points count as 5000. Signature (Cluster: 650) took build time as 50 sec and data points count as 2800. Signature (Cluster: 400) took build time as 48 sec and data points count as 1500. Signature (Cluster: 200) took build time as 34 sec and data points count as 800. Signature (Cluster: 100) took build time as 25 sec and data points count as 400.

From this evaluation we have two observations, if we are concerned about build time and data point size, we can choose Signature (Cluster: 200) or Signature(Cluster: 100) approach as both has less build time and less data size(20% of original feature data count). If we concern about accuracy we can go for Signature (Cluster: 650) as it has more accuracy 89% compared with total feature data accuracy 88%. It’s a tradeoff between accuracy, time/size in this scenario.

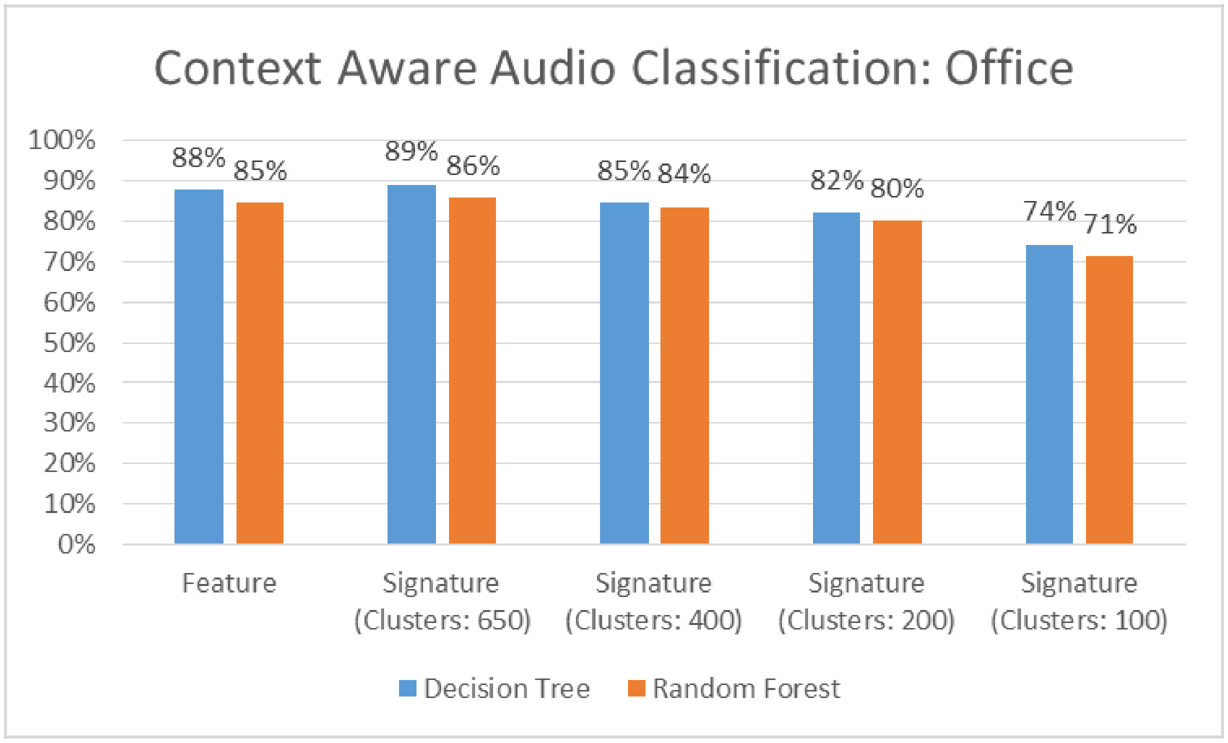


Figure 45: Context Aware Audio Classification Office- Accuracy

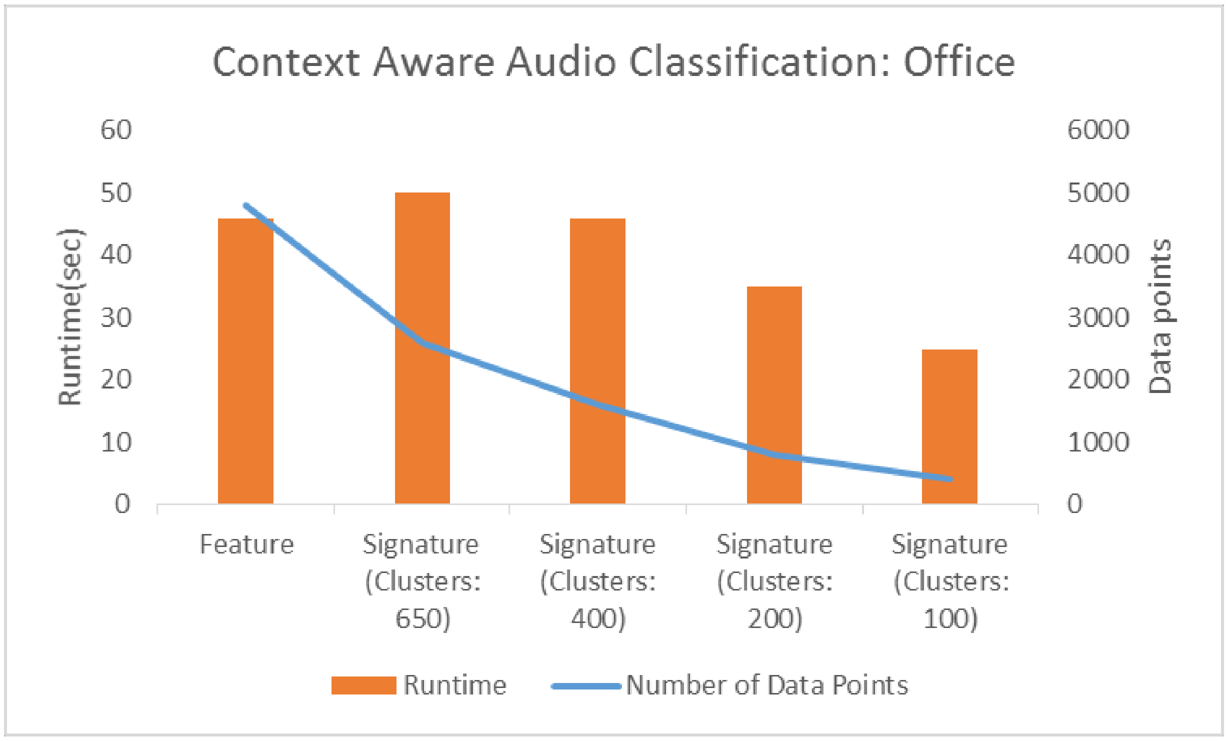


Figure 46: Context Aware Audio Classification Office-time/space

Context Aware Audio Classification: Traffic

In figure 47, we are evaluating Traffic context model accuracy with all feature data seta and clusters featured data set ranging from 650 to 100. From figure 47, we can observe that Traffic context model with total feature has an accuracy of 82% for decision tree and 80% for random forest. Whereas Signature (cluster: 650) features has an accuracy of 90% for decision tree and 89% for random forest. Signature (cluster: 400) features has an accuracy of 90% for decision tree and 85% for random forest. Signature (cluster: 200) features has an accuracy of 88% for decision tree and 89% for random forest. Signature (cluster: 100) features has an accuracy of 83% for decision tree and 85% for random forest.

From figure 48, we can observe that total feature data set took build time as 70 sec and data points count as 13000. Signature (Cluster: 650) took build time as 55 sec and data points count as 2800. Signature (Cluster: 400) took build time as 48 sec and data points count as 2200. Signature (Cluster: 200) took build time as 38 sec and data points count as 1200. Signature (Cluster: 100) took build time as 25 sec and data points count as 800.

From this evaluation we can say that, Signature (cluster: 400) has more accuracy 90% that of total features accuracy 82%. Even build time has reduce from 70 sec to 48 sec, and data point count has reduce from 12000 to 2200 (16% of total size).

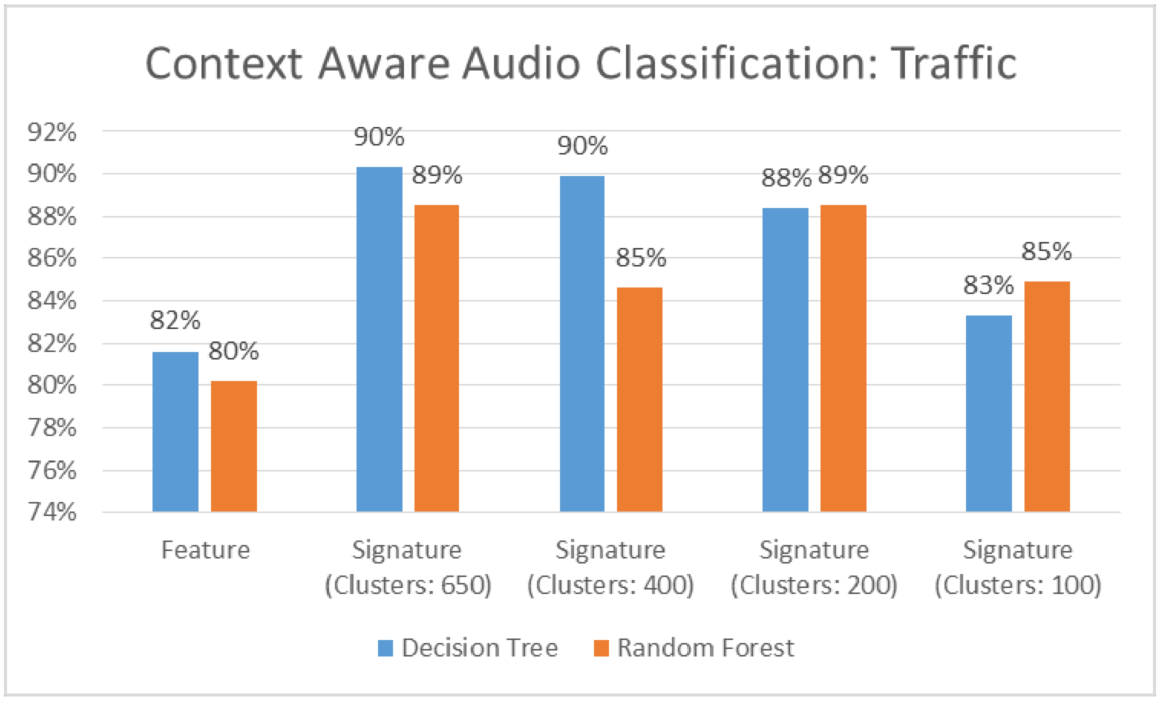


Figure 47: Context Aware Audio Classification Traffic- Accuracy

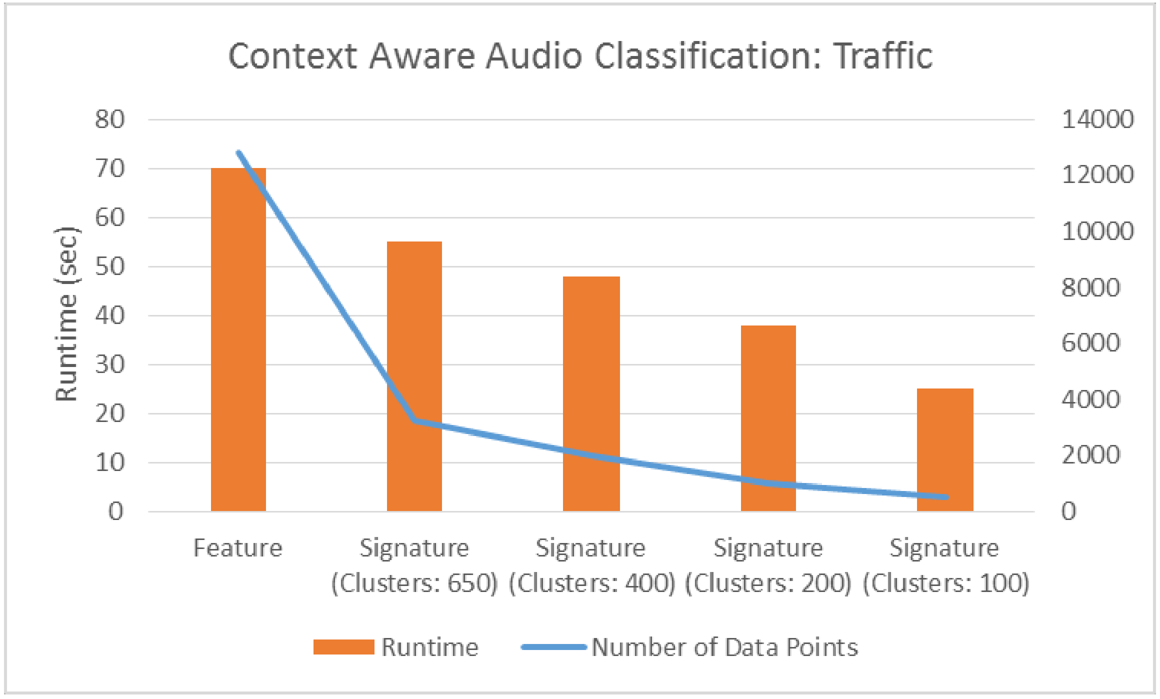


Figure 48: Context Aware Audio Classification Office-time/space

5.7.2 Context Aware Audio Classification: Static Learning/ Dynamic Recognition

Context Aware Audio Classification: Household

From Figure 49, we can observer that static learning and static recognition with all features has an accuracy as 71% for decision tree and 68% for random forest. Static learning and static recognition with Signature (Cluster: 20) has an accuracy as 65% for decision tree and 60% or random forest. But Static leaning and Dynamic recognition has very low accuracy as 43% without Signature, with signature it has even low accuracy as 35% for decision tree and 30% for random forest.

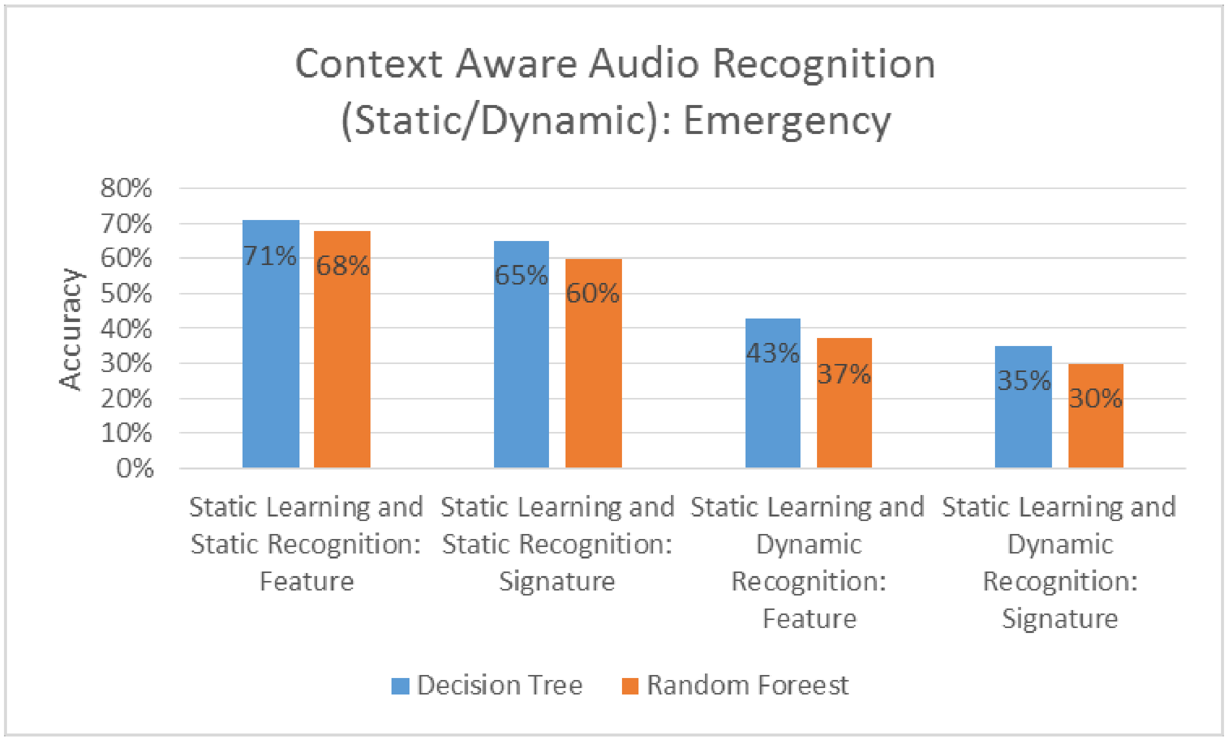


Figure 49: Context Aware Audio Recognition (Static/Dynamic): Household -accuracy

Figure50 shows the build time of the models and data point size. Signature (Cluster: 20) has a build time of 120 sec and data point size of 80. Whereas feature without signature had a build time of 160 sec and data point size of 160. Which are way more than our proposed approach. Even our prosed approach has great accuracy than this approach.

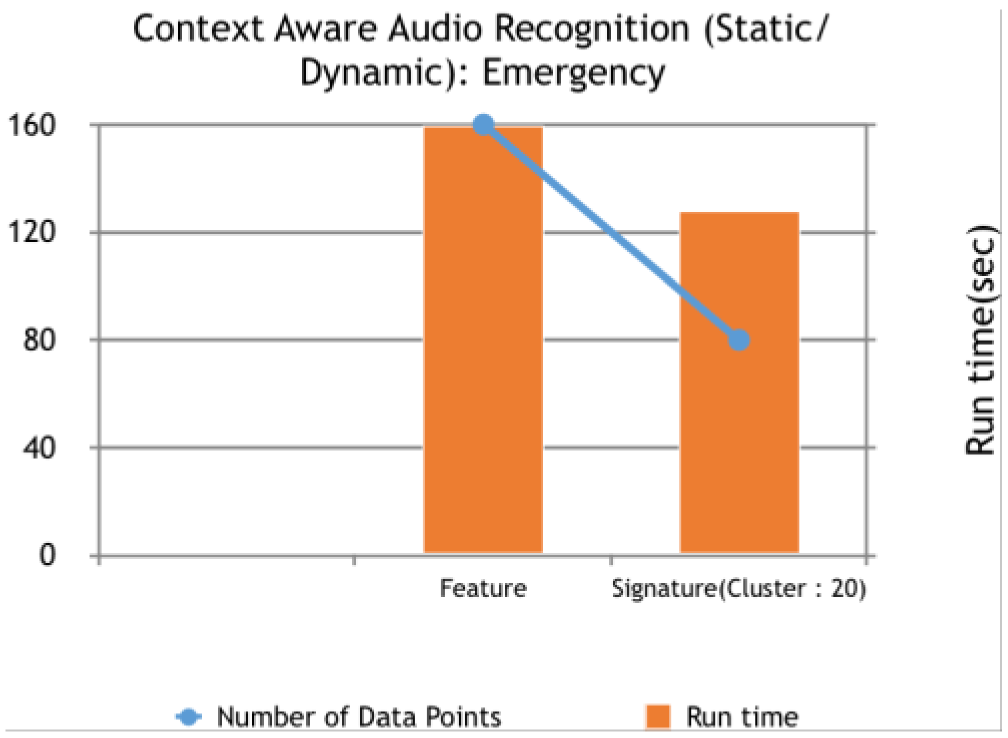


Figure 50: Context Aware Audio Recognition(Static/Dynamic): Household - Runtime

Context Aware Audio Classification: Emergency

From figure 51, we can observer that static learning and static recognition with all features has an accuracy as 53% for decision tree and 54% for random forest. Static learning and static recognition with Signature (Cluster: 20) has an accuracy as 51% for decision tree and 49% or random forest. But Static leaning and Dynamic recognition has very low accuracy as 19% without Signature for decision tree and 21% for random forest, with signature it has even low accuracy as 20% for decision tree and 18% for random forest.

Figure 52 shows the build time of the models and data point size. Signature (Cluster: 20) has a build time of 120 sec and data point size of 80. Whereas feature without signature had a build time of 160 sec and data point size of 160. Which are way more than our proposed approach. Even our prosed approach has great accuracy than this approach.

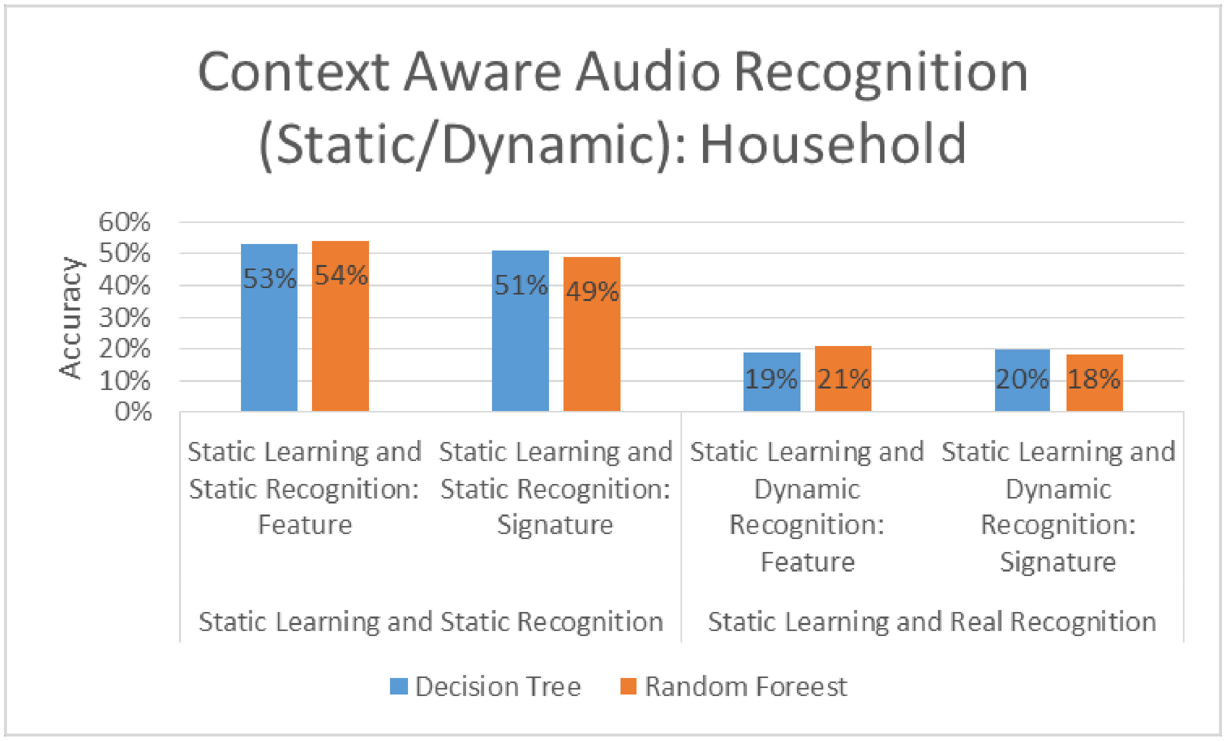


Figure 51: Context Aware Audio Recognition(Static/Dynamic): Emergency –accuracy

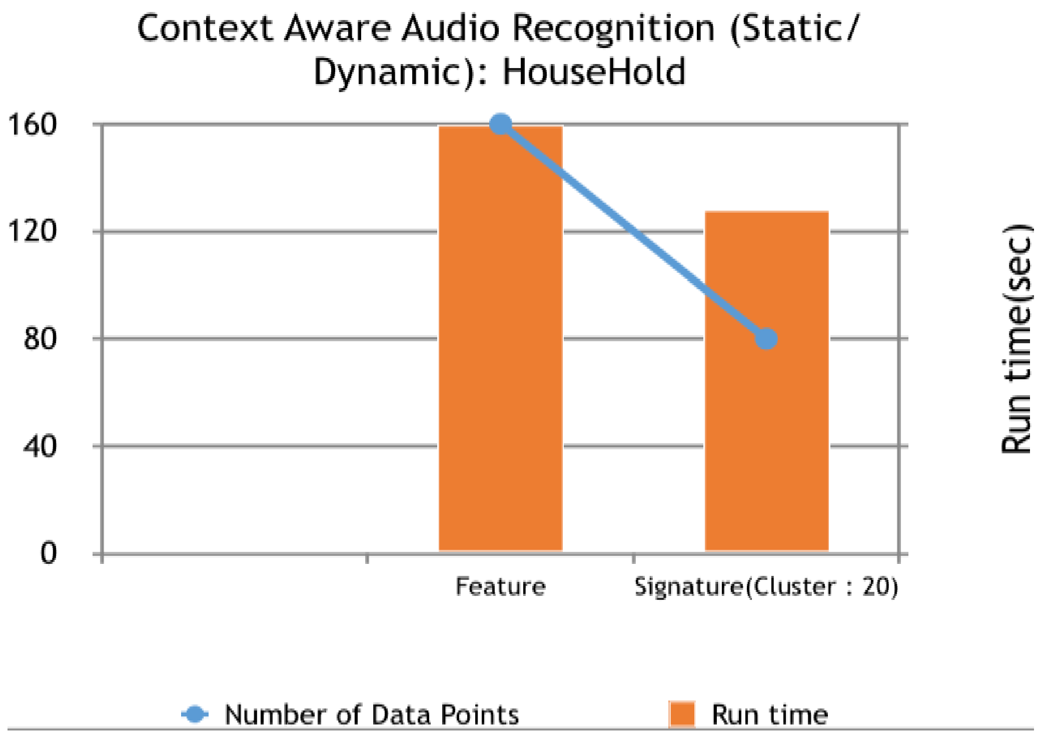


Figure 52: Context Aware Audio Recognition(Static/Dynamic): HouseHold - Runtime

Context Aware Audio Classification: Animals

From figure 53, we can observe that static learning and static recognition with all features has an accuracy as 90% for decision tree and 81% for random forest. Static learning and static recognition with Signature (Cluster: 20) has an accuracy as 81% for decision tree and 73% or random forest. But Static leaning and Dynamic recognition has very low accuracy as 53% without Signature for decision tree and 47% for random forest, with signature it has even low accuracy as 45% for decision tree and 40% for random forest.

Figure 54, shows build time of the models and data point size. Signature (Cluster: 20) has a build time of 120 sec and data point size of 80. Whereas feature without signature had a build time of 160 sec and data point size of 160. Which are way more than our proposed approach. Even our prosed approach has great accuracy than this approach.

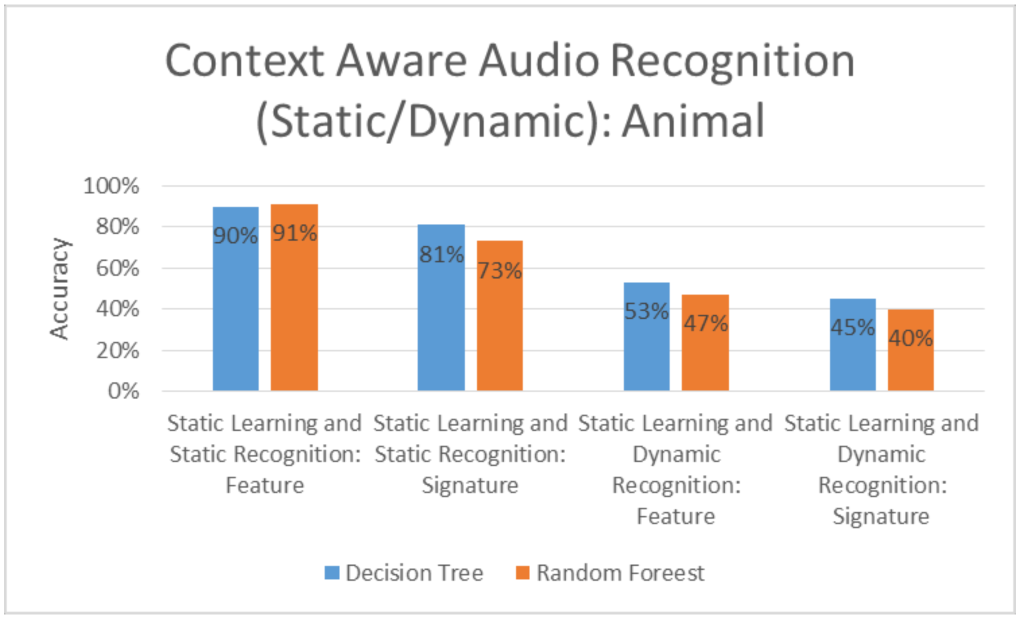


Figure 53: Context Aware Audio Recognition(Static/Dynamic): Animals - Accuracy

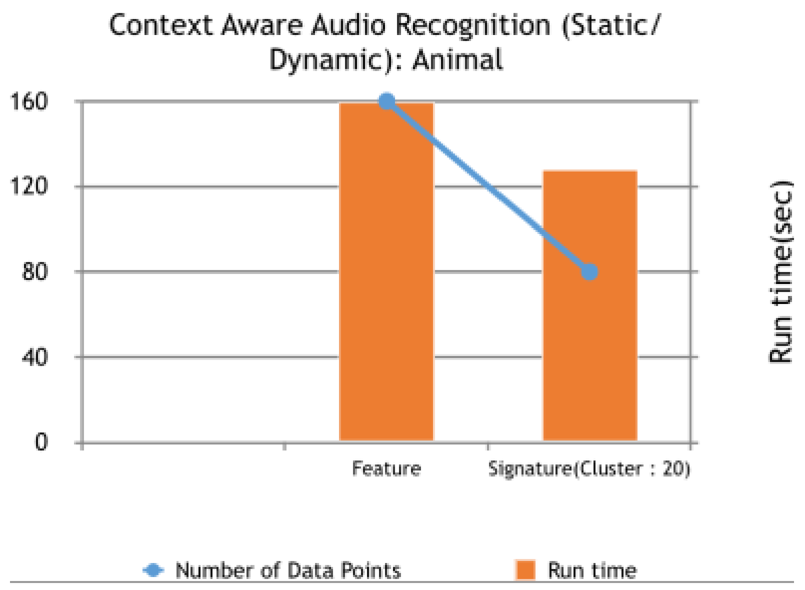


Figure 54: Context Aware Audio Recognition(Static/Dynamic): Animals - Runtime

Context Aware Audio Classification: Nature

From figure 55, we can observer that static learning and static recognition with all features has an accuracy as 65% for decision tree and 60% for random forest. Static learning and static recognition with Signature (Cluster: 20) has an accuracy as 63% for decision tree and 65% or random forest. But Static leaning and Dynamic recognition has very low accuracy as 38% without Signature for decision tree and 33% for random forest, with signature it has even low accuracy as 34% for decision tree and 35% for random forest.

Figure 56, shows build time of the models and data point size. Signature (Cluster: 20) has a build time of 120 sec and data point size of 80. Whereas feature without signature had a build time of 160 sec and data point size of 160. Which are way more than our proposed approach. Even our prosed approach has great accuracy than this approach.

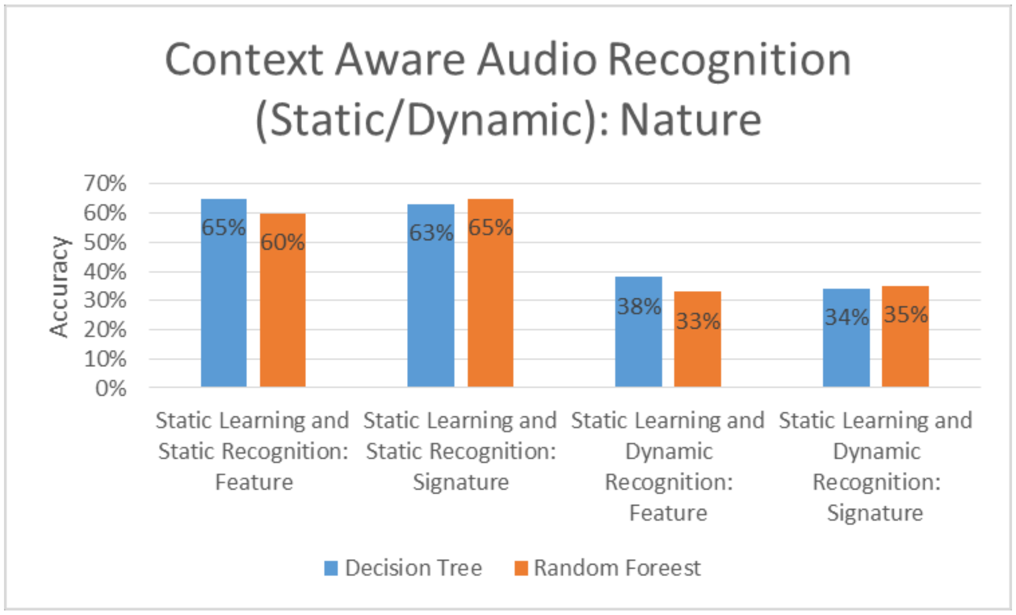


Figure 55: Context Aware Audio Recognition(Static/Dynamic): Nature -accuracy

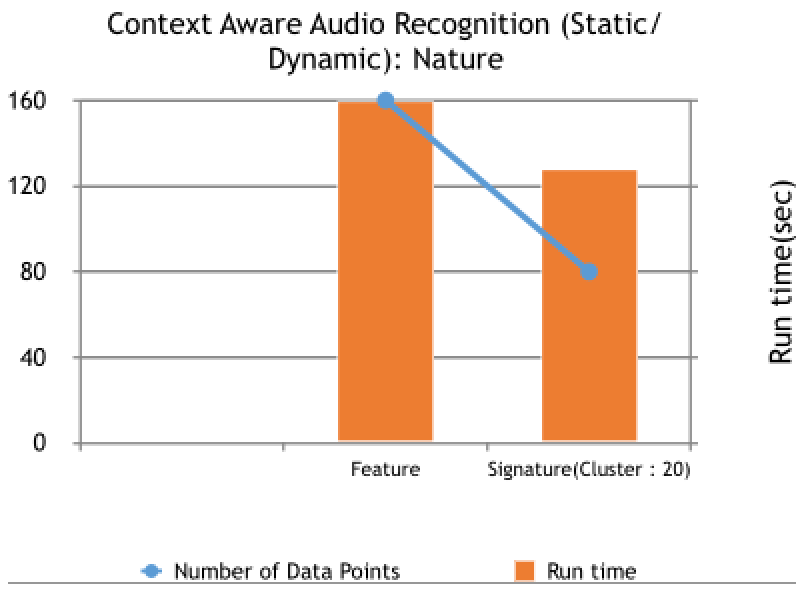


Figure 56: Context Aware Audio Recognition (Static/Dynamic): Nature - Runtime

Context Aware Audio Classification: Traffic

From figure 57, we can observer that static learning and static recognition with all features has an accuracy as 75% for decision tree and 79% for random forest. Static learning and static recognition with Signature (Cluster: 20) has an accuracy as 68% for decision tree and 67% or random forest. But Static leaning and Dynamic recognition has very low accuracy as 32% without Signature for decision tree and 28% for random forest, with signature it has even low accuracy as 29% for decision tree and 26% for random forest.

Figure 58, shows build time of models and data point size. Signature (Cluster: 20) has a build time of 120 sec and data point size of 80. Whereas feature without signature had a build time of 160 sec and data point size of 160. Which are way more than our proposed approach. Even our prosed approach has great accuracy than this approach.

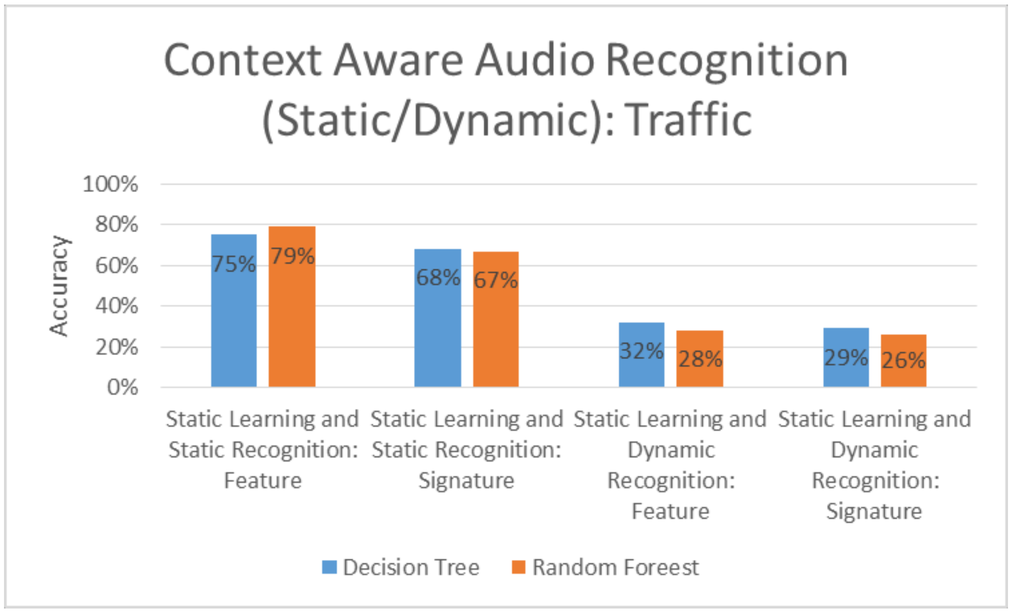


Figure57: Context Aware Audio Recognition(Static/Dynamic): Traffic -Accuracy

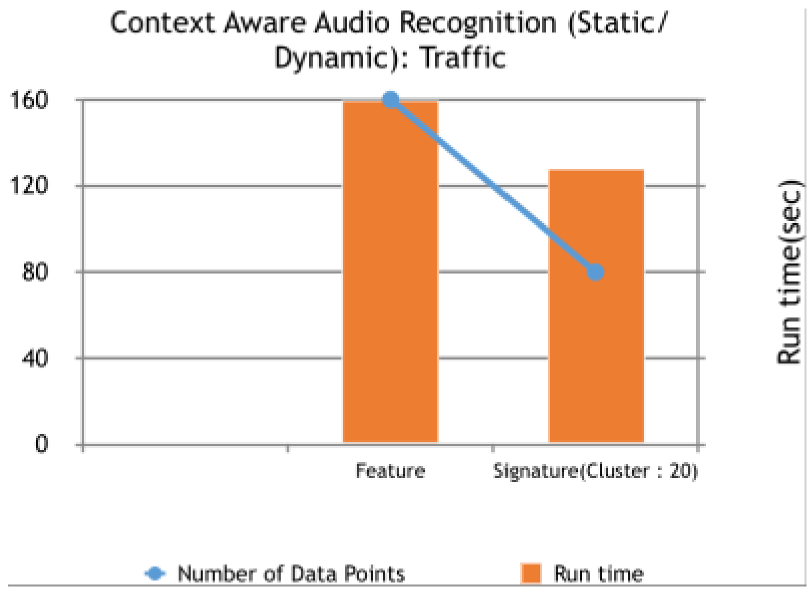


Figure58: Context Aware Audio Recognition(Static/Dynamic): Traffic - Runtime

5.8 Without Context Aware Audio Classification

In this without Context aware audio classification, we build models with all the context together.

5.8.1 Audio Classification w/o Context Aware: Dynamic Learning/ Dynamic Recognition

In audio classification without context aware approach, we have 14 audio classes namely Silent data, DoorBell, Dog Bark, Vacuum, Fire Alarm, Ocean Wave, Wind, Rain, Thunder, Police Siren, Ambulance, Bike Exhaust, Car Horn, Baby Cry. These sounds are collected at client side, so these are considered dynamic sounds. Following table shows the number of data points used, Model building time and number of classed for both server side and client side feature extraction process.

From figure 59, feature extraction on server side has an accuracy of 31% and 29% with signature for decision tree. Feature extraction on client side has an accuracy of 64% and 59% with signature for decision tree. But the build time for Client side feature Extraction took 180 sec compared to Server side feature extraction which took 820 sec build time.

	Number of Classes	Number of Data Points	Model Building Time
Feature	14	820	300 sec
Signature (cluster#:20)	14	410	180 sec

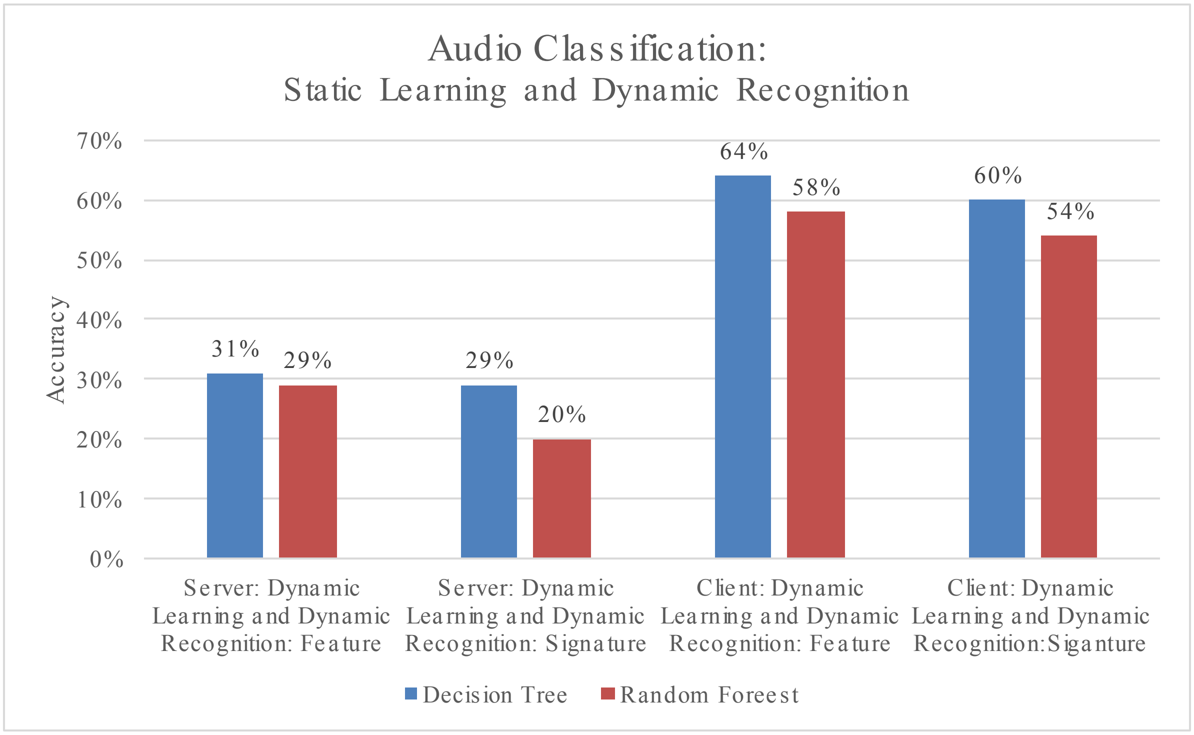


Figure 59: Audio Classification w/o context aware: Dynamic Learning/ Dynamic Recognition

5.8.2 Audio Classification w/o context aware: Static Learning/ Dynamic Recognition

In audio classification without context aware approach we have 14 audio classes namely Silent data, Door Bell, Dog Bark, Vacuum, Fire Alarm, Ocean Wave, Wind, Rain, Thunder, Police Siren, Ambulance, Bike Exhaust, Car Horn, Baby Cry. Which are static sounds, sounds collected from open source. Following table shows the number of data points used, Model building time and number of classed for both server side and client side feature extraction process. From figure 59, feature extraction on server side has an accuracy of 69% and 62% with signature for decision tree. Feature extraction on client side has an accuracy of 58% and 61% with signature for decision tree. But the build time for Client side feature Extraction took 100 sec compared to Server side feature extraction which took 600 sec build time.

		#Class	#Data Points	Model Building Time
Server Side Feature Extraction	Feature	14	15650	600 sec
	Signature (clusters: 650)	14	6921	400 sec
Client Side Feature Extraction	Feature	14	15650	100 sec
	Signature (clusters: 650)	14	6921	90 sec

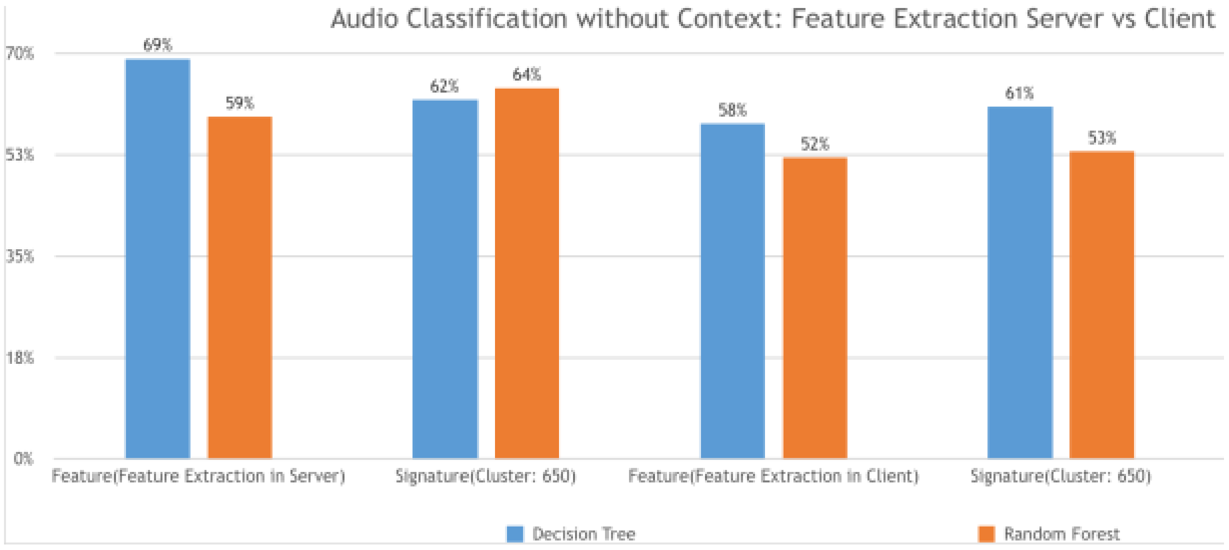


Figure 60: Audio Classification w/o context aware: Static Learning/ Dynamic Recognition

5.9 Context Base Audio Classification Results

Context's used in iHear namely are 'Households', 'Emergency', 'Nature', 'Traffic', 'Animal', 'Human'.

We have generated models and compared accuracy with and without Signature Base approach.

Context: Animals

Classes: [Dog, Cat, Crow, Tiger]

5.9.1 Without Signature Based Approach

A: Decision tree

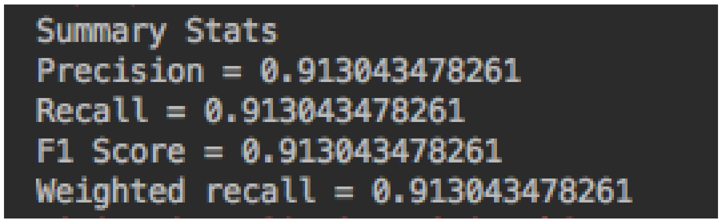
Dense matrix:

([[9., 0., 1.],

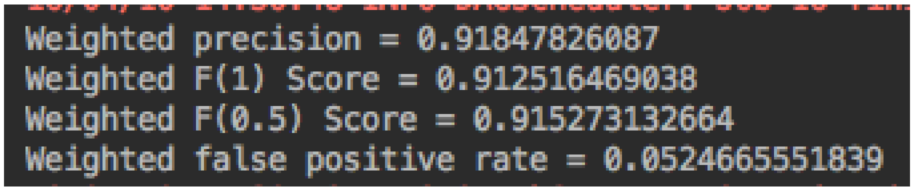
[1., 5., 0.],

[0., 0., 7.]])

Results:



```
Summary Stats
Precision = 0.913043478261
Recall = 0.913043478261
F1 Score = 0.913043478261
Weighted recall = 0.913043478261
```



```
Weighted precision = 0.91847826087
Weighted F(1) Score = 0.912516469038
Weighted F(0.5) Score = 0.915273132664
Weighted false positive rate = 0.0524665551839
```

Model:

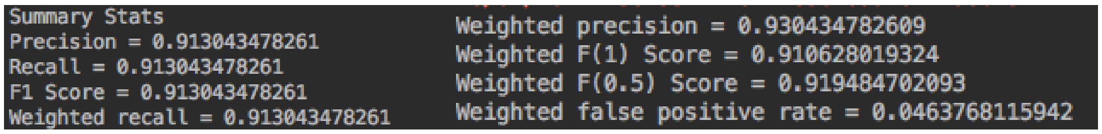
```
DecisionTreeModel classifier of depth 5 with 27 nodes

If (feature 13 <= 0.3095735766)
 If (feature 15 <= 0.519294869879)
  If (feature 6 <= 0.0123245933296)
   If (feature 2 <= 0.164728472586)
    If (feature 12 <= -21.166776586)
     Predict: 1.0
    Else (feature 12 > -21.166776586)
     Predict: 0.0
   Else (feature 2 > 0.164728472586)
    If (feature 1 <= 0.0657805609184)
     Predict: 2.0
    Else (feature 1 > 0.0657805609184)
     Predict: 0.0
  Else (feature 6 > 0.0123245933296)
   Predict: 0.0
 Else (feature 15 > 0.519294869879)
  If (feature 1 <= 0.11265557791)
   If (feature 8 <= 0.0234175696078)
    Predict: 1.0
   Else (feature 8 > 0.0234175696078)
    If (feature 0 <= 0.111075834543)
     Predict: 1.0
    Else (feature 0 > 0.111075834543)
     Predict: 0.0
  Else (feature 1 > 0.11265557791)
   If (feature 0 <= 0.325717793085)
    Predict: 1.0
   Else (feature 0 > 0.325717793085)
    Predict: 0.0
Else (feature 13 > 0.3095735766)
 If (feature 14 <= -0.880529673183)
  If (feature 0 <= 0.206005079826)
   Predict: 2.0
  Else (feature 0 > 0.206005079826)
   Predict: 0.0
 Else (feature 14 > -0.880529673183)
  If (feature 8 <= 0.0313854480779)
   Predict: 0.0
  Else (feature 8 > 0.0313854480779)
   Predict: 1.0
```

B. Random Forest:

Dense matrix:

([[8., 0., 0.], [2., 5., 0.],[0., 0., 8.]])

Results:



```
Summary Stats                                   Weighted precision = 0.930434782609
Precision = 0.913043478261                      Weighted F(1) Score = 0.910628019324
Recall = 0.913043478261                         Weighted F(0.5) Score = 0.919484702093
F1 Score = 0.913043478261                       Weighted false positive rate = 0.0463768115942
Weighted recall = 0.913043478261
```

Model:

5.9.2 With Signature Based Approach

A: Decision tree

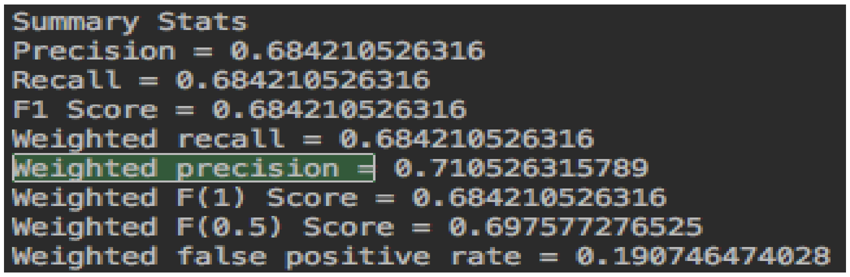
Dense matrix:

([[5., 0., 2.],

[0., 2., 0.],

[2., 2., 6.]])

Results:



```
Summary Stats
Precision = 0.684210526316
Recall = 0.684210526316
F1 Score = 0.684210526316
Weighted recall = 0.684210526316
Weighted precision = 0.710526315789
Weighted F(1) Score = 0.684210526316
Weighted F(0.5) Score = 0.697577276525
Weighted false positive rate = 0.190746474028
```

Model:

```
TreeEnsembleModel classifier with 4 trees

        Tree 0:
         If (feature 12 <= -19.7230045699)
          If (feature 9 <= 0.157965861208)
           If (feature 7 <= 0.00273870369136)
            If (feature 5 <= 0.146407837446)
             Predict: 2.0
            Else (feature 5 > 0.146407837446)
             Predict: 0.0
           Else (feature 7 > 0.00273870369136)
            If (feature 13 <= -0.385364180297)
             Predict: 1.0
            Else (feature 13 > -0.385364180297)
             Predict: 2.0
          Else (feature 9 > 0.157965861208)
           If (feature 8 <= 0.0403033222464)
            If (feature 12 <= -24.2445666624)
             Predict: 1.0
            Else (feature 12 > -24.2445666624)
        Tree 1:
         If (feature 14 <= -1.56713498347)
          If (feature 12 <= -23.231224707)
           Predict: 1.0
          Else (feature 12 > -23.231224707)
           Predict: 2.0
         Else (feature 14 > -1.56713498347)
          If (feature 15 <= 0.519294869879)
           If (feature 12 <= -24.7961806584)
            Predict: 2.0
           Else (feature 12 > -24.7961806584)
            If (feature 11 <= 0.016203566137)
             Predict: 0.0
            Else (feature 11 > 0.016203566137)
        Tree 2:
         If (feature 13 <= 0.42642344765)
          If (feature 11 <= 0.0473753195075)
           If (feature 12 <= -19.2189816432)
            If (feature 7 <= 0.00188516350107)
             Predict: 1.0
            Else (feature 7 > 0.00188516350107)
          If (feature 3 <= 1.74099306304)
           If (feature 10 <= 0.00149249523254)
            If (feature 10 <= 5.8150243508E-4)
```

```
      Predict: 2.0
     Else (feature 10 > 5.8150243508E-4)
      Predict: 2.0
    Else (feature 10 > 0.00149249523254)
     Predict: 0.0
   Else (feature 3 > 1.74099306304)
    Predict: 0.0
Tree 3:
  If (feature 15 <= 0.519294869879)
   If (feature 13 <= 0.3095735766)
    If (feature 6 <= 0.00373707406344)
     If (feature 15 <= -0.137616282964)
      Predict: 1.0
     Else (feature 15 > -0.137616282964)
      Predict: 0.0
    Else (feature 6 > 0.00373707406344)
     Predict: 0.0
   Else (feature 13 > 0.3095735766)
    If (feature 5 <= 0.273040638607)
     If (feature 14 <= -1.11319420325)
      Predict: 2.0
     Else (feature 14 > -1.11319420325)
      Predict: 0.0
    Else (feature 5 > 0.273040638607)
     Predict: 0.0
  Else (feature 15 > 0.519294869879)
   If (feature 14 <= 0.159735785723)
    If (feature 3 <= 1.81351368042)
     Predict: 1.0
    Else (feature 3 > 1.81351368042)
     If (feature 4 <= 6.98362075678E-4)
      Predict: 1.0
     Else (feature 4 > 6.98362075678E-4)
      Predict: 0.0
   Else (feature 14 > 0.159735785723)
    Predict: 0.0
```

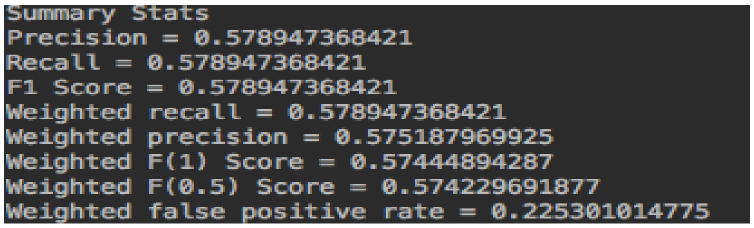
```
('Test Error = ', '0.315789473684')

Learned classification tree model:
DecisionTreeModel classifier of depth 5 with 21 nodes
  If (feature 1 <= 0.0108737286367)
   If (feature 4 <= 9.34086282949E-4)
    If (feature 7 <= 0.00232303277822)
     Predict: 2.0
    Else (feature 7 > 0.00232303277822)
     Predict: 1.0
   Else (feature 4 > 9.34086282949E-4)
    Predict: 2.0
  Else (feature 1 > 0.0108737286367)
   If (feature 15 <= 0.993272408765)
    If (feature 9 <= 0.0601045446553)
     If (feature 7 <= 0.0038569066591)
      Predict: 0.0
     Else (feature 7 > 0.0038569066591)
      If (feature 7 <= 0.0169248880105)
       Predict: 2.0
      Else (feature 7 > 0.0169248880105)
       Predict: 0.0
    Else (feature 9 > 0.0601045446553)
     If (feature 11 <= 0.0263763035622)
      If (feature 0 <= 0.152966558792)
       Predict: 0.0
      Else (feature 0 > 0.152966558792)
       Predict: 0.0
     Else (feature 11 > 0.0263763035622)
      If (feature 2 <= 0.205317816878)
       Predict: 2.0
      Else (feature 2 > 0.205317816878)
       Predict: 0.0
   Else (feature 15 > 0.993272408765)
    Predict: 2.0
```

B. Random Forest:

Dense matrix:

```
([[ 4.,  0.,  2.],

[ 2.,  2.,  1.],

[ 1.,  2.,  5.]])
```

Results:



```
Summary Stats
Precision = 0.578947368421
Recall = 0.578947368421
F1 Score = 0.578947368421
Weighted recall = 0.578947368421
Weighted precision = 0.575187969925
Weighted F(1) Score = 0.57444894287
Weighted F(0.5) Score = 0.574229691877
Weighted false positive rate = 0.225301014775
```

Model:

```
('Test Error', '0.421052631579')

Learned classification forest model:
TreeEnsembleModel classifier with 4 trees
  Tree 0:
    If (feature 14 <= -1.76991222867)
     If (feature 1 <= 0.172839857773)
      If (feature 7 <= 0.0122359328798)
       Predict: 0.0
      Else (feature 7 > 0.0122359328798)
       If (feature 10 <= 1.70333470364E-4)
        Predict: 0.0
       Else (feature 10 > 1.70333470364E-4)
        Predict: 2.0
     Else (feature 1 > 0.172839857773)
      Predict: 2.0
    Else (feature 14 > -1.76991222867)
     If (feature 4 <= 9.34086282949E-4)
      If (feature 8 <= 0.00271168056689)
       Predict: 2.0
      Else (feature 8 > 0.00271168056689)
       If (feature 1 <= 0.0108737286367)
        Predict: 1.0
       Else (feature 1 > 0.0108737286367)
        Predict: 1.0
     Else (feature 4 > 9.34086282949E-4)
      If (feature 12 <= -24.3853562605)
       If (feature 10 <= 0.)
        Predict: 0.0
       Else (feature 10 > 0.00116921918265)
        Predict: 2.0
```

```
Tree 1:
 If (feature 6 <= 0.0512517391794)
  If (feature 14 <= -1.52951474303)
   If (feature 9 <= 0.232447913979)
    If (feature 8 <= 0.00308263664871)
     Predict: 2.0
    Else (feature 8 > 0.00308263664871)
     Predict: 0.0
   Else (feature 9 > 0.232447913979)
    If (feature 5 <= 0.103714285714)
     Predict: 0.0
    Else (feature 5 > 0.103714285714)
     Predict: 1.0
  Else (feature 14 > -1.52951474303)
   If (feature 7 <= 0.00435695823762)
    If (feature 9 <= 0.387369536095)
     Predict: 1.0
    Else (feature 9 > 0.387369536095)
     Predict: 2.0
   Else (feature 7 > 0.00435695823762)
    If (feature 4 <= 0.00170840635891)
     Predict: 2.0
    Else (feature 4 > 0.00170840635891)
     Predict: 2.0
 Else (feature 6 > 0.0512517391794)
  Predict: 1.0
Tree 2:
 If (feature 1 <= 0.0108737286367)
  Predict: 1.0
 Else (feature 1 > 0.0108737286367)
  If (feature 15 <= 0.778421199568)
   If (feature 11 <= 0.00128463024832)
    Predict: 2.0
   Else (feature 11 > 0.00128463024832)
    If (feature 15 <= -1.11486907692)
     Predict: 0.0
    Else (feature 15 > -1.11486907692)
     Predict: 0.0
  Else (feature 15 > 0.778421199568)
   If (feature 13 <= -3.81910360002)
    Predict: 0.0
   Else (feature 13 > -3.81910360002)
    Predict: 2.0
Tree 3:
 If (feature 1 <= 0.0108737286367)
```

```
    If (feature 15 <= 0.460278224262)
     If (feature 8 <= 0.00271168056689)
      Predict: 2.0
     Else (feature 8 > 0.00271168056689)
      Predict: 1.0
    Else (feature 15 > 0.460278224262)
     Predict: 2.0
   Else (feature 1 > 0.0108737286367)
    If (feature 14 <= -1.76426319502)
     If (feature 0 <= 0.0805515239478)
      If (feature 11 <= 0.0132908883877)
       Predict: 2.0
      Else (feature 11 > 0.0132908883877)
       Predict: 0.0
     Else (feature 0 > 0.0805515239478)
      If (feature 15 <= 0.993272408765)
       Predict: 0.0
      Else (feature 15 > 0.993272408765)
       Predict: 2.0
    Else (feature 14 > -1.76426319502)
     If (feature 15 <= -0.617488118807)
      If (feature 13 <= -0.923105494181)
       Predict: 2.0
      Else (feature 13 > -0.923105494181)
       Predict: 0.0
     Else (feature 15 > -0.617488118807)
      If (feature 8 <= 0.011216017488)
       Predict: 1.0
      Else (feature 8 > 0.011216017488)
       Predict: 2.0
```

5.10 iHear Application:

5.10.1 Implementation of iHear Server using Storm and Spark

We are performing all the classification tasks in Spark using the machine-learning library. Twitter stream is being directed from Storm to Spark. The data from the Spark is sent to Robot using socket programming.

5.10.2 Implementation of User Mobile App

iHear mobile client provides certain functionalities to the user like enhancing sounds, applying filters for better hearing, increasing sound on the left or right channel, choosing the number of channels etc. As a user selects functionality and provides required audio data or sends ongoing sound dynamically, iHear then stores the sound data in Mongodb. Later Mongodb send's the sound data to Spark for processing the data, Spark sends the output to Mongodb. Finally iHear displays the output from Mongodb.

5.10.3 Implementation of Robo

The Robot has Dialogue Manager, which is used to handle the user interaction with the Robot. iHear Robot catches the ongoing sound and sends it directly to Spark for processing the sound data, Spark processes the data and depending on classification models developed it gets the output and sends it to the robot. iHear robot on getting the result from spark, it notifies the user in which environment the user is in.

5.10.4 iHear User App

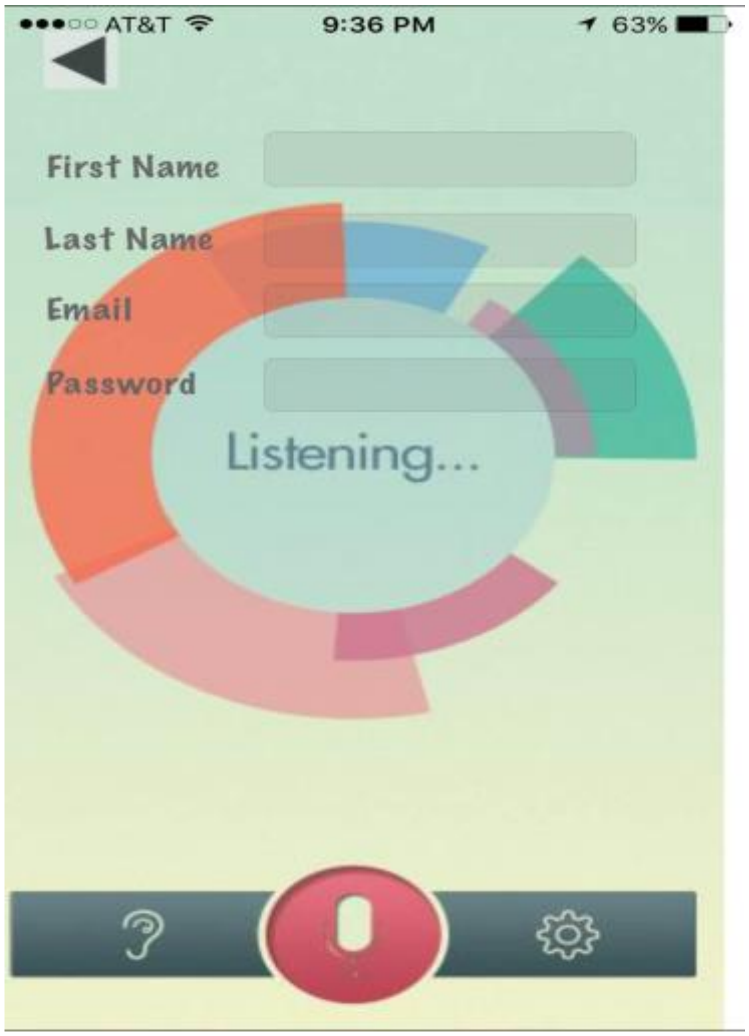


Figure 61: Registration Page

Before using the iHear, User has to first sign up providing the first name, last name, email and password, this data will be stored in Mongolab for the time being.

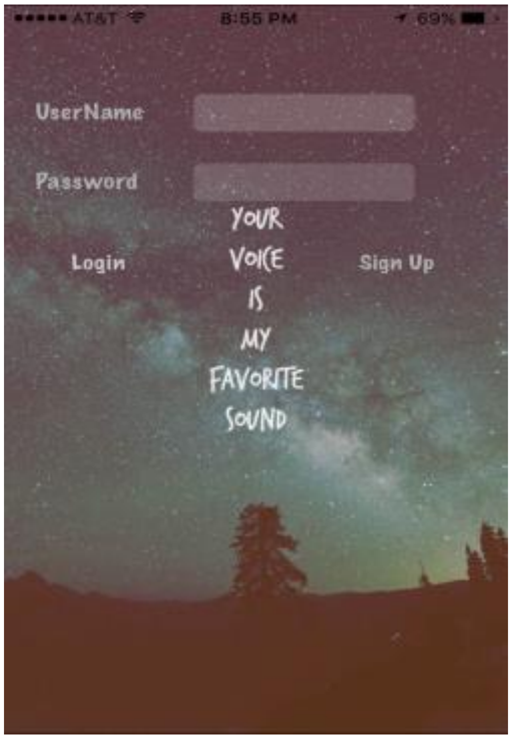


Figure 62: User Login Page

After Signing up. User can login through iHear by providing user name and password, iHear verifies the information from mongo lab and allows the user for the next step in Voice enhancement.

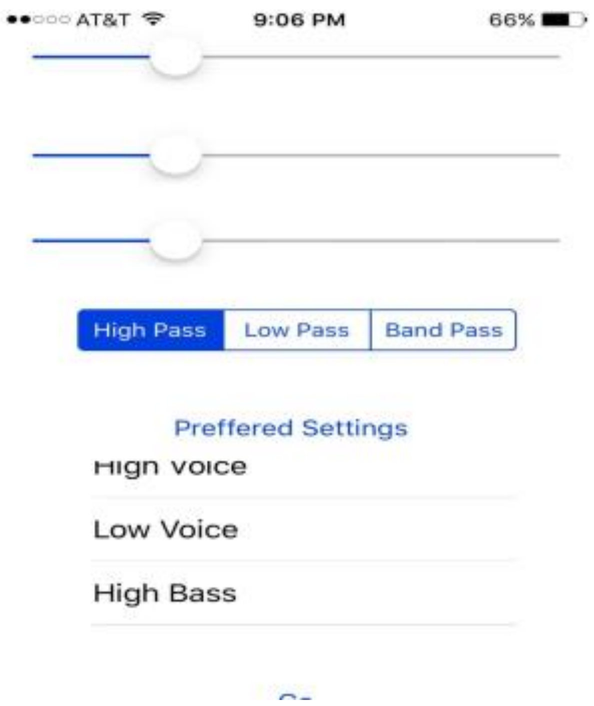


Figure 63: Home Screen

The User can select any filter with adjustable frequency for better result in filtered audio. User can even choose the preferred setting for his interest like High pass, Low pass, High voice, low voice.

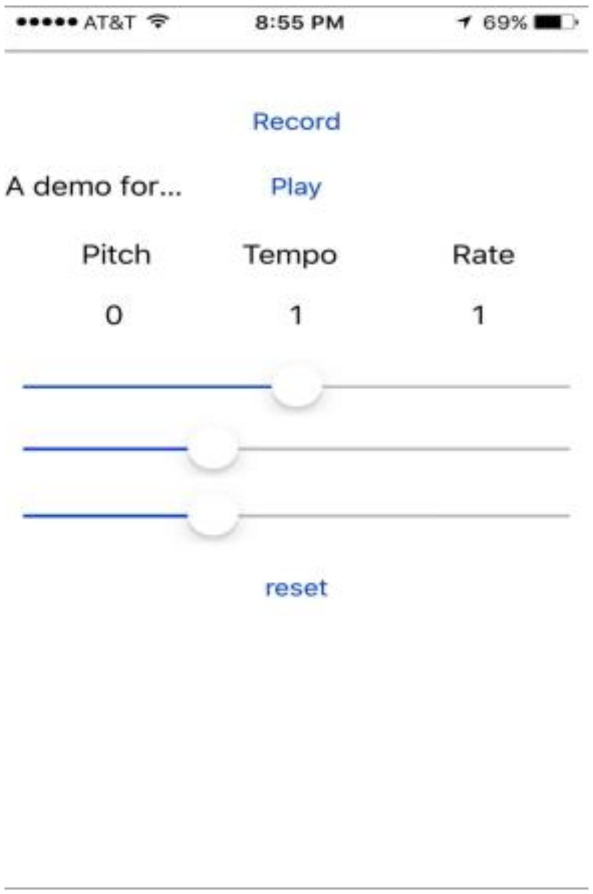


Figure 64: Voice Changer

On clicking Voice changer, users will be navigated to this page, where the user can adjust the Pitch value, Tempo Value and Rate value to get the desired sound user wants to hear back from iHear. iHear can manipulate the voices into female voice, funny voice like, Voice with bass sound etc. The more user plays by changing pitch, tempo and rate values the more results user gets from it.

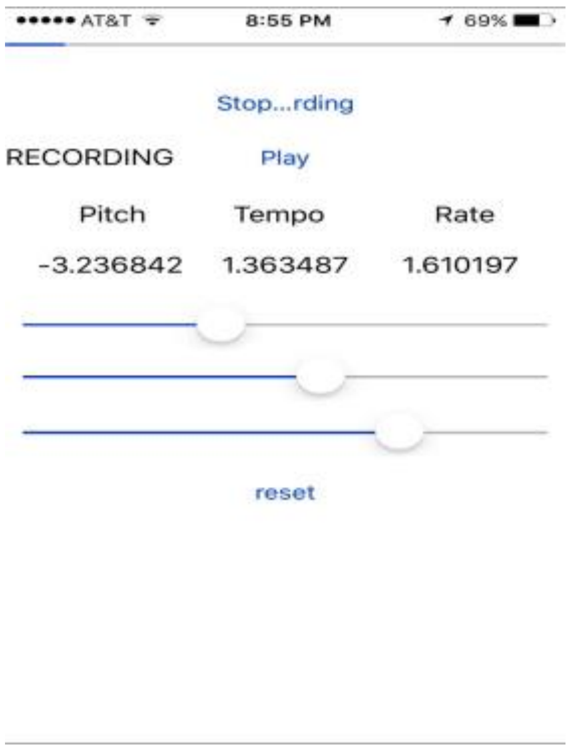
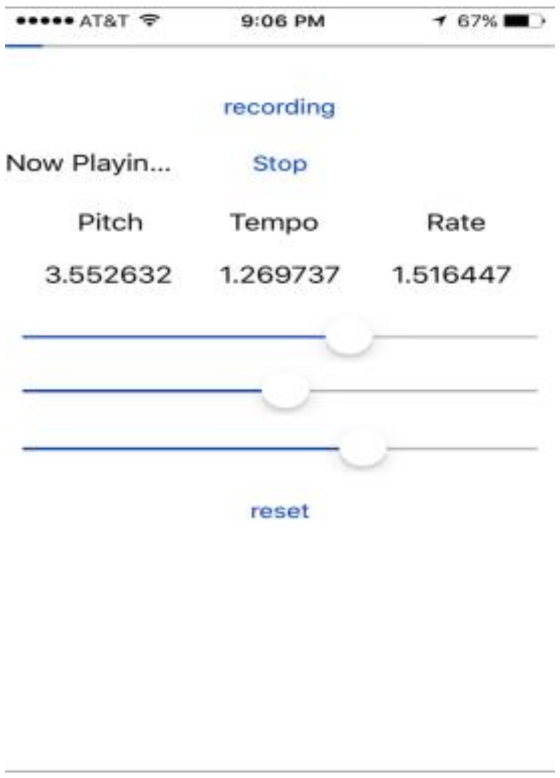


Figure 65: Voice Changer (Recording)

The User has to adjust the pitch, tempo and rate slider and its respective values will be shown on the screen. On clicking Record button the user voice will be recorded by iHear. This recorded sound is stored in a file.

iHear has Sound detector on top of the app to notify the user whether iHear is listening to what user said or not. If it remains unchanged, then iHear is not listening. If sound detector detects sound it changes as per the peak values of sound.



Pitch	Tempo	Rate
3.552632	1.269737	1.516447

Figure 66: Voice Changer (Playing).

Example:

Female Voice:

By providing the following value female voice can be obtained.

Pitch Value: 3. 15

Tempo Value: 1.08

Rate Value: 0.94

Male Base Voice:

By providing the following value Male Base voice can be obtained.

Pitch Value: -5.28

Tempo Value: 1.08

Rate Value: 2.447

Chipmunk Voice:

By providing the following value Chipmunk voice can be obtained.

Pitch Value: 7.55

Tempo Value: 1.55

Rate Value: 1.447

5.10.5 iHear Robo

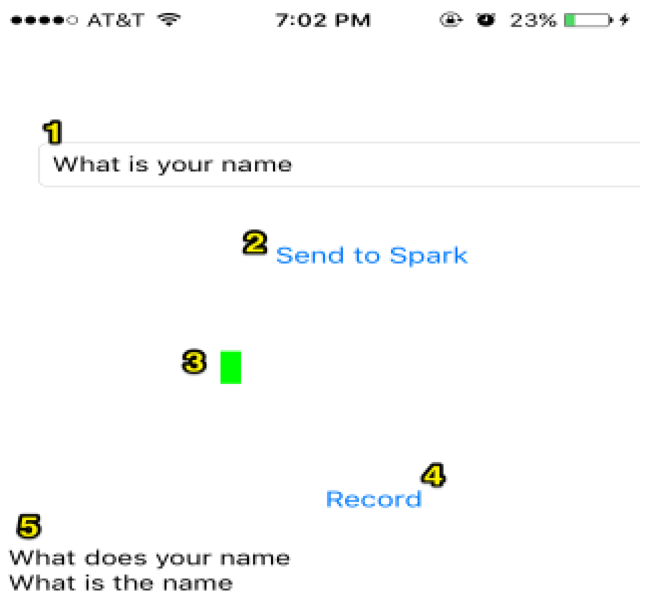


Figure 67: Robo Home Screen

Description of the Figure: Home Screen

1. User's dialogue shall be shown on this label.

2. This button is used to send the above text to Spark.

3. This shows the progress of user’s audio frequency.
4. Used to record the user's speech and convert to text. The modified Pitch, Tempo and Rate values are applied to the recorded sound and played back to the user when the Play button is clicked.
5. These are possible speech to text from the user's speech.

CHAPTER 6

CONCLUSION AND FUTURE WORK

iHear achieves Machine Learning on Client side with collaboration between server and cloud environment with very less computational power, processing time and good accuracy. iHear Engine with the context aware model has better accuracy and reduced build time on the server.

iHear IOS application can distinguish different sounds based on user location and time. It also gives the flexibility to the user to apply filters on dynamic real time data coming from receiver 'mic'.

Considering a problem we face in day-to-day life. It often occurs as a difficulty for people with hearing and communicating with the opposite person when in public places like restaurants, concerts, shows and railway stations due to the unavoidable surrounding noises. Using this app shall favor them in reducing the unwanted/unnecessary background noises and helps them concentrate only on the sounds they want to hear. They can easily reduce the background noise by applying filters and clearly hear what the opposite person is saying or has to say. This is another authentic way of using this app.

Another best place this app can find as a use is when people would like to study or work on the go. Most of the people make time in the travel to manage their office/personal work. It can be some work they are supposed to finish on their laptop or a journal to read or a conference call to attend. Turning on this app will automatically reduce the background noises and favors them in providing a better surrounding to work just like the beats headphones helps in cancelling the background noise and enhancing the music we want to hear.

7.1 Future Work

The proposed model can be extended to multiple platforms like android, windows. In depth analysis may be carried out in terms of new filters. iHear application can be made as third party application which acts as intermediate to receive calls to make it better audible. Increase the Class size by collecting more data.

REFERENCES

1. Chen, Yao, et al. "An Intelligent Hearing Aid System Based on Real-Time Signal Processing." *Cyber-Physical Systems, Networks, and Applications (CPSNA), 2014 IEEE International Conference on*. IEEE, 2014.

2. Mills, Mara. "Hearing aids and the history of electronics miniaturization." *IEEE Annals of the History of Computing* 33.2 (2011): 24-45.

3. Chen, Yao, et al. "An Intelligent Hearing Aid System Based on Real-Time Signal Processing." *Cyber-Physical Systems, Networks, and Applications (CPSNA), 2014 IEEE International Conference on*. IEEE, 2014.

4. "Millions have hearing loss that can be improved or prevented". http://www.who.int/mediacentre/news/notes/2013/hearing_loss_20130227/en/ [Accessible on 6/20/2016]

5. Jarng S S, Swanson C, Lee F, et al. Noise Reduction Algorithm applied for Hearing Aids[J]. Vol. 25, 2013, *onlinepresent.org/proceedings/vol25_2013/62.pdf* [Accessible on 6/20/2016]

6. Büchler, Michael, et al. "Sound classification in hearing aids inspired by auditory scene analysis." *EURASIP Journal on Applied Signal Processing* 2005 (2005): 2991-3002.

7. Banchhor, Sumit Kumar, and Arif Khan. "Musical Instrument Recognition using Spectrogram and Autocorrelation‖." *International Journal of Soft Computing and Engineering (IJSCE)* 2.1 (2012): 1-4.

8. "Audio Signal Processing for Music Applications":Overview, Retrieved March 6, 2016, from Wikipedia:

   https://www.coursera.org/course/audio

9. "Statistical Classification": Overview, Retrieved March 6, 2016, from Wikipedia:

https://en.wikipedia.org/wiki/Statistical_classification

10. Audio Signal: Overview, Retrieved March 6, 2016, from Wikipedia:

    https://en.wikipedia.org/wiki/Audio_signal

11. Nordqvist, Peter, and Arne Leijon. "An efficient robust sound classification algorithm for hearing aids." *The Journal of the Acoustical Society of America* 115.6 (2004): 3033-3041.

    Büchler, Michael Christoph. *Algorithms for sound classification in hearing instruments*. Diss. SWISS.

12. Python Audio Analysis Package: Overview, Retrieved March 6, 2016, from Github:

    https://github.com/tyiannak/pyAudioAnalysis

13. Feature selection: Overview, Retrieved March 6, 2016, from Wikipedia

    https://en.wikipedia.org/wiki/Feature_selection

14. Audio Kit Examples: Overview, Retrieved March 6, 2016, from Audiokit

    http://audiokit.io/examples/

15. Auido Kit Open Source Project: Overview, Retrieved March 6, 2016, from Github

    https://github.com/audiokit/AudioKit

16. Awesome iOS Open Source Project: Overview, Retrieved March 6, 2016, from Github

    https://github.com/vsouza/awesome-ios

17. Open Source Platform for Audio Synthesis Processing and Analysis: Overview, Retrieved March 10,2016-06-24, form synthtopia.

http://www.synthtopia.com/content/2015/01/12/audiokit-an-open-source-platform-for-audio-synthesis-processing-and-analysis/

VITA

Guru Teja Mannava completed his Bachelor’s degree in Computer Science from Jawaharlal Nehru Technological University in Hyderabad and then worked as a Programmer Analyst in Cognizant Technological Solutions for 2 years. Mr. Guru Teja Mannava started his masters in Computer Science at the University of Missouri-Kansas City (UMKC) in January 2015, specializing in Data Sciences and Software Engineering. Upon completion of his requirements for the Master’s Program, Mr. Guru plans to work as a Data Scientist.