5-2018

Standard Machine Learning Techniques in Audio Beehive Monitoring: Classification of Audio Samples with Logistic Regression, K-Nearest Neighbor, Random Forest and Support Vector Machine

Prakhar Amlathe
Utah State University

STANDARD MACHINE LEARNING TECHNIQUES IN AUDIO BEEHIVE MONITORING: CLASSIFICATION OF AUDIO SAMPLES WITH LOGISTIC REGRESSION, K-NEAREST NEIGHBOR, RANDOM FOREST AND SUPPORT VECTOR MACHINE

by

Prakhar Amlathe

A thesis submitted in partial fulfillment
of the requirements for the degree

of

MASTER OF SCIENCE

in

Computer Science

Approved:

Vladimir Kulyukin, Ph.D. Major Professor	Nicholas Flann, Ph.D. Committee Member
Haitao Wang, Ph.D. Committee Member	Mark R. McLellan, Ph.D. Vice President for Research and Dean of the School of Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2018

ABSTRACT

Standard Machine Learning Techniques in Audio Beehive Monitoring: Classification of Audio Samples with Logistic Regression, K-Nearest Neighbor, Random Forest and Support Vector Machine

by

Prakhar Amlathe, Master of Science

Utah State University, 2018

Major Professor: Vladimir Kulyukin, Ph.D.
Department: Computer Science

This thesis inaugurates and takes the first step towards the use of audio data in the Electronic Beehive Monitoring System (BeePi) with an aim to classify audio samples with standard machine learning techniques. It covers two objectives, firstly for three-way classification problem, it classifies a 2-second audio sample into one of three class categories - bee buzzing, ambient noises or cricket chirping and secondly for time of the day classification problem, an audio sample is categorized into one of the six-time periods of the day class 0-5, class 5-9, class 9-13, class 13-17, class 17-20 and class 21-00. Four machine learning algorithms were used to train the classification models. The results indicate that models performed on par and can be used in audio beehive monitoring.

(56 pages)

PUBLIC ABSTRACT

Standard Machine Learning Techniques in Audio Beehive Monitoring: Classification of Audio Samples with Logistic Regression, K-Nearest Neighbor, Random Forest and Support Vector Machine

Prakhar Amlathe

Honeybees are one of the most important pollinating species in agriculture. Every three out of four crops have honeybee as their sole pollinator. Since 2006 there has been a drastic decrease in the bee population which is attributed to Colony Collapse Disorder (CCD). The bee colonies fail/ die without giving any traditional health symptoms which otherwise could help in alerting the Beekeepers in advance about their situation.

Electronic Beehive Monitoring System has various sensors embedded in it to extract video, audio and temperature data that could provide critical information on colony behavior and health without invasive beehive inspections. Previously, significant patterns and information have been extracted by processing the video/image data, but no work has been done using audio data. This research inaugurates and takes the first step towards the use of audio data in the Electronic Beehive Monitoring System (BeePi) by enabling a path towards the automatic classification of audio samples in different classes and categories within it. The experimental results give an initial support to the claim that monitoring of bee buzzing signals from the hive is feasible, it can be a good indicator to estimate hive health and can help to differentiate normal behavior against any deviation for honeybees.

ACKNOWLEDGMENTS

I take this opportunity to express my sincere appreciation and gratitude to my advisor, Dr. Vladimir Kulyukin, for his consistent and valuable academic guidance that led to the success of this project. His experience and knowledge helped me to overcome numerous challenges. He has been an inspirational figure and I would always be grateful for some of the treasured life lessons I have learned from him during this journey.

I acknowledge my gratitude to my committee members, Dr. Nicholas Flann and Dr. Haitao Wang for their continuous support, feedback and helping me understand difficult concepts.

Finally, I would like to thank my beloved family for their unconditional love and support. I thank my friends for cheering me up and helping me to smile during hard times which made this journey much memorable.

Prakhar Amlathe

CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Honeybees are a flying insect which makes up 80 percent of the world's pollinating insects. The best known honeybee is the Western honeybee which has been domesticated for honey production and crop pollination. These are also known as Apis millifera. Honeybees represent only a small fraction of the roughly 20,000 known species of bees [1]. They are the critical pollinator and pollinate 70 of the around 100 crop species that feed around 90 percent of the world. A world without bees could struggle to sustain the food needs of global world population [2]. Whole Foods recently imagined what a grocery store would be like in a world without bees by removing more than half of the market's produce [3]. The Western honeybee is responsible for one out of every three daily mouthfuls that the average U.S. resident eats. Their overall health can be considered as an indicator of health of our eco-system as they are like the glue which holds our ecosystem intact. Since 2006 honeybees have been disappearing from amateur and commercial apiaries. This trend has been called the colony collapse disorder (CCD) [4].

Colony collapse disorder occurs when majority of worker bees in a colony disappear and leave behind a queen, plenty of food and a few nurse bees to care for the remaining immature bees and the queen. In a period of six years from 2006 to 2013, more than 10 million beehives were lost, often to CCD, nearly twice the normal rate of loss. Several possible causes for CCD have been proposed, but no single has gain widespread acceptance among the scientific community [5]. Since, there is no such single factor responsible for CCD, regular hive inspections and timely precautionary measures seem to be a way to address this issue. Survey shown in the Fig 1.1, shows the colony losses in the U.S against the beekeepers' acceptable values [6].

In general, beekeepers monitor the honeybee colony by inspecting the beehive for pollen, nectar, brood quantities and for the presence of any disease or pests. Regular checks are

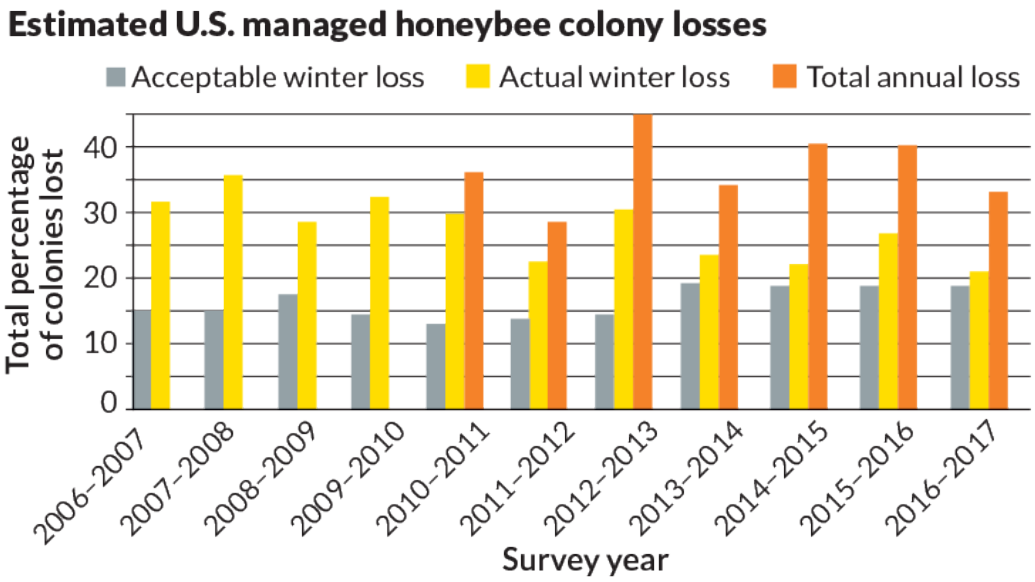

Fig. 1.1: Honeybee colony loss survey

critical for finding and prevention of pests and diseases such as viruses, mites, and bacterial infection [7]. However, monitoring beehives manually to obtain the information about honeybee's health, activity and behavior is a time-consuming process for beekeepers. Most of the hives are generally located on the outskirts, so every inspection also incurs a transportation cost to beekeepers. Also, manual hive inspections often disrupt the life cycles of bee colonies. Thus, electronic beehive monitoring (EBM) makes it possible to capture large amounts of useful information about the behavior of the bees without interrupting the life cycles of honeybee colonies and increasing physical demands on beekeepers. Understanding the benefits of such approach, a consensus is rising among analysts and professionals that EBM can help to collect critical information on the colony behavior.

Every action of the bees in the bee colonies produces data such as temperature inside the beehive, buzzing audio frequencies, weight changes and pictures and videos of forager traffic. Each kind of data reveals certain critical information about the hive health. Analyzing all this data together continuously over a period can help to come up with patterns that can eventually help us to differentiate what is normal behavior and what is the deviation for honeybees. This will help the beekeepers by altering then in advance about the state of the

hive so that they can precautionary measures to restore the health the hive.

In this study, the focus is on analyzing the buzzing sound generated by the bees. Bees buzz for two reasons. Firstly, the rapid wing beats of bees create vibrations that can be heard as buzzes to us. Secondly, when the bees visit a flower, they create buzzing vibrations by using their wing muscles and thorax which causes the pollen to shake off from the flower and attach to the bee body which then transfers pollen to other flowers leading to pollination [8]. The loudness of the buzzing sound of the bees changes with the time of the day. Also, in case of the Varroa mites and other viruses attack, the buzzing pattern changes thus constant monitoring of the bee buzzing can provide useful information and can be a good estimator for the health of the hive.

The remainder of this thesis is organized as follows. In Chapter 2, related work is presented. In Chapter 3, a procedure for the audio classification of the bee buzzing signals is presented. In Chapter 4, the experiments conducted on the audio samples are discussed. In Chapter 5, the results of the experiments are presented. In Chapter 6, the conclusion of the whole study is presented and the possible enhancements in the form of future work are discussed.

CHAPTER 2

RELATED WORK

There has been growing interest and emerging consensus among the beekeepers and the research community that Electronic Beehive Monitoring is the way forward as it helps to gather critical data from the hive without invasive hive inspections which generally affects the bee colonies cycle [9]. Bees are affected by sensory changes such as pollution and Wi-Fi connection in the vicinity. Capturing such changes can help us to understand their behavior and Colony Collapse Disorder but currently, the issue that research community faces is that they do not have enough data to analyze it. As mentioned in the report published by United States Department of Agriculture (USDA): "*Another issue complicating the research is that, so far, researchers only have samples taken after a CCD incident is reported. With just the one set of samples, especially since the adult bees have disappeared, researchers cannot look for specific changes in affected bee colonies preceding the collapse*". Since beekeepers cannot afford regular inspections due to obvious reasons of logistics and fatigue, its difficult to capture crucial colony information at the right time. Thus, Electronic Beehive Monitoring can help to capture all such changes. The Second International Workshop on Hive and Bee Monitoring predicted that market for EBM is going to be vast in next few years [10].

For an open source Electronic Beehive System project, key aspect should be that they should be made up of off-the-shelves components so that beekeepers can replicate the hardware design on their own. BeePi takes a step in this direction . There are many commercial systems such as (e.g., Arnia [11] and HiveMind [12]) that offer hive monitoring through audio and video sensors. Arnia provides hive data acquisition in individual hives and access to the data through Internet-enabled devices.

In [13] Mezquida and Martinez presented a platform to capture colony buzz using sensors and sending data to a common database for further analysis and sound processing which helped them to extract plenty of patterns and tendency lines related to colony activities

and conditions. The purpose was to co-relate swarm activities with parameters registered through sensors.

In [14] Ramsey et al. utilize the fact that honeybees use vibrational communication pathways to transfer information. One such short vibrational pulse named stop signal has been focused in their study for a period of 9 months. The study concludes that vibrational pulse is generated under many different circumstances and monitoring such changes for this signal could provide a way to access the colony status of hives.

Swarming is the process by which a new honey colony is formed when the queen bee leaves the colony with a large group of worker bees [15]. As swarming leads to honey loss to beekeepers, S. Ferrari et al. [16] proposed a method to predict swarming in advance that can help to prevent the queen from leaving the hive. In their study, an acoustic method was proposed that when taken in conjunction with the temperature data captured for the hive provides some significant insights for the swarming event and can be used as predictors for the event. Results indicate that as swarming is approached, changes in the acoustic features of the sound recorded in a hive are observed such as the increase in power spectral density along with the augmentation of amplitude and frequency.

In [17], Bencsik et al. presents an approach to predict swarming by analysis of the time course of the normal vibrations recorded on the hive walls using accelerometers as a sensor in their EBM system. By logging data continuously for 8 months, they found that amplitudes of the independent signals formed a multi-dimensional time-varying vector which provides a signature highly specific to swarming process which helps to predict the phenomenon in advance.

J.Ruben et al. [18] presented an approach for identifying a place using environmental background sounds such as Rain, Casino, Playground etc. Fourteen classes of environmental background sounds were considered with 5 audio files in each category resulting to 70 audio files. Temporal, Frequency and Statistical features were extracted to create an audio fingerprint of length 62 units.The proposed method was evaluated with two classifiers and got an acceptable accuracy with both the classifiers, Random Forest and Support Vector

Machine.

In [19], Kulyukin et al. presented an audio processing algorithm for digitizing bee buzzing signals into A440 piano note sequences. The purpose for the method was, when viewed as time series, note sequences can be correlated with other timestamped data for pattern recognition.

Tzanetakis and Cook [20] proposed a method for automatic music genre classification of audio signals which had become a benchmark for all the future works in the music information retrieval field and other audio related fields. Musical genres are categorical labels that help to differentiate and characterize pieces of music. A dataset for 10 music genres was created which is widely known as GTZAN dataset. Three features set for representing timbral texture, rhythmic context and pitch content of the music were proposed and evaluated using statistical pattern recognition classifiers. An accuracy of 61 percent was achieved for classification of audio samples into different genres in non-real time. We took inspiration from this work to formulate and evaluate our three-way classification problem as different environmental sounds can be put in parallel with different musical genres for classification problem.

Salamon et al. [21] presented a taxonomy of urban sounds and a new dataset Urban Sounds which have been utilized widely in the field urban informatics and multimedia retrieval. The dataset makes up to 27 hours of audio with 18.5 hours of annotated sound events. A series of classification experiments were carried out as a form of an open ended question on this dataset to study the challenges presented by it. Experiments utilized 10-fold cross-validation approach, a variation of slicing of samples and extracting Mel-Frequency cepstrum coefficients (MFCCs) as part of feature extraction to fed to classifiers. Classification accuracy was evaluated using five different algorithms: decision trees, k-NN, random forest, support vector machine and a baseline majority vote classifier.

Qandour et al. [22] proposed a system that can remotely detect the presence of pest infestation on a honeybee colony by comparing the acoustic fingerprint of a hive with a fingerprint of a healthy hive. As a part of an acoustic [23] analysis, four features were

extracted Peak Frequency, Spectral Centroid, Bandwidth and Root Variance Frequency. These features were then fed to three classification tools such as Principal Component Analysis (PCA), Support Vector Machines (SVM) and Linear Discriminant Analysis (LDA) in conjunction and independently to develop four separate approaches to compare the results. Among the three, PCA was most suitable and was able to separate a healthy hive from an infected hive in the acoustical domain. However, the training and test samples utilized for experiments were in a small amount and were of a low quality which required multiple preprocessing.

CHAPTER 3
AUDIO CLASSIFICATION TECHNIQUES IN BEEPI

3.1 Overview

This chapter describes the techniques and procedures in detail that have been utilized to achieve the objective of audio classification in BeePi. The whole chapter has been categorized into multiple sections. Firstly, the problem statements that have been addressed are described. Later sections focus on data capturing and processing, features extracted, and machine learning models used.

3.1.1 Three-way classification problem

Three-way classification problem has been formulated with an aim to classify audio samples collected from BeePi into one of the three categories - bee buzzing, cricket chirping or ambient noise. Noise can be anything but in real environment, it is mainly constituted by microphone clicks, breeze sound and the random background noises. Automatic classification of samples in these three categories will enable analysis of sound patterns in BeePi environment. Any deviation in classification pattern over a period of few days can alarm the beekeepers of possible foreign bodies attack on the hive such as Varroa mites or wing virus. Our analysis and observation of data over a period of four months showed that bee buzzing starts from early morning and goes till the evening with quite a lot variation in their amplitude/volume throughout the day. Cricket chirping starts from the early midnight and goes till early morning hours. Crickets are absent during the months of June and July.

3.1.2 Time of day classification problem

Time of day classification problem has been formulated with an aim to classify audio samples collected from BeePi into one of the six-time categories using machine learning models [0.00 5.00), [5.00 -8.00), [9.00-12.00), [12.00-17.00), [17.00-20.00) and [21.00-0.00).

Time slots are divided by observing the bee buzzing behavior and how it changes from one-time category to other. Forager traffic is the number of bees entering or exiting the bee hive over a given period. Previously, a vision-based algorithm has been proposed to estimate the forager traffic by counting the bees from the images captured using camera sensor of BeePi [24]. Another approach for estimating forager traffic can be analyzing bee buzzing behavior over a day. This problem is a step in this direction which can help to classify a buzz sample into different time categories and can eventually lead to a rough estimate of the hive behavior for forager traffic by creating a pattern for variation of volumes of bee buzz over a day. Any deviation from it can then give us critical insights into the hive health.

3.2 BeePi: solar-powered, multi-sensor electronic beehive monitoring device

Electronic beehive monitoring system (EBM) extracts critical information on colony behavior and health without invasive hive inspections. It is a multisensory electronic beehive monitor that consists of a Raspberry Pi 3 computer, a camera, a clock, a temperature sensor and three microphones [25].Two BeePi EBMDs were assembled and deployed in Logan, UT (41.7370 *N, 111.8338 *W) and North Logan, UT (41.7694 *N, 111.8047 *W) from May to September to collect 150 GB of audio, temperature, and image data in different weather conditions. Thus, EBM makes it possible to capture large amounts of useful information about the behavior of the bees without interrupting the life cycles of honeybee colonies and increasing physical demands of beekeepers.

An essential target of BeePi design is its reproducibility: other researchers and beekeepers should be able to replicate the system and reproduce the proposed results with ease. BeePi is built to work with standard Langstroth beehives and the hardware used is from off the shelf components which makes it much cheaper as compared to other commercial systems. Thus, except for drilling narrow holes in inner hive covers for temperature sensors and microphone wires, no structural hive modifications are required for deployment [24].

Currently, BeePi is capable of estimating forager traffic which refers to a number of bees going in and out of the hive per unit time using the vision sensors. The most critical component pertaining to this research is the audio sensor which captures the 30 seconds

audio sample in every 15 minutes through a thread running as a part of software design and attaches a timestamp to it while naming the file. All the samples captured are in .wav format and are saved on 60G SD card inserted into the Raspberry Pi 3. Fig 3.1 and Fig 3.2 shows BeePi and its hardware components.

Fig. 3.1: BeePi in the field

3.3 Audio data collection through BeePi

Audio sensors in BeePi are utilized to collect audio data.The main audio sensor utilized are the microphones. The Neewer (TM) mini lapel microphone is used for the EBM system. It consists of a 3.5 mm audio jack. Small and compact design of mini lapel microphone, allows us to place it inside the second-deep super without any disturbance to the bees. There are total three microphones placed inside the deep super, each is placed on a different side

Fig. 3.2: BeePi hardware components in a langstroth super

of the hive except for the front. A six-way multiport 3.5 mm jack splitter has been used to connect three audio sensors to an audio adapter which is itself connected to the Raspberry Pi 3 via USB port. Fig 3.3 shows a Neewer (TM) microphone and a USB microphone hub with three microphones [26].

Fig. 3.3: Neewer made hands free mini lapel microphone with 3.5 mm Jack Splitter

3.4 Data extraction and preprocessing

3.4.1 Understanding data

A spectrogram is a visual way of representing the signal strength, or loudness, of a signal over time at various frequencies present in a waveform. It expresses the graph of the energy content of a signal expressed as function of frequency and time. Unlike other graphs, it splits the frequency components of a waveform and depicts the amplitude of the frequency component over time. A spectrogram is a very detailed and accurate image of our audio that gives a 3D view.This helps us to visualize the audio samples and how the parameters varies over the time for bee, noise and cricket class as shown in Fig 3.4, Fig 3.5 and Fig 3.6 respectively.

X Axis: The horizontal axis which shows the time

Y Axis: The vertical axis which shows the frequency.

Graph Region: The amplitude of the frequency components is expressed by means of the degree of color darkness. The light color signifies the low amplitude/energy for that frequency component when compared against time whereas as dark color signifies high/amplitude or energy.

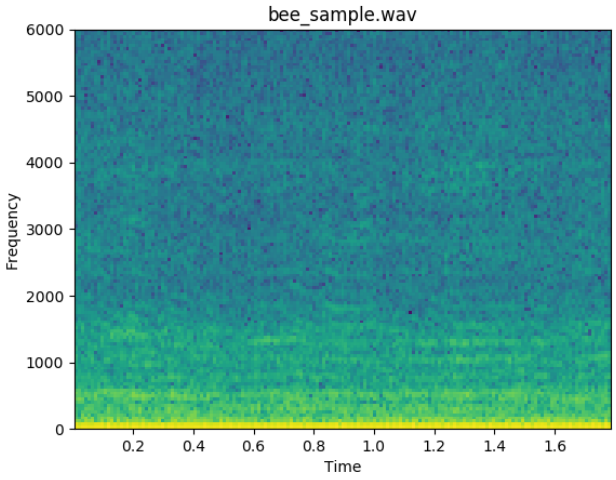

Fig. 3.4: Spectrogram for bee sample

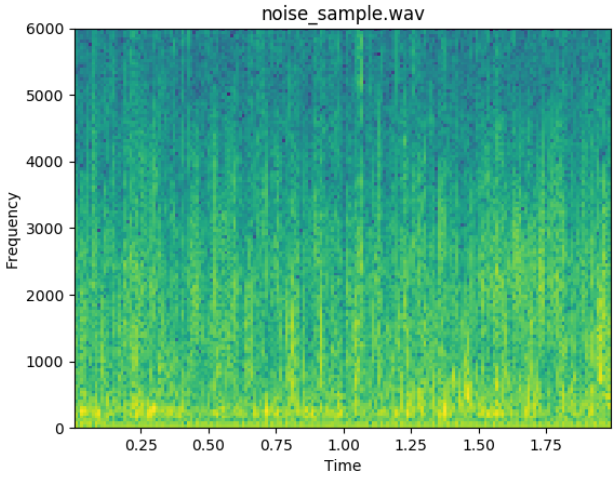

Fig. 3.5: Spectrogram for noise sample

3.4.2 Initial preprocessing approach - noise removal

The audio data collected from the BeePi installed during the season of May-August 2016 was very noisy. Also, for season 2017, it was noisy for few of the hives. Since any kind of feature extraction and processing on this data would have resulted in invalid results, thus the first task was to remove the noise from the audio samples. There were two processes that were tried to remove the noise.

Firstly, Wavelet transform technique is used to remove noise from the signals by deconstructing a signal into step function and wavelets by applying wavelet transformation and then constructing back the signal by leaving behind wavelets. Wavelets capture the difference of two narrower step functions, in our case, it corresponds to noise. The reconstructed signal should have been noiseless; however, the signal was still having noisy signals. This technique was used using pywavelet package which is an open source wavelet transform software for the Python programming language.

Audacity is a free open source digital audio editor which is also used for recording and other audio tasks [27]. Initially, noise removal was tried using the noise reduction feature in the Audacity. Firstly, a small sample of noise is taken to get the noise profile and then

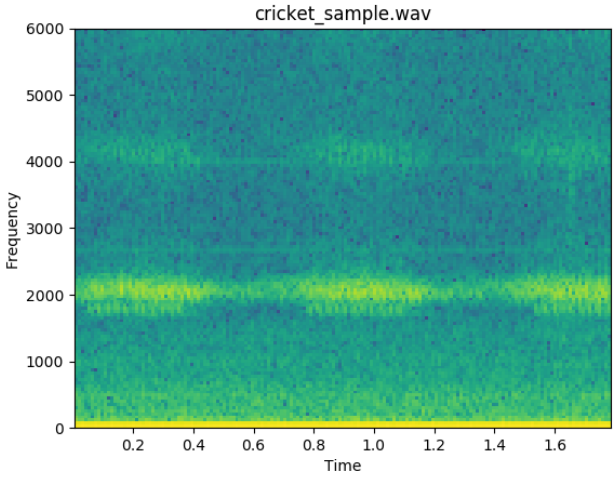

Fig. 3.6: Spectrogram for cricket sample

it was applied to the whole sample. This technique did not work as along with the noise, original signal i.e. bee signal also was removed. Another technique used was to identify and extract the two seconds samples from the 30 sec sample which most likely corresponds to the pure original signal. Though it worked but the efforts required to get such samples were huge as we must go again a single audio file multiple times to get a pure signal portion.

3.4.3 Preprocessing and labelling of data samples

Data was collected from six different hives for season 2017 as follows:

- Craig East (CE): Microphones inside
- Craig West (CW): Microphones inside
- Richard East (RE): Microphones inside
- Richard West (RW): Microphones inside
- Richard East Outside (REO): Microphones outside
- Richard West Outside (RWO): Microphones outside

The assumption for labeled data was that it should completely belong to any of the class-category i.e. a person should be able to clearly identify the class of a sample by listening to it. For hives CE, CW, RE and RW audio samples were not of good quality in a sense that it had a lot of microphones clicks, thus getting a pure sample of 2-seconds in any form (Bee, cricket, noise) was a challenging task as most of the time we would get overlapping sounds. Audacity which is a free open source and audio editing software was used to extract 2-seconds chunks from 30 seconds audio samples. It was more time-consuming task since there use to be very few pure samples belonging to any one of the category.

While checking for REO and RWO, we found that data was very pure. After further analysis, we came to a conclusion that when microphones are kept outside the hives, it gives good quality audio samples in comparison to ones where the microphones are put inside. For season 2018, we will be placing microphones outside for all the hives. Utilizing this observation, Python scripts were written to automatically chunk 30 sec audio sample into 28 overlapping samples. Script also took care of labelling, indexing the samples and putting them in respective category folders automatically. The scripts reduced the manual labelling efforts for getting the sample data to feed to machine learning models to 30 percent.The labelling task can be categorized into three steps:

- Selecting an audio file and listening it to get a good idea about the quality of that sample, whether whole sample of 30 sec can be utilized or just a part of it is useful.
- After selecting a sample, running a Python script to break it down into 2-seconds 28 overlapping samples.
- Renaming and indexing the samples based on their class by using a Python script to make them more readable and easy to use.

For three-way classification problem, total of 9110 two seconds of audio samples were utilized as shown in Table 3.1 whereas for time of day classification problem, total of 12,600 two seconds samples were utilized as shown in Table 3.2.

Table 3.1: Sample distribution for three-way classification problem

Bee	3000 samples
Noise	3110 samples
Cricket	3000 samples

Table 3.2: Sample distribution for time of day classification problem

class 0-5	2100 samples	[0.00 am to 5.00 am)
class 5-9	2100 samples	[5.00 am to 9.00 am)
class 9-13	2100 samples	[9.00 am to 13.00 pm)
class 13-17	2100 samples	[13.00 pm to 17.00 pm)
class 17-20	2100 samples	[17.00 pm to 20.00 pm)
class 21-00	2100 samples	[21.00 am to 0.00 pm)

3.5 Audio features extraction

Features are the essential ingredients for solving any machine learning problem. A machine learning model can perform well only if the features fed to it are good and relevant.In this study, focus has been on extracting spectral features of an audio signal.

All the features have been extracted using Librosa. Librosa is a Python package for music and audio analysis. It provides an implementation for a wide range of audio features and thus acts as a building block necessary to create audio classification systems [28]. Function for extracting all the features from an audio sample and creating a feature vector have been given in Fig 3.9. These feature vectors are then fed to the classifiers. Feature extraction process is inspired from [29]. Total length of a feature vector is 193 units (40 + 12 + 128 + 7 + 6)

MFCC - Mel frequency cepstral coefficients (Mean, axis=0) - Length (40)

Chroma sfft (Mean, axis=0) Length (12)

Mel Spectrogram (Mean, axis=0) Length (128)

Spectral contrast (Mean, axis=0) Length (7)

Tonnetz (Mean, axis=0) Length (6)

STFT - Short Time Fourier Transform (absolute values, 1000), Not directly fed to classifier. This feature is used to extract spectral contrast and chroma features

- MFCCs

Mel Frequency Cepstrum (MFC) encodes the power spectrum of a sound whereas the Mel-frequency cepstral coefficients [MFCCs] collectively make up the MFC. MFCC coefficients model the spectral energy distribution in a more persistent and meaningful way, thus they are most widely feature for speech recognition, music information retrieval and audio classification. The standard MFCC implementation follows the steps [30–34] shown in Fig 3.7

For extracting the coefficients using Librosa library, following parameters needs to be taken in account where

librosa.feature.mfcc(y=None, sr=22050, S=None, n-mfcc=40)

y : NumPy array for the audio signal

sr : sampling rate of the signal y

n-mfcc : number of MFCC coefficients to return

- Chroma-stft

Chroma-stft computes a chromagram from a waveform or power spectrogram. Chroma features are powerful representation for an audio signal where the entire spectrum is projected onto 12 bins representing the 12 distinct semitones of the musical octave. Thus, they represents 12 distinct musical chroma of octave corresponding to each time frame. For processing, an audio signal is converted into a sequence of chroma features each expressing how the short-time energy of the signal is spread over the twelve chroma bands [35–37]. For extracting chroma features using Librosa library, following parameters needs to be taken in account where

librosa.feature.chroma-stft(S=stft, sr=sample-rate)

S: Power spectrogram, in our case we have utilized short-time Fourier transform

sr : sampling rate of the signal

n-fft (FFT window size) and hop-length is required only when y is provided

- Spectral Contrast

This refers to Octave-based Spectral Contrast feature that represents the spectral characteristics of an audio sample. Features like MFCC uses an average spectral envelope to represent the spectral characteristics of audio. This kind of features averages the spectra in each sub-band and reflects the average spectral characteristics, but it could not represent the relative spectral characteristics in each sub-band, which seem more important to discriminate different types of audio.

Thus, Octave-based Spectral Contrast takes care of the relative spectral characteristics of audio. It considers the strength of spectral peaks and spectral valleys in each sub-band separately, so that it could represent the relative spectral characteristics, and then roughly reflect the distribution of harmonic and nonharmonic components. The block diagram in Fig 3.8 shows the procedure to get the Octave-based spectral contrast feature [38, 39].

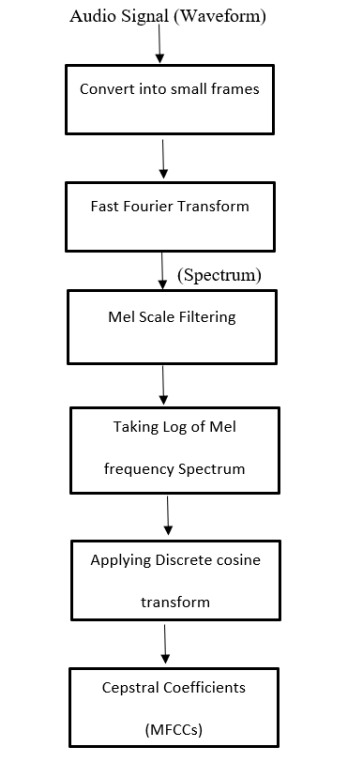

Fig. 3.7: MFCC implementation

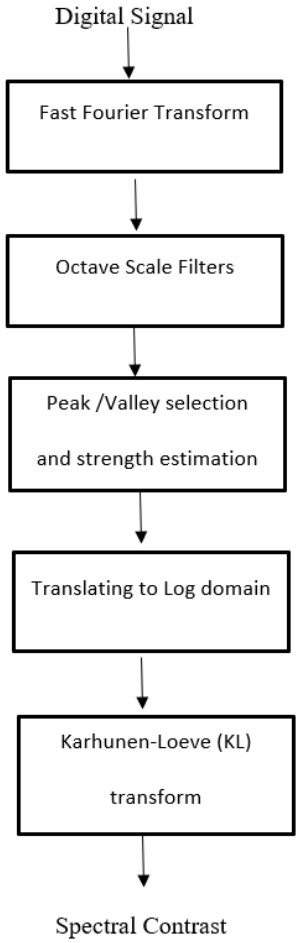

Fig. 3.8: Octave based spectral contrast feature

- Tonnetz

The harmonic network or the Tonnetz is a conceptual lattice diagram representing the planar representations of pitch relations. It can be utilized to show traditional harmonic relationships. The tonnetz feature extraction function utilized through Librosa computes the tonal centroid features whose implementation is based on [40]. A novel feature detection for audio data was proposed in this paper. It presents a process in the form Harmonic Change Detection Function (HCDF) for detecting changes in the harmonic content of audio signals which is based on a new model for equal-tempered tonal space. HCDF system comprises of

several distinct elements. The process starts with applying Constant-Q Transform on the audio data which is followed by a 12-semi-note Chromagram decomposition. A harmonic Centroid transform is then applied to the Chroma vectors which is then smoothed with a Gaussian filter before the Euclidian distance measure is calculated. The HDFS finds many applications as a pre-processing stage for further harmonic recognition and classification algorithms [41]. Librosa usage as follows:

librosa.feature.tonnetz(y=librosa.effects.harmonic(X), sr=sample-rate)

Tonal centroid features for each frame are returned in the form of 6D vector. Tonnetz function returns the Tonal centroid features for each frame. Tonnetz dimensions as follows:

0: Fifth x-axis 3: Minor y-axis

1: Fifth y-axis 4: Major x-axis

2: Minor x-axis 5: Major y-axis

- Melspectogram

Mel-frequency analysis of an audio signal is based on human perception. It is observed that human ear acts as a filter as it concentrates only on certain frequency components. These filters are non-uniformly spaced on the frequency axis i.e. more filters in the low-frequency regions whereas less number of filters in the high-frequency regions. The mel-scale is a perpetual scale of pitches judged by listeners to be equal in distance from one another. It mimics the non-linear human ear perception of sound, by being more discriminative at lower frequencies and less discriminative at higher frequencies. An object of type MelFilter represents an acoustic time-frequency representation of sound: the power spectral density P(f,t), expressed in dBs. It is sampled into a number of points around equally spaced times t and frequencies f on a Mel frequency scale [42–44]. The Hertz(f) and Mel(m) can be converted to each other using below equations

$$m = 2595 log 10(1 + f/700)$$

$$f = 700(10m/25951)$$

Librosa Melspectogram feature extraction library computes a Mel-scaled power spectrogram. If a spectrogram input S is provided to the function, then it is mapped directly to the Mel scale to compute the power spectrogram whereas if a time series input is provided, then first magnitude spectrogram is computed and then mapped to the Mel scale.

- Short-time Fourier transform

The short-time Fourier transform (STFT) is a Fourier-related transform used to determine the sinusoidal frequency and phase content of local sections of a signal as it changes over time. The general procedure for computing STFTs is to divide a longer time signal into shorter segments of equal length and then compute the Fourier transform separately on each shorter segment. This allows us to get the Fourier spectrum on each shorter segment. Changing spectra is then plotted as a function of time [45]. In our feature extraction process, STFT has been used to extract spectral-contrast and chroma features.

```
def create_mother_vector(file_name):

    X, sample_rate = librosa.load(file_name)

    stft = np.abs(librosa.stft(X))
    mfccs = np.mean(librosa.feature.mfcc(y=X, sr=sample_rate, n_mfcc=40).T,axis=0)
    chroma = np.mean(librosa.feature.chroma_stft(S=stft, sr=sample_rate).T,axis=0)
    mel = np.mean(librosa.feature.melspectrogram(X, sr=sample_rate).T,axis=0)
    contrast = np.mean(librosa.feature.spectral_contrast(S=stft, sr=sample_rate).T,axis=0)
    tonnetz = np.mean(librosa.feature.tonnetz(y=librosa.effects.harmonic(X),
    sr=sample_rate).T,axis=0)

    mother_vector = np.hstack([mfccs,chroma,mel,contrast,tonnetz])

    head, tail = os.path.split(file_name)
    add_path = os.path.join(head,"mother_files")
    name_file,ext = os.path.splitext(tail)
    new_name = name_file + ".mvector"
    new_path_file = os.path.join(add_path,new_name)
    np.save(new_path_file,mother_vector)
```

Fig. 3.9: Code snippet for extracting audio features

3.6 Model selection: classification models

Supervised learning is a machine learning technique that deals with tasks where we try to predict unknown data attributes based on the known attributes of an entity. Each observation is treated as a pair of feature object and the target object where the feature

object will always be known but the target object is known only for the training examples. Thus, the task is to predict the target object for the out of sample data based on the learning of the model from the training dataset. In other words, when we have an input variable (x) and an output variable (y), we use an algorithm that can learn a mapping function from the input to the output [46] i.e. for Y=f(X) where f will be the mapping function. This is known as a supervised learning because the learning process is taking place with an aim to get a fixed or definite outcome from the set of desired outcomes.

Classification problem falls under the supervised machine learning approach where a model classifies a new observation into one of the predefined class labels or categories. It is a task of approximating a member function that could map an input observation(x) to one of the discrete out variables(y) [47]. This function approximation or the model training is done using the training datasets for which the output class labels are known. Problems that we are dealing with are classification problems because we have a fixed set of outcomes i.e. three categories for 3-way classification problem (bee, cricket and noise) and six categories for different times of the day for day classification problem.

3.6.1 Logistic regression

Logistic Regression is a regression model which is used when the target variable is categorical whereas the predictors (independent variable) can be continuous or categorical. It comes up with the sigmoid function whose coefficients are estimated while training using the training dataset which then predicts the probability of test samples for each target category and assigns a category with the highest probability to this test instance when used for multi-class classification problem.A logistic regression equation is like:

$$y = e^{b0+b1*x}/(1 + e^{b0+b1*x})$$

Here y is the predicted output or the target value, b0 is the bias or intercept term and b1 is the coefficient for the single input value (x). Each feature or the input variable will have an associated b coefficient (a constant real value) that must be learned from our

training dataset [48].

3.6.2 Random forest

A decision tree is a flowchart-like graphical depiction of a decision and every possible outcome to make that decision. On the other hand, Random Forest is an ensemble method which makes use of decision trees. In this method, multiple models are built using a subset of features and trained on different training sets which allow trees to decorrelate from each other. Each model is then used individually to predict an independent result, after which a vote is taken to predict the best class in case of classification problems. Decision trees are more prone to suffer from high variance or high bias whereas Random Forest takes care of the bias-variance tradeoff by finding out a balance between the two extreme sides [49].

3.6.3 K-nearest neighbor

K-Nearest Neighbor is a supervised learning method that makes no assumption at all about the probability distribution of the feature vectors. For the classification problems, the training instances are stored in the feature space with their class labels. For a test instance, the distance from it to all the training points in the dataset are calculated and stored in the sorted order. It is then classified by a majority vote of its neighbors, with the instance being assigned to the class most common among its k-nearest neighbors, where k is positive integer passed as a parameter. Below is the Figure 3:10 illustrating the concept of k-NN classification [50].

In Figure 3:10, green circle refers to a test sample whereas the blue square and the red triangle refers to the two output classes. If k = 3, the area within search will be up to the solid boundary. The test sample will be assigned to red class as there are two red against one blue. For k = 5, it will be blue class as there are three blue against two red.

3.6.4 Support vector machine with OneVsRestClassifier

OneVsAll, also known as the One-vs-the-rest strategy, consists of fitting one classifier per class. For each classifier, the class is fitted against all the other classes i.e. for N

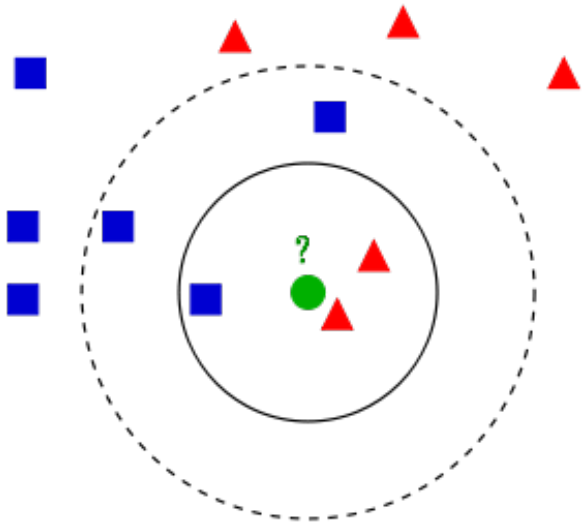

Fig. 3.10: Finding the category of test sample using K-NN classification

classes, there will be N different binary classifiers, each trained against the remaining N-1 classes. This helps in breaking down multi-class classification problem into multiple binary classification problems. Thus, also helps in gaining knowledge about a class by inspecting its corresponding classifier [51]. Some classification algorithm such as Logistic Regression can be used for multi-class label problems whereas others are restricted to only binary classification problem i.e. outcome can be out of the two classes. Thus, using this approach, all such algorithms can be used for the multi-class problems.

A Support Vector Machine is a supervised machine-learning approach used to build linear, non-probabilistic binary classifiers. It explicitly requires training instances for learning of the model in which instances are represented in a feature space (n-dimensional - hypercube) and a hyperplane boundary is looked for which is a linear function, that can separate the two categories in a best possible way so that the distance from it to the nearest data point on each side is maximized. The new test examples are then assigned one or the other category by finding on which side of the boundary they fall in the feature space [52]. In this thesis, SVC implementation of SVM with linear kernel have been utilized [53, 54] to deal with multi-class classification. Fig 3.11 displays the three possible hyperplanes (H1, H2, H3). Since H3 hyperplane maximizes the distance between the two classes, it will be

the ideal choice for consideration.

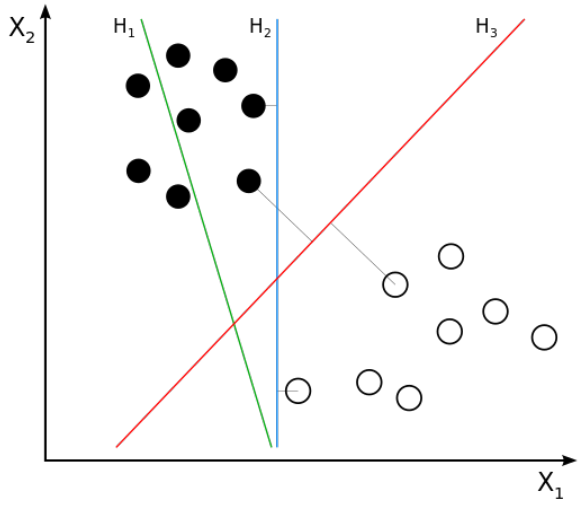

Fig. 3.11: Svm - finding hyperplane separating the two classes in best way

CHAPTER 4
EXPERIMENTS

4.1 Overview

This chapter describes the experiments we conducted to evaluate the accuracy of models used to classify audio samples in various categories and classes. The audio samples belonged to two Northern apiaries captured. The detailed data collection process is described in section 3.4. The ground truth was obtained in the form manually labelled data. Two human evaluators worked on labelling the data who listened to each audio sample of 30 seconds and then placed them in different categories for further pre-processing. Each evaluator was given specific instructions on how to identify whether a given sample falls under the category of bee, noise or cricket. Each phase of the experimental procedure has been described in detailed in the below sections.

4.2 Experimental procedure

This research participates in a complete data science loop, starting from data preprocessing to data prediction. Data was collected from BeePi Monitors from a period of May 2017 to September 2017. An audio sample of 260 30 sec wav files (2 cycles of Anker battery) was taken and automatically segmented into 28 2-second overlapping audio samples. For three-way classification problem, each 2-second sample was manually labelled into three categories (bee buzzing, cricket chirping and ambient noises) which resulted in a total of 9100 samples. For day classification problem, each 2-second sample was categorized into six categories (class 0-5, class 5-9, class 9-13, class 13-17, class 17-20, class 21-00) with a total sample count of 12,600. As a part of Feature engineering tried out an experimental/analytical approach to select best possible audio features. using Python and its scientific stack. Five audio features were extracted for each audio sample to create a feature vector

of length 193 units which were then fed into machine learning models. All the training was done with scikit-learn package of Python 2.7. The flow diagram in Fig 4.1 mentions all the phases of our experimental procedure, starting from data collection and ending at prediction.

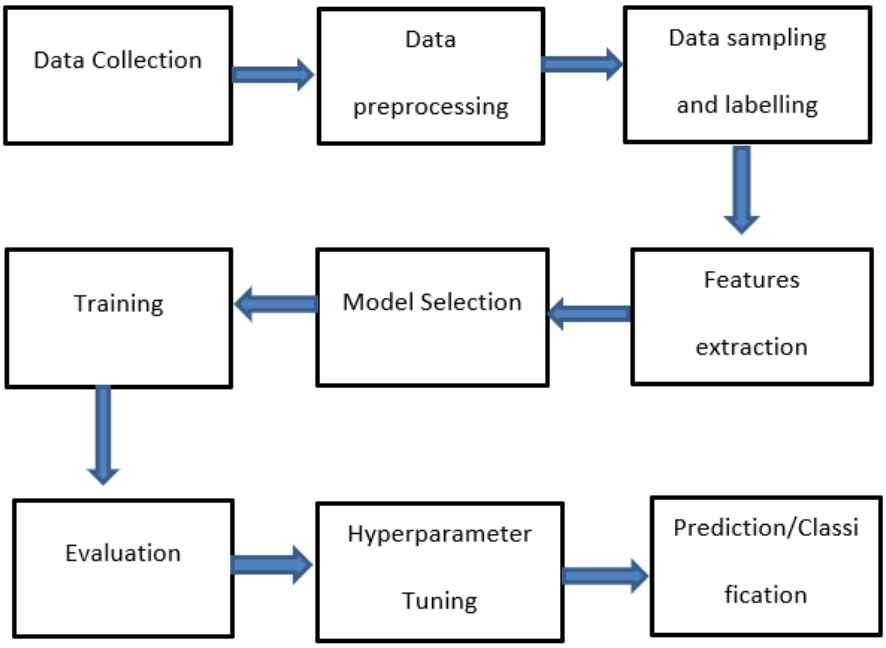

Fig. 4.1: Flow diagram for the experimental procedure

4.3 Preprocessing of extracted features

Preprocessing or feature scaling is used to standardize the predictors or the features of the data. Features are at different level of magnitude i.e. they are at different scale and when we train algorithms such as SVM or others which are sensitive to scaling of data, they may not perform correctly as few of the features which are dominant in magnitude will only govern the outcome and the other features contribution will be minimum leading to incorrect model. Thus, scaling at same level helps to properly train our algorithms.

All the features preprocessing is done using the scikit-learn preprocessing package. Types of scaling performed are mentioned below [55–57].

1. **No scaling**

   In this approach, after extracting features from an audio sample, we dont do any kind of pre-processing, the raw feature vector is directly fed to the classifier.

   X - Feature matrix, will be unscaled. We will feed this matrix to our classifier for training/testing purpose.

2. **Standard Scaling**

   In this approach, the result of standardization (or Z-score normalization) is that the features will be rescaled so that theyll have the properties of a standard normal distribution with mean as 0 and standard deviation as 1.

   Standard scores (also called z scores) of the samples are calculated as follows:

$$Z = (x - \mu)/\sigma$$

   Standardizing the features helps in making them centred around 0 with a standard deviation of 1. This helps in bringing everything to the same unit scale before feeding the matrix to the machine learning model. This makes sure that all the values contribute equally.

3. **Min-Max scaling**

   In this approach, data is scaled to a fixed range of [0,1). This helps in compressing the standard deviations which in turn suppresses the effect of outliers. A Min-Max scaling is typically done via the following equation where the transformation i.e. performed around axis = 0

$$Xnorm = (X - Xmin)/(Xmax - Xmin)$$

4. **Normalization**

   It is the process of scaling individual samples to have unit form. There are two types of normalization L1 norm and L2 norm.

   L1 norm: Least absolute deviation (total sum of absolute values of each attribute for each row = 1). It refers to minimizing the sum of absolute differences (S) between the target value (Y) and the estimated value f(x). It is sensitive to outliers.

$$S = \sum_{i=1}^{n} |(y_i - f(x_i)|$$

   L2 norm: Least squares (total sum of squares of each attribute values for each row equal to 1). It refers to minimizing the sum of the square of the differences (S) between the target value (Y) and the estimated value f(x). It takes outliers into consideration

$$S = \sum_{i=1}^{n} (y_i - f(x_i))^2$$

4.4 Training of the models

All the model training has been achieved using scikit-learn module [58]. It is a machine learning library for Python that provides a range of supervised and unsupervised learning algorithms. Four supervised learning models have been utilized for carrying out the experimentation process.

- Logistic Regression
- Random Forests
- K-Nearest Neighbors
- Support Vector Machines

4.5 Model evaluation procedures

There are two model evaluation techniques have been utilized for the experimentation process

- Train/test split procedure

  This procedure allows us to split the whole dataset into two pieces, one piece on which we do our training, and, on another piece, we do the testing. It is simple as well as flexible as it allows us to set the train size and the test size for various experimentation purposes. Throughout the process, the training process, model is not exposed to test dataset, thus any predictions made on the test dataset later will be an indicator of the performance of our model. The splitting is done using scikit learn with Train size = 0.6 , Test size = 0.4 and random state is equal to 4.

- K fold cross validation In the K fold cross-validation technique, the original sample is randomly divided into k folds or sub samples of the dataset (K number given by user). Out of k folds then, k-1 folds are used for training and one-fold is retained for testing/validation. This process is repeated k times with each fold getting used for testing purpose exactly once. The accuracy of model on each test fold is then averaged across to get a final accuracy. This gives us a better unbiased view as we have utilized the complete dataset for training/testing purpose. All the experiments have been performed with K values as 10. Fig 4.2 demonstrates the cross-validation process with value of K as 4 [59].

4.6 Hyperparameter tuning

A machine learning model is built over various parameters and constraints. Depending upon the problem in focus, the same machine learning model may require different constraints, weights or learning rates to generalize different data patterns or to generate better predictions. These measures are called hyperparameters and have to be tuned so that the model can optimally solve the machine learning problem [60].

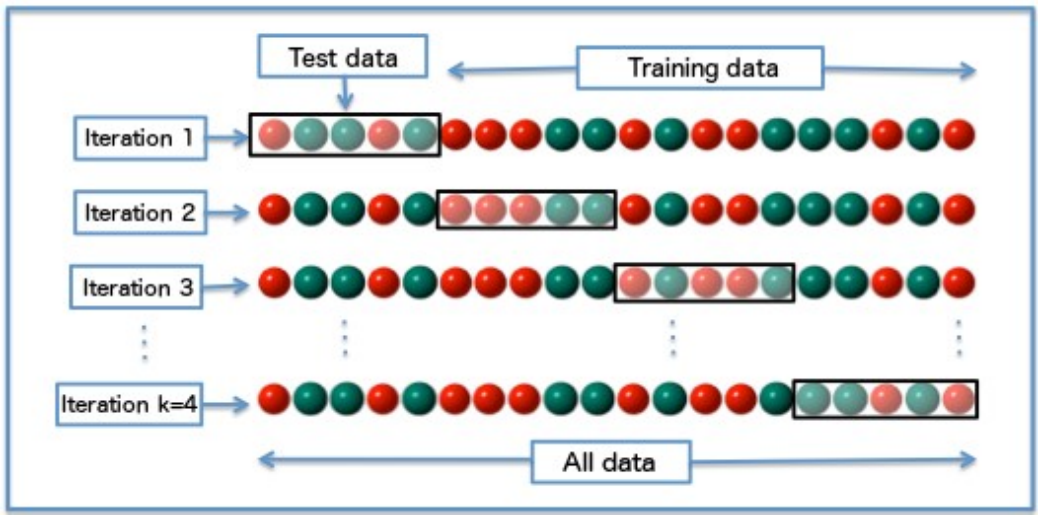

Fig. 4.2: Demonstrating cross validation with k = 4

Grid search is one of the parameter tuning method which has been utilized for the experiments using the scikit-learn library implementation. Given a subset of parameters, it does an exhaustive search with all the permutation-combinations of the parameters and returns the parameters that provide best results.

4.7 Evaluation metrics

There are three model evaluation metrics have been utilized for the experimentation process

- Classification accuracy

  It is the ratio of number of correct predictions to the total number of all the predictions, and when multiplied by 100 gives us the percentage accuracy.

  Accuracy percent = (Correct predictions/ Total number of predictions) * 100

  It is the easiest way to evaluate a classification problem but does not provide much depth to the underlying distribution of predicted values. Thus, does not indicate the type of errors our model is making such as False positives or False negatives.

Table 4.1: Confusion matrix

(Label)	Predicted (0)	Predicted (1)
Actual (0)	True Negative	False Positive
Actual (1)	False Negative	True Positive

- Confusion matrix Confusion matrix is a metric that allows us to visualize the predictions of a classification model in a table layout. It gives us the distribution of labels that our classifier has predicted over the test dataset. Thus, it gives a complete picture of type of errors that our model is making i.e. True Positives, True Negatives, False positives and False negatives can easily be calculated by interpreting the matrix. For a binary classification problem, it can be viewed as

  Accuracy percent = (TP + TN) / (TP + TN + FP + FN) * 100

- Receiving Operating Characteristic (ROC)

  ROC curves plot the True Positive Rate (Sensitivity) on Y axis against a False Positive Rate (1- Specificity) on X axis . For an ideal classifier, the Area Under the Curve will be 1.0. It helps us to analyze the tradeoff between sensitivity and specificity while dealing with a binary problem. To convert a multi-class problem to a binary problem, we have used OneVsRest classifier. Fig 4.3 shows a ROC curve [61].

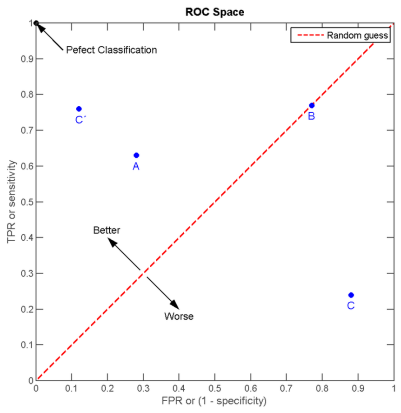

Fig. 4.3: Understanding ROC curve

CHAPTER 5

RESULTS

Four machine learning algorithms were used to train the classification models with a 60/40 train/test data split as well as the K-fold cross-validation (K = 10) strategy separately: Logistic Regression, K-Nearest Neighbor (KNN), OneVsRestClassifier (using Support Vector Machines) and Random Forests. The best accuracy of 99.97 percent for three-way classification problem was achieved using Random Forest no scaling. For day classification, the best accuracy achieved was 96.18 percent using KNN classifier with K = 1 and Normalize L2 feature scaling.

5.1 Three-way classification problem

Total time taken for creating 9110 .npy files (representing feature object) from 9110 audio files was 40 minutes.

- **Logistic regression**

  Experimental results are covered from Table 5.1 to 5.6

- **K-nearest neighbor**

  Experimental results are covered in Table 5.7. Fig 5.1 refers to variation of testing accuracy against different values of k for knn for different pre-processing methods when applied using train-test procedure

- **Random forest**

  Experimental results are covered in Table 5.8. Since, Random Forest doesn't requires feature scaling, only raw features have been fed to the classifier.

- **OneVsRestClassifier - Svm**

  Experimental results are covered in Table 5.9. Fig 5.2 refers to the ROC curves

Table 5.1: Classification accuracy with Logistic regression

Index	Classifier Type	Testing procedure	Scaling Type	Classification Accuracy
1	Logistic Regression	Train/test	No scaling	99.83%
2	Logistic Regression	Train/test	Standard scaling	99.86%
3	Logistic Regression	Train/test	Min max scaling	99.34%
4	Logistic Regression	Train/test	Normalize L1	90.64%
5	Logistic Regression	Train/test	Normalize L2	93.11%
6	Logistic Regression	K-fold , K =10	No scaling	99.91%
7	Logistic Regression	K-fold , K =10	Standard scaling	99.89%
8	Logistic Regression	K-fold , K =10	Min max scaling	99.63%
9	Logistic Regression	K-fold , K =10	Normalize L1	92.34%
10	Logistic Regression	K-fold , K =10	Normalize L2	93.77%

Table 5.2: Three-way classification : confusion matrix for Logistic regression 1st experiment

	Bee	Noise	Cricket
Bee	1174	4	0
Noise	2	1243	0
Cricket	0	0	1221

Table 5.3: Three-way classification : confusion matrix for Logistic regression 2nd experiment

	Bee	Noise	Cricket
Bee	1176	2	0
Noise	1	1242	2
Cricket	0	0	1221

Table 5.4: Three-way classification : confusion matrix for Logistic regression 3rd experiment

	Bee	Noise	Cricket
Bee	1172	6	0
Noise	14	1227	4
Cricket	0	0	1221

Table 5.5: Three-way classification : confusion matrix for Logistic regression 4th experiment

	Bee	Noise	Cricket
Bee	1178	0	0
Noise	64	927	254
Cricket	23	0	1198

Table 5.6: Three-way classification : confusion matrix for Logistic regression 5th experiment

	Bee	Noise	Cricket
Bee	1176	2	0
Noise	57	1038	150
Cricket	41	1	1179

Table 5.7: Classification accuracy with K-nearest neighbor

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	KNN	Train/test	No scaling	99.86%
2	KNN	Train/test	Standard scaling	99.87%
3	KNN	Train/test	Min max scaling	99.91%
4	KNN	Train/test	Normalize L1	99.84%
5	KNN	Train/test	Normalize L2	99.89%
6	KNN	K-fold , K =10	No scaling	99.93%
7	KNN	K-fold , K =10	Standard scaling	99.94%
8	KNN	K-fold , K =10	Min max scaling	99.96%
9	KNN	K-fold , K =10	Normalize L1	99.95%
10	KNN	K-fold , K =10	Normalize L2	99.96%

Table 5.8: Classification accuracy with Random forest

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	Random Forest	Train/test	No scaling	99.97%
2	Random Forest	K-fold, K =10	No scaling	99.94%

Table 5.9: Classification accuracy with OneVsRestClassifier - Svm

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	OneVsRestClassifier	Train/test	No scaling	99.83%
2	OneVsRestClassifier	Train/test	Standard scaling	99.91%
3	OneVsRestClassifier	Train/test	Min max scaling	99.86%
4	OneVsRestClassifier	Train/test	Normalize L1	91.19%
5	OneVsRestClassifier	Train/test	Normalize L2	93.08%
6	OneVsRestClassifier	K-fold, K =10	No scaling	99.89%
7	OneVsRestClassifier	K-fold , K =10	Standard scaling	99.78%
8	OneVsRestClassifier	K-fold , K =10	Min max scaling	99.67%
9	OneVsRestClassifier	K-fold , K =10	Normalize L1	92.20%
10	OneVsRestClassifier	K-fold , K =10	Normalize L2	95.06%

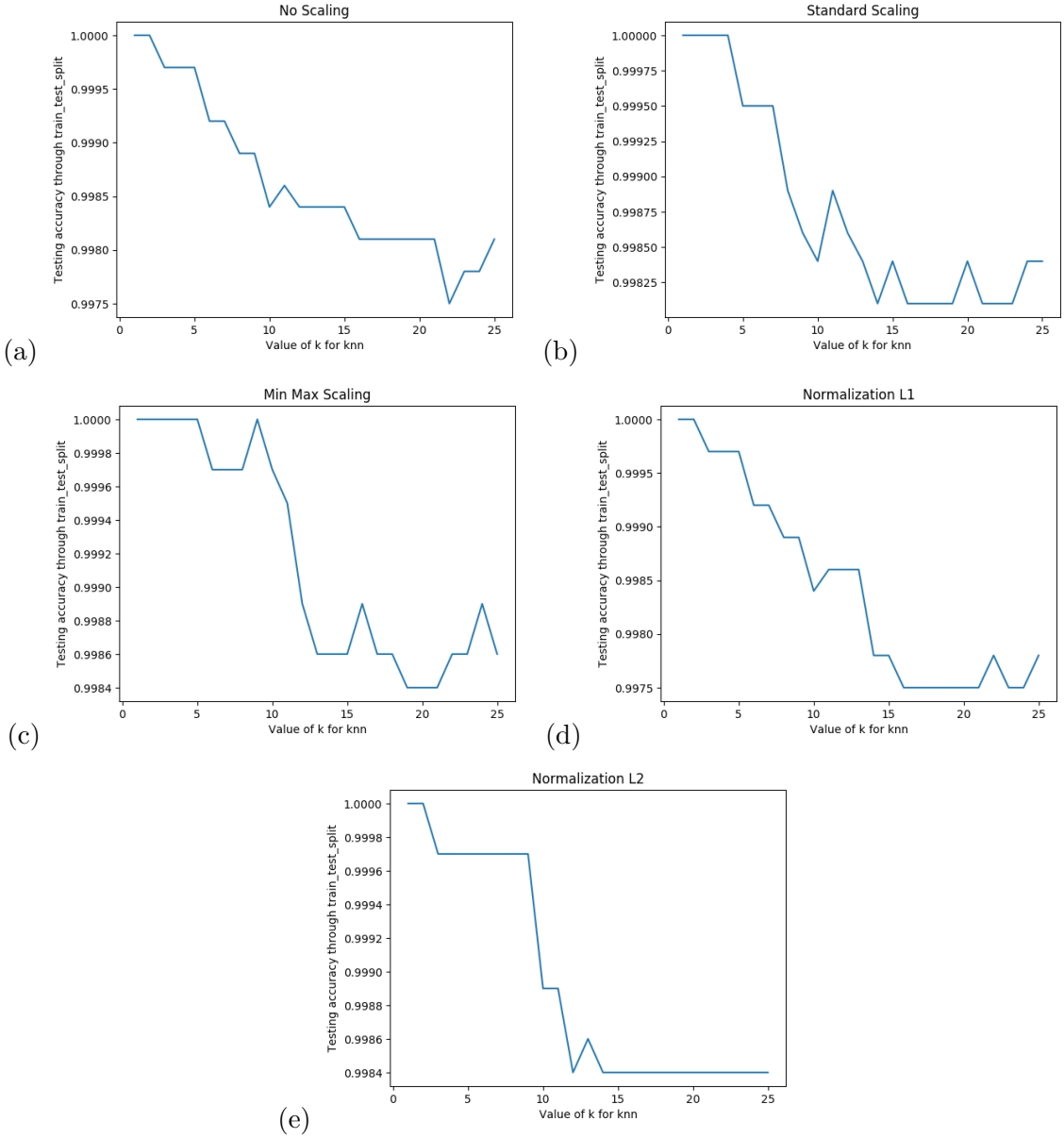

Fig. 5.1: Variation of testing accuracy against different values of k for knn for different pre-processing methods when applied using train-test procedure.

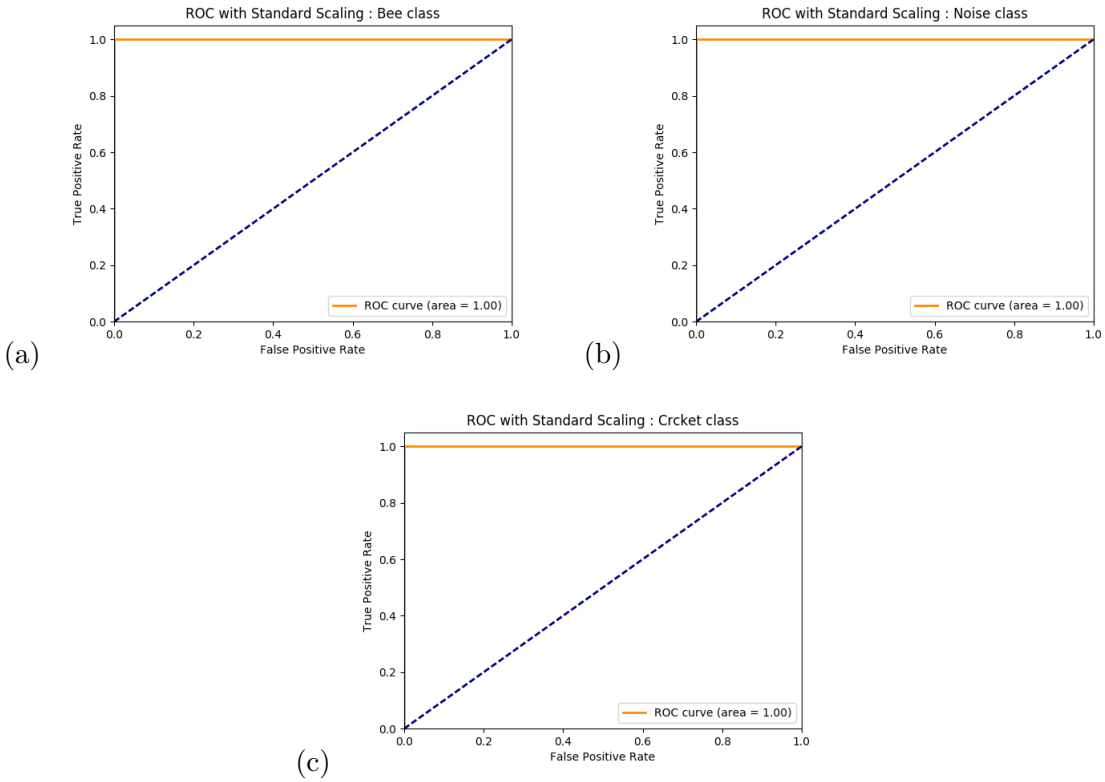

Fig. 5.2: ROC curves for bee, noise and cricket class using train/test procedure for Svm-OneVsRestClassifier

- **Out of sample data testing : sample data from RWO**

  Table 5.10 refers to experimental results corresponding to out of sample data. Total number of samples used were 1150 with 300 Bee samples, 350 Noise samples and 500 cricket samples. Since the noise and cricket data consist of many samples that can be considered in vice-versa category, thus a bit lower test accuracy for these two categories was expected.

- **Checked model tuning for K-NN using GridSearchCV for 2 parameters:**

  k-range = range(1,26) : values for k in range from 1 to 25, k is k parameter in knn

  weight-options=[uniform, distance] : how to give distribute weights

  Uniform: same weights for all distances

  Weighted: give more weight to nearby points in comparison to far points

Table 5.10: Classification accuracy on out of sample data

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	Logistic Regression	On Bee - Train/test	Standard scaling	99.66%
2	Logistic Regression	On Noise - Train/test	Standard scaling	85.31%
3	Logistic Regression	On Cricket - Train/test	Standard scaling	96%
4	Random Forest	On Bee - Train/test	No scaling	100%
5	Random Forest	On Noise - Train/test	No scaling	93.13%
6	Random Forest	On Cricket - Train/test	No scaling	87.6%

Table 5.11: Best parameters for K-NN using GridSearch

Accuracy	99.98%
N-Neighbors	3
Weight	Uniform

param-grid = dict(n-neighbors=[k-range], weights=[weight-options])

The function GridSearch goes with exhaustive testing with all possible permutation and combinations of these parameters and finally gives the best parameters to be used.Best Result came with the parameters (On raw feature matrix without any pre-processing) mentioned in Table.

5.2 Time of day classification problem

Total time for creating 12600 .npy file (consisting feature object) was 145 minutes (2 hrs 25 minutes)

- **Logistic regression**

  Experimental results are shown in Table 5.12 and confusion matrix in Table 5.17-5.18.

- **K-Nearest Neighbor**

  Experimental results are covered in Table 5.13. Checked model tuning for KNN using GridSearchCV for 2 parameters with min-max scaling. The best parameters are listed in Table 5.14. Fig 5.3 refers to variation of testing accuracy against different values of k for knn for different pre-processing methods when applied using train-test procedure.

Table 5.12: Time of day classification - accuracy with Logistic regression

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy
1	Logistic Regression	Train/test	No scaling	80.05%
2	Logistic Regression	Train/test	Standard scaling	84.63%
3	Logistic Regression	Train/test	Min max scaling	79.43%
4	Logistic Regression	Train/test	Normalize L1	56.94%
5	Logistic Regression	Train/test	Normalize L2	60.04%
6	Logistic Regression	K-fold , K =10	No scaling	79.75%
7	Logistic Regression	K-fold , K =10	Standard scaling	84.54%
8	Logistic Regression	K-fold , K =10	Min max scaling	79.94%
9	Logistic Regression	K-fold , K =10	Normalize L1	59.71%
10	Logistic Regression	K-fold , K =10	Normalize L2	62.45%

Table 5.13: Time of day classification - accuracy with K-nearest neighbor

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	KNN	Train/test	No scaling	88.68%
2	KNN	Train/test	Standard scaling	89.33%
3	KNN	Train/test	Min max scaling	91.06%
4	KNN	Train/test	Normalize L1	90.25%
5	KNN	Train/test	Normalize L2	91.88%
6	KNN	K-fold , K =10	No scaling	93.96%
7	KNN	K-fold , K =10	Standard scaling	94.36%
8	KNN	K-fold , K =10	Min max scaling	95.58%
9	KNN	K-fold , K =10	Normalize L1	94.89%
10	KNN	K-fold , K =10	Normalize L2	96.18%

Table 5.14: Time of day classification - best accuracy using GridSearchCV for K-NN

Accuracy	99.95%
N-Neighbors	1
Weight	Uniform

- **Random forest**

  Experimental results are covered in Table 5.15

Table 5.15: Time of day classification - accuracy with Random forest

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	Random Forest	Train/test	No scaling	91.76%
2	Random Forest	K-fold, K =10	No scaling	93.98%

- **OneVsRestClassifier - Svm**

  Experimental results are covered in Table 5.16

Table 5.16: Time of day classification - accuracy with Svm OneVsRestClassifier

Index	Classifier Type	Testing procedure	Scaling Type	Accuracy(Mean)
1	OneVsRestClassifier	Train/test	No scaling	84.27%
2	OneVsRestClassifier	Train/test	Standard scaling	87.57%
3	OneVsRestClassifier	Train/test	Min max scaling	84.53%
4	OneVsRestClassifier	Train/test	Normalize L1	50.85%
5	OneVsRestClassifier	Train/test	Normalize L2	60.12%
6	OneVsRestClassifier	K-fold , K =10	No scaling	85.30%
7	OneVsRestClassifier	K-fold , K =10	Standard scaling	89.67%
8	OneVsRestClassifier	K-fold , K =10	Min max scaling	84.43%
9	OneVsRestClassifier	K-fold , K =10	Normalize L1	50.91%
10	OneVsRestClassifier	K-fold , K =10	Normalize L2	63.22%

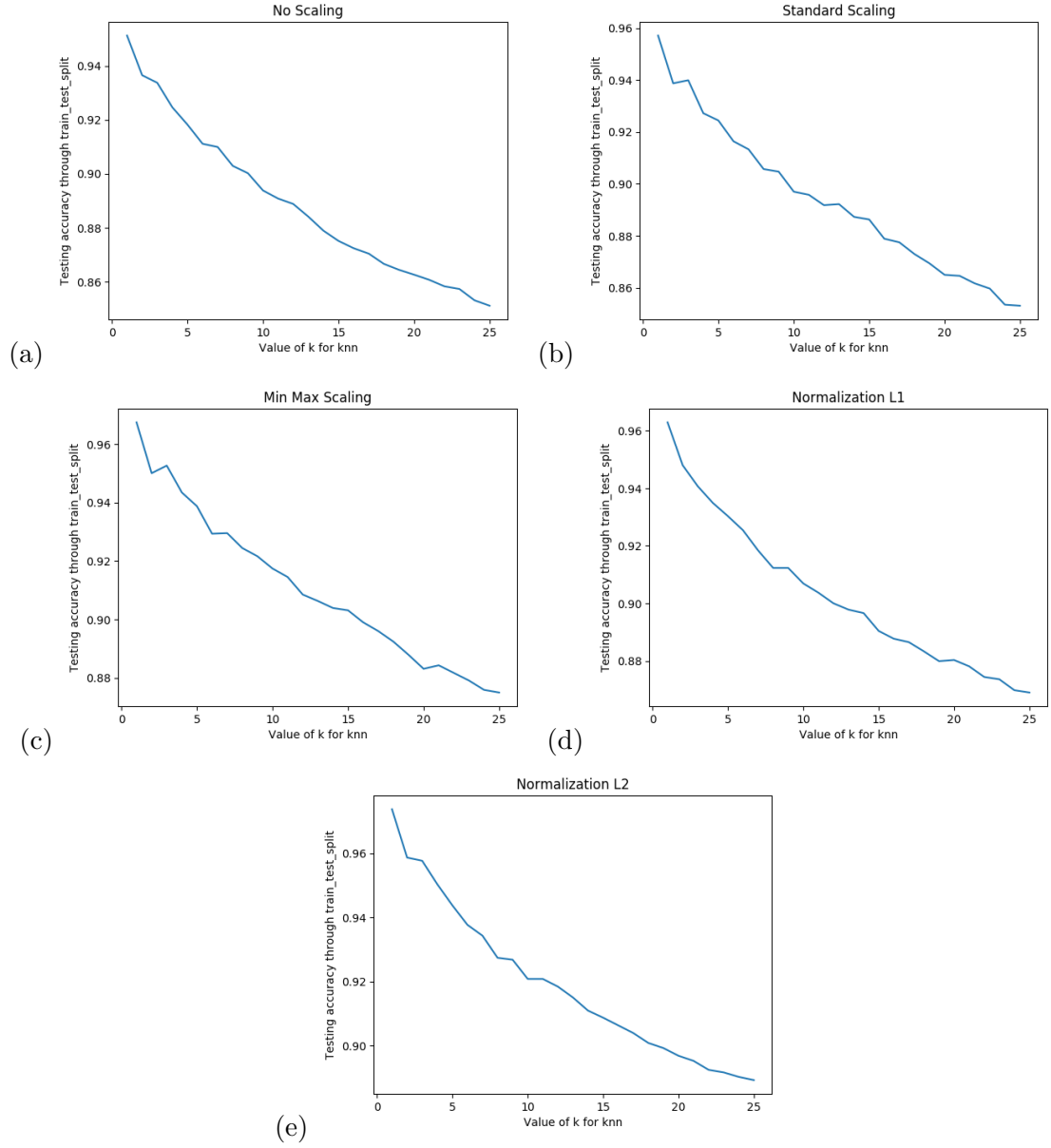

Fig. 5.3: Variation of testing accuracy against different values of k for knn for different pre-processing methods when applied using train-test procedure

Table 5.17: Time of day classification best performance : best confusion matrix for Logistic regression with standard scaling

	class 0-5	class 5-9	class 9-13	class 13-17	class 17-20	class 21-00
class 0-5	778	35	0	0	0	13
class 5-9	26	764	12	1	16	24
class 9-13	1	21	708	69	61	0
class 13-17	0	5	98	609	94	0
class 17-20	3	14	77	138	608	10
class 21-00	40	13	0	0	3	797

Table 5.18: Time of day classification worst performance: confusion matrix for Logistic regression with normalize L1

	class 0-5	class 5-9	class 9-13	class 13-17	class 17-20	class 21-00
class 0-5	618	60	0	0	2	137
class 5-9	187	555	2	2	65	0
class 9-13	2	76	172	511	99	0
class 13-17	0	4	21	740	41	0
class 17-20	2	119	31	457	241	0
class 21-00	131	168	0	0	11	543

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

This research utilizes the bee buzzing signals in the form of audio data which is a good source to estimate the state of a bee colony. A systematic data science procedure is presented that starts with data acquisition and ends at audio samples classification, utilizing the spectral features of the audio for feeding the machine learning models. This study enables a path towards an automatic classification of audio samples in different classes and categories within Electronic Bee Hive Monitoring System. The result indicates that all four machine learning models perform on par and when used in BeePi can assist the beekeepers by reducing their manual audio monitoring efforts and can eventually help them to estimate the health of the hive.

In our future work, we propose to deploy the project onto the Raspberry Pi 3, which is part of the BeePi EBMD hardware and classify the samples in real-time at the time of audio capturing itself. We also plan to compare and analyze the audio data for multiple hives with an aim to distinguish a healthy hive from a dead hive.

REFERENCES

[1] "Honey Bee". [Online]. Available: https://en.wikipedia.org/wiki/Honey_bee

[2] "What would happen if bees went extinct". [Online]. Available: http://www.bbc.com/future/story/20140502-what-if-bees-went-extinct

[3] Dina Spector. "What Our World Look Like Without Honeybees". [Online]. Available: http://www.businessinsider.com/the-world-without-honeybees-2013-6

[4] Bryan Walsh. "The Plight of the Honeybee". [Online]. Available: http://content.time.com/time/subscriber/article/0,33009,2149141-1,00.html

[5] "Colony Collapse Disorder". [Online]. Available: https://en.wikipedia.org/wiki/Colony_collapse_disorder

[6] Susan Milius. "The mystery of vanishing honeybees is still not definitely solved". [Online]. Available: https://www.sciencenews.org/article/mystery-vanishing-honeybees-still-not-definitively-solved

[7] "United states patent application publication pub no us2010/0062683 A1 Mar 11, 2010."

[8] "Why do bees buzz". [Online]. Available: https://www.scientificamerican.com/article/why-do-bees-buzz/

[9] John Zsiray. "USU professor hopes BeePi hive sensors will help honeybees". [Online]. Available: https://news.hjnews.com/agriculture/usu-professor-hopes-beepi-hive-sensors-will-help-honeybees/article_7c205ecb-1d64-51d3-96c1-b1754de27d6f.html

[10] M. T. Sanford, "2nd international workshop on hive and bee monitoring." American Bee Journal, December 2014, pp. 1351–1353.

[11] "Arnia". [Online]. Available: http://www.arnia.co.uk/

[12] "HiveMind". [Online]. Available: http://hivemind.co.nz/

[13] D. Atauri Mezquida and J. Llorente Martnez, "Short communication.: Platform for bee-hives monitoring based on sound analysis. a perpetual warehouse for swarms daily activity," vol. 7, 12 2009.

[14] M. Ramsey, M. Bencsik, and M. I. Newton, "Long-term trends in the honeybee whooping signal revealed by automated detection," *PLOS ONE*, vol. 12, no. 2, pp. 1–22, 02 2017. [Online]. Available: https://doi.org/10.1371/journal.pone.0171162

[15] "Swarming". [Online]. Available: https://en.wikipedia.org/wiki/Swarming_(honey_bee)

[16] S. Ferrari, M. Silva, M. Guarino, and D. Berckmans, "Monitoring of swarming sounds in bee hives for early detection of the swarming period," *Computers and Electronics in Agriculture*, vol. 64, no. 1, pp. 72 – 77, 2008, smart Sensors in precision livestock farming. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169908001385

[17] M. Bencsik, J. Bencsik, M. Baxter, A. Lucian, J. Romieu, and M. Millet, "Identification of the honey bee swarming process by analysing the time course of hive vibrations," *Computers and Electronics in Agriculture*, vol. 76, no. 1, pp. 44 – 50, 2011. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0168169911000068

[18] J. R. Delgado-Contreras, J.-P. García-Vázquez, and R. F. Brena, "Classification of environmental audio signals using statistical time and frequency features," *2014 International Conference on Electronics, Communications and Computers (CONIELECOMP)*, pp. 212–216, 2014.

[19] V. A. Kulyukin, M. Putnam, and S. K. Reka, "Digitizing buzzing signals into a440 piano note sequences and estimating forager traffic levels from images in solar-powered, electronic beehive monitoring," in *Proceedings of the International MultiConference of Engineers and Computer Scientists*, vol. 1, 2016.

[20] G. Tzanetakis and P. R. Cook, "Musical genre classification of audio signals," *IEEE Trans. Speech and Audio Processing*, vol. 10, pp. 293–302, 2002.

[21] J. Salamon, C. Jacoby, and J. P. Bello, "A dataset and taxonomy for urban sound research," in *ACM Multimedia*, 2014.

[22] A. Qandour, I. Ahmad, D. Habibi, and M. Leppard, "Remote beehive monitoring using acoustic signals," vol. 42, pp. 204–209, 12 2014.

[23] "Acoustic signals". [Online]. Available: https://www.encyclopedia.com/science/news-wires-white-papers-and-books/acoustic-signals

[24] S. K. Reka, "A vision-based bee counting algorithm for electronic monitoring of langsthroth beehives (2016)," in *All Graduate Theses and Dissertations. 4960.* [Online]. Available: https://digitalcommons.usu.edu/etd/4960

[25] V. A. Kulyukin and S. K. Reka, "A computer vision algorithm for omnidirectional bee counting at langstroth beehive entrances," 2016.

[26] K. Shah, "Power analysis of continuous data capture in beepi, a solar-powered multi-sensor electronic beehive monitoring system for langstroth beehives (2017)," in *All Graduate Theses and Dissertations. 6507.* [Online]. Available: https://digitalcommons.usu.edu/etd/6507

[27] "Audacity". [Online]. Available: https://www.audacityteam.org/

[28] "LibROSA". [Online]. Available: https://librosa.github.io/librosa/

[29] A. Saeed. "Urban Sound Classification part 1". [Online]. Available: http://aqibsaeed.github.io/2016-09-03-urban-sound-classification-part-1/

[30] "Mel Frequency cepstrum". [Online]. Available: https://en.wikipedia.org/wiki/Mel-frequency_cepstrum/

[31] K. Mahkonen. "MFCCs and gammatone filter banks". [Online]. Available: http://www.cs.tut.fi/~sgn14006/PDF2015/S04-MFCC.pdf

[32] M. Lutter. "Mel-Frequency Cepstral Coefficients". [Online]. Available: http://recognize-speech.com/feature-extraction/mfcc

[33] "MFCC tutorial". [Online]. Available: http://www.practicalcryptography.com/miscellaneous/machine-learning/guide-mel-frequency-cepstral-coefficients-mfccs/

[34] B. Logan, "Mel frequency cepstral coefficients for music modeling," 11 2000.

[35] M. Mller and S. Ewert, "Chroma toolbox: Matlab implementations for extracting variants of chroma-based audio features," in *Proceedings of the 12th International Conference on Music Information Retrieval (ISMIR), 2011. hal-00727791, version 2 - 22 Oct 2012*.

[36] D. P. Ellis, "Chroma feature analysis and synthesis," 04 2007. [Online]. Available: https://labrosa.ee.columbia.edu/matlab/chroma-ansyn/

[37] "Chroma feature". [Online]. Available: https://en.wikipedia.org/wiki/Chroma_feature

[38] J. Yang, F.-L. Luo, and A. Nehorai, "Spectral contrast enhancement: Algorithms and comparisons," *Speech Communication*, vol. 39, no. 1, pp. 33 – 46, 2003. [Online]. Available: http://www.sciencedirect.com/science/article/pii/S0167639302000572

[39] D.-N. Jiang, L. Lu, H.-J. Zhang, J.-H. Tao, and L.-H. Cai, "Music type classification by spectral contrast feature," in *Proceedings. IEEE International Conference on Multimedia and Expo*, vol. 1, 2002, pp. 113–116 vol.1.

[40] C. Harte, M. Sandler, and M. Gasser, "Detecting harmonic change in musical audio," in *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, ser. AMCMM '06. New York, NY, USA: ACM, 2006, pp. 21–26. [Online]. Available: http://doi.acm.org/10.1145/1178723.1178727

[41] "Tonnetz". [Online]. Available: https://en.wikipedia.org/wiki/Tonnetz

[42] K. Prahallad. "Spectogram, Cepstrum and Mel-Frequnecy Analysis". [Online]. Available: http://www.speech.cs.cmu.edu/15-492/slides/03_mfcc.pdf

[43] "Melspectogram". [Online]. Available: http://www.fon.hum.uva.nl/praat/manual/MelSpectrogram.html

[44] H. Fayak. "Speech Processing for Machine Learning: Filter banks, Mel-Frequency Cepstral Coefficients (MFCCs) and What's In-Between". [Online]. Available: http://haythamfayek.com/2016/04/21/speech-processing-for-machine-learning.html

[45] "Short-time Fourier Transform". [Online]. Available: https://en.wikipedia.org/wiki/Short-time_Fourier_transform

[46] Jason Brownie. "Supervised and Unsupervised Machine Learning Algorithm ". [Online]. Available: https://machinelearningmastery.com/supervised-and-unsupervised-machine-learning-algorithms/

[47] ——. "Difference between Classification and Regression in machine learning". [Online]. Available: https://machinelearningmastery.com/classification-versus-regression-in-machine-learning/

[48] ——. "Logistic Regression for machine learning". [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[49] "Data Mining - Random Forest". [Online]. Available: https://gerardnico.com/data_mining/random_forest

[50] "K-NN classification". [Online]. Available: https://en.wikipedia.org/wiki/K-nearest_neighbors_algorithm

[51] "Multiclass Classification". [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[52] "Support Vector Machine". [Online]. Available: https://en.wikipedia.org/wiki/Support_vector_machine

[53] "Scikit-learn Support Vector Machines". [Online]. Available: http://scikit-learn.org/stable/modules/svm.html

[54] "SVM - SVC". [Online]. Available: http://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html#sklearn.svm.SVC

[55] S. Raschka. [Online]. Available: http://sebastianraschka.com/Articles/2014_about_feature_scaling.html#about-standardization

[56] [Online]. Available: https://en.wikipedia.org/wiki/Feature_scaling

[57] "Differences between the L1 Norm and L2 Norm". [Online]. Available: http://www.chioka.in/

[58] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, V. Dubourg, J. Vanderplas, A. Passos, D. Cournapeau, M. Brucher, M. Perrot, and E. Duchesnay, "Scikit-learn: Machine learning in python," *J. Mach. Learn. Res.*, vol. 12, pp. 2825–2830, Nov. 2011. [Online]. Available: http://dl.acm.org/citation.cfm?id=1953048.2078195

[59] [Online]. Available: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#/media/File:K-fold_cross_validation_EN.jpg

[60] "Hyperparameter Optimization". [Online]. Available: https://en.wikipedia.org/wiki/Hyperparameter_optimization

[61] [Online]. Available: https://en.wikipedia.org/wiki/Receiver_operating_characteristic