FEATURE SELECTION AND ANALYSIS FOR STANDARD MACHINE LEARNING CLASSIFICATION OF AUDIO BEEHIVE SAMPLES

by

Chelsi Gupta

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	Richard Inouye, Ph.D. Vice Provost for Graduate Studies

UTAH STATE UNIVERSITY
Logan, Utah

2019

ABSTRACT

Feature Selection and Analysis for Standard Machine Learning Classification of Audio Beehive Samples

by

Chelsi Gupta, Master of Science

Utah State University, 2019

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

Deep learning (DL) methods such as convolutional neural networks are effective classifiers but are considerably expensive as they require long training times and continuous access to GPUs. In contrast, standard machine learning (ML) methods take less time to train but require feature engineering to classify data. Feature engineering involves selecting features that reduce the complexity of a classifier and improve its performance. A central claim of this thesis is that with sufficient feature engineering, standard ML methods can perform on par with DL methods such as convolutional neural networks on the same datasets. Our method involves applying feature selection techniques to reduce the current feature space to optimal feature subsets. Selected subsets are used with four standard ML methods (logistic regression, k-nearest neighbors, support vector machines and random forests) to achieve the most accurate classification across three different audio datasets collected from BeePi, a multi-sensor electronic beehive monitoring system. The performance of the standard ML methods is compared with the performance of convolutional neural networks on the same datasets.

(60 pages)

PUBLIC ABSTRACT

Feature Selection and Analysis for Standard Machine Learning Classification of Audio Beehive Samples

Chelsi Gupta

The beekeepers need to inspect their hives regularly in order to protect them from various stressors. Manual inspection of hives require a lot of time and effort. Hence, many researchers have started using electronic beehive monitoring (EBM) systems to collect critical information from beehives, so as to alert the beekeepers of possible threats to the hive. EBM collects information by applying multiple sensors into the hive. The sensors collect information in the form of video, audio or temperature data from the hives.

This thesis involves the automatic classification of audio samples from a beehive into bee buzzing, cricket chirping and ambient noise, using machine learning models. The classification of samples in these three categories will help the beekeepers to determine the health of beehives by analyzing the sound patterns in a typical audio sample from beehive. Abnormalities in the classification pattern over a period of time can notify the beekeepers about potential risk to the hives such as attack by foreign bodies (Varroa mites or wing virus), climate changes and other stressors.

ACKNOWLEDGMENTS

I would like to take this opportunity to thank my major professor Dr. Vladimir Kulyukin, whose continuous guidance and support helped me to achieve success in this research. He always guided me towards a right direction whenever I struggled at various points in the research. His recommendations and suggestions were very essential for the accomplishment of this project.

I would also like to acknowledge my gratitude towards my graduate committee members Dr. Nicholas Flann and Dr. Haitao Wang, for providing appropriate support and feedback on various aspects of my research.

Finally, I would like to thank my beloved family and friends for always cheering me up in the difficult times.

Chelsi Gupta

CONTENTS

LIST OF TABLES

LIST OF FIGURES

ACRONYMS

MFCC	mel frequency cepstral coefficient
CCD	colony collapse disorder
EBM	electronic beehive monitoring
MANOVA	multivariate analysis of variance
ANOVA	analysis of variance
DFT	discrete fourier transform
PCA	principal component analysis
RFE	recursive feature elimination
SFS	sequential forward selection
RBA	relief based algorithms
KNN	k nearest neighbor
RF	random forest
LR	logisitc regression
SVM	support vector machine
OVR	one vs rest
DL	deep learning
ML	machine learning

CHAPTER 1

INTRODUCTION

1.1 Background

Honeybees also known as Apis millifera are some of the most diligent creatures on the planet. We owe many things to this amazing insect. Pollination is one of those important things as bees are responsible for pollinating about one-sixth of the flowering plant species worldwide and approximately 400 different agricultural plants [6]. The health of bees and beehives characterizes the health of our ecosystem. Since 2006, many colonies of honeybees have been disappearing in North America, Europe and some countries in Asia and Africa. This syndrome was named the colony collapse disorder (CCD).

CCD is the phenomenon that occurs when the majority of worker bees in a colony disappear and leave behind the queen, plenty of food and a few nurse bees to care for the remaining immature bees and the queen [7]. Millions of beehives have been lost to CCD since 2006. There are several possible reasons for the phenomenon to occur such as attack by pests (eg. Varroa mites), high use of pesticides, emerging diseases and viruses, poor nutrition or climatic changes in the environment. But no single reason attributes to the major cause of this issue, hence the beekeepers need to inspect the hives regularly so as to take precautionary measures to prevent CCD. Examining the beehives regularly inculcates a lot of time and effort from the beekeepers. They often have to travel to their hives at distant locations, which increases transportation costs. Also manually inspecting beehives interferes with the life cycle of honey bees. Therefore, many researchers and beekeepers have now started using electronic beehive monitoring (EBM) systems as a replacement for manual inspections. These systems collect critical information from the beehives without disturbing the the routines of the hive.

Much information can be extracted from the beehives, one such critical information is

the sound of bee buzzing. Abnormal changes in the patterns of bee buzzing would be a useful indicator for any possible threats to the hive. While collecting the bee buzzing sounds from a hive several other environmental sounds also come into play, which can indicate the presence of any stresses in the hive. Examining these sounds over a period of time would be helpful in determining the health of a hive.

In this study, the audio data collected from beehives using BeePi (a multi-sensor EBM), is categorized into three classes which are bee buzzing, cricket chirping and ambient noise. Classification of samples in these three categories will help the beekeepers to determine the health of beehives by analyzing the sound patterns in a typical audio sample from beehive. Classification is achieved using several machine learning algorithms. We have tried to improve the performance of these algorithms by determining the most important features that could sufficiently represent an audio sample.

1.2 Related Work

The current work is an extension of Amalthe's master's thesis [8], which extracts features (using Librosa) from an audio sample collected from BeePi and classifies them using machine learning techniques. The library used for feature extraction in this research (i.e. Librosa), faced some problems while installing on the current version of the Raspberry Pi system, so a new python library pyAudioAnalysis [1] was used to extract features out of the audio files in the current research. Also selecting the best amongst all the extracted features gave a further extension to the previous research. Several researchers have done audio processing and feature selection in their respective works. Some of them are discussed below.

In the research by Yaslan and Cataltepe [9],firstly 10 different classifiers were used to perform audio music genre classification on the Tzanetakis dataset. Then some dimensionality reduction techniques, such as forward and backward selection and principal component analysis (PCA), were used to get the important subset of features. The reduced features were then used to obtain the test genre classification accuracy. Finally, some classifier combinations were used to improve the precision of classifiers. The research observed that the

best subset of features depend on the classifier being used.

One more project that deals with audio classification by making use of support vector machines and wavelets to categorize audio data, was developed by Lin et al. [10]. This research proposes a method in which the acoustical features (such as subband power and pitch information) of an audio are first extracted using wavelet information. The method then applies a bottom-up SVM over these acoustic features and some additional parameters such as frequency cepstral coefficients. They have used the Muscle Fish public audio database to evaluate the performance of their proposed method.

Kiktova et al. [11], performed some front end feature extraction and selection to provide audio surveillance in urban environments. Some spectral and temporal features, such as MFCC, MPEG-7, MELSPEC and FBANK were extracted from an acoustic signal. Mutual information, which is a filter based selection algorithm, was used to select the most important features. Performance of the chosen features was then evaluated using the Hidden Markov Model based classifier.

Contreras et al. [12], applied chi squared filter to reduce the 62 features of a complex environmental sound into 15 features. These features were obtained using statistical and frequency domain analysis. The reduced feature vectors were then used with SVMs to perform a 10 class environmental sound classification.

An approach similar to the one used in this research is also applied by Patel and Patwardhan [13]. The research also uses some feature selection techniques such as information gain, CFS and SVM using the weka tool [14] to rank the features according to their relevance. The classification accuracy of the SVM classifier is checked by removing low ranked features iteratively.

1.3 Current Work

Features are an important part for any classification problem. Extracting the most important features from a sample can have a significant impact on the performance of our classifier. In this study, firstly an audio sample is reduced to a set of 34 features which represent the full sample. Secondly, some feature selection techniques are applied, which

reduces the current set of 34 features into a small subset of just 13 features. Finally, these features were used with different classifiers to improve their performance.

The thesis is organized as follows. In chapter 2, the materials and methods used for data collection are described. Chapter 3 gives an overview of the datasets that have been classified in this research. Chapters 4 and 5 discuss the audio feature extraction and selection techniques used to improve the current classification techniques. Chapter 6 describes the machine learning models used. Chapter 7 gives a detailed description of the experimental setup and the results obtained from those experiments. Finally, chapter 8 provides a conclusion and discusses the future scope for this research.

CHAPTER 2
MATERIALS AND METHODS

2.1 BeePi

The audio data for this research was captured by BeePi, which is a solar powered, multi-sensor electronic beehive monitoring system, designed to determine the condition of a bee colony. It extracts video, audio and temperature information from a beehive without intrusive hive examinations. A typical BeePi system consists of a Raspberry Pi 3 computer, a camera, a clock, a temperature sensor and four microphones, all placed in a Langstorth super [15]. The microphones are either placed above the beehive's landing pad or are embedded in the beehive walls. The results from BeePi can be reproduced with minimum cost and time, as most of the hardware is from off the shelf components. The BeePi kickstarter [16] is an interesting source to get further details about the design and descriptions of BeePi.

Five deployments have been made for the BeePi monitors across different seasons and locations [17]. First deployment was made in September 2014 at Logan, UT, USA and ran for two weeks. Second deployment was between December 2014 and January 2015 at Garland, UT, USA and operated successfully for nine out of the fourteen days of deployment. The third deployment was between April 2016 and November 2016 and consisted of four BeePi units placed into four beehives at two small apiaries in North Logan, UT, USA. Fourth deployment also consisted of four BeePi units placed into four beehives at two small apiaries in both Logan and North Logan, UT, USA between April 2017 and September 2017. Fifth deployment was between May 2018 and July 2018 at Logan, UT, USA with four BeePi monitors placed in four new beehives.

2.2 Audio Data

A BeePi monitor saves a 30-second audio wav file every 15 minutes on a USB storage device connected to the Raspberry Pi [17]. A python script was used to chunk the 30-second audio sample into 2-second wav files with an overlap of 1-second. This resulted in 28 2-second wav files for each 30-second audio sample. The script automatically labels and indexes the samples and finally puts them in their respective category folders.

The ground truth for classification was obtained by first listening to each 2-second audio sample and then manually placing it in one of the three non-overlapping categories: bee buzzing (Bee), cricket chirping (Cricket) and ambient noise (Noise) [17]. Ambient noise here refers to the random microphone clicks, human conversations, breeze, rain and relative silence (i.e. sounds which cannot be detected by humans).

The 2-second wav files were then read and stored into a numpy array using a routine of the python library pyAudioAnalyis [1], so as to facilitate feature extraction and selection from these wav files (see chapter 3 for further details on this).

2.3 Datasets

In this thesis, the machine learning models have been trained, tested and validated across 3 different datasets.

- BUZZ1: contains a total of 10,260 2-second audio samples.
- BUZZ2: contains a total of 12,914 2-second audio samples.
- BUZZ3: contains a total of 15,254 2-second audio samples.

The datasets have been divided into three categories i.e. the train data, the test data and the validation data which were separated from each other by beehive and location. All datasets have approximately the same percentage distributions across all three categories. The sample distributions of the datasets are given in tables 2.1, 2.2 and 2.3. The tables show the number of bee buzzing, cricket chirping and ambient noise samples in the training, testing and validation data of these three datasets.

Table 2.1: BUZZ1 sample distribution of 2-second audio samples.

	Bee	**Cricket**	**Noise**	**Total**
Train	2215	2474	2234	6923
Test	785	526	876	2187
Validate	300	500	350	1150
Total	3300	3500	3460	10260

Table 2.2: BUZZ2 sample distribution of 2-second audio samples.

	Bee	**Cricket**	**Noise**	**Total**
Train	2402	3000	2180	7582
Test	898	500	934	2332
Validate	1000	1000	1000	3000
Total	4300	4500	4114	12914

Table 2.3: BUZZ3 sample distribution of 2-second audio samples.

	Bee	**Cricket**	**Noise**	**Total**
Train	2880	3600	2520	9000
Test	1071	577	1098	2746
Validate	1170	1169	1169	3508
Total	5121	5346	4787	15254

The training, testing and validation data in all the three datasets are completely separated and different as they come from different hives. We have further provided evidence to the claim of training, testing and validation data being completely different from each other, by applying Multivariate Analysis of Variance (MANOVA) on all the three datasets i.e. BUZZ1, BUZZ2 and BUZZ3.

MANOVA is an extension of the univariate analysis of variance (ANOVA). ANOVA uses hypothesis testing to account for the difference between two or more groups based on the mean of variance of `only one` dependent variable. Whereas, MANOVA is used when we have to compare data from two or more groups based on the means of variances of `more than one` dependent variable. MANOVA involves performing several statistical tests on each dependent variable separately.

We have used all the 34 features that pyAudioAnalysis extracts out of an audio sample, as dependent variables for performing MANOVA on the three datasets. The feature extraction is described in detail in chapter 3. The independent variable was the class attribute containing bee, cricket and noise as the target classes. The results of performing MANOVA on BUZZ1, BUZZ2 and BUZZ3 are shown in tables 2.4, 2.5 and 2.6 respectively. The tables show Pillai Coefficient, F-Factor and P-Value as a result of applying MANOVA to compare Train and Test data, Train and Validate data and Test and Validate data of all the three datasets BUZZ1, BUZZ2 and BUZZ3.

Table 2.4: BUZZ1 MANOVA Results.

	Pillai Coefficient	**F-Factor**	**P-Value**
Train-Test	1.8208	2711.7	2.2e-16
Train-Validate	1.7871	1984.4	2.2e-16
Test-Validate	1.7802	786.49	2.2e-16

Table 2.5: BUZZ2 MANOVA Results.

	Pillai Coefficient	**F-Factor**	**P-Value**
Train-Test	1.808	2735.4	2.2e-16
Train-Validate	1.7338	2020.8	2.2e-16
Test-Validate	1.7827	1278	2.2e-16

Table 2.6: BUZZ3 MANOVA Results.

	Pillai Coefficient	**F-Factor**	**P-Value**
Train-Test	1.5738	1271.7	2.2e-16
Train-Validate	1.5547	1280.6	2.2e-16
Test-Validate	1.7478	1267.4	2.2e-16

Our null hypothesis is that all our groups train, test and validate of the datasets are similar i.e. they have the same population means. Since the p-value for comparison between all the groups is 2.2e-16, which is significantly lower than 0.05, so we can reject the null hypothesis. This proves that all our groups have different population means and are significantly different from each other.

CHAPTER 3

FEATURE EXTRACTION

Features play a key role while addressing any machine learning problem. Feature extraction is the first step in the the preprocessing phase of our dataset. Feature extraction is essential for removing irrelevant and redundant values from raw data, so that the machine learning model can perform well. In this thesis, the time, frequency and cepstral domain features of an audio signal have been extracted.

Feature extraction is done using pyAudioAnalysis. pyAudioAnalysis is a Python library that can perform a wide range of audio analysis tasks including extraction of audio features. All the features that pyAudioAnalysis extracts are listed in Table 3.1. The temporal features (features 1-3) are extracted directly from the raw signal samples (the .wav audio files in our case). The spectral features (features 4-34, except the MFCCs) are established on the magnitude of the Discrete Fourier Transform (DFT). Finally, the cepstral features (i.e. the MFCCs) are a result of applying the Inverse DFT on the logarithmic spectrum [1]. In total, 34 features are extracted from an audio signal.

3.1 Short and Mid-term Analysis

The previously mentioned features can be extracted in two ways. First, extraction of features on a short-term basis, where the signal is first partitioned into short-term windows (frames) and then all 34 features are calculated for each frame [1]. The outcome is a sequence of short-term feature vectors with 34 elements each. Broadly acknowledged short-term window sizes are 20 to 100 milliseconds (ms). The framing during short-term feature extraction in pyAudioAnalysis can be overlapping (frame step is shorter than the frame length) or non-overlapping (frame step and frame length are equal). In this thesis, some window sizes between 20 to 100 ms have been tried, combining them with both overlapping and non-overlapping framing procedures, so as to determine the best performing

Table 3.1: List of 34 short term features extracted by pyAudioAnalysis [1].

Feature ID	**Feature Name**	**Description**
0	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
1	Energy	The sum of squares of the signal values, normalized by the respective frame length.
2	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
3	Spectral Centroid	The center of gravity of the spectrum.
4	Spectral Spread	The second central moment of the spectrum.
5	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
6	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
7	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.
8-20	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
21-32	Chroma Vector	A 12-element representation of the spectral energy where the bins represent the 12 equal-tempered pitch classes of western-type music (semitone spacing).
33	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

combination and the best performance was achieved using a window of size 20ms and a step of size 20ms.

Second, extracting features on a mid-term basis, where the signal is first partitioned into short-term windows (segments) and then the short-term processing stage is carried out for every segment. Typically, the mid-term segment sizes can range from 1 to 10 seconds. Therefore mid-term feature extraction is used in cases of long recordings (eg. music tracks), where a long-term averaging of the mid-term features is required to represent the whole

signal [1]. Since our datasets consist of audio samples of length 2 seconds, so we have just used the short-term feature extraction here. The process of feature extraction using pyAudioAnalysis involves 2 steps:

1. The 2-second .wav file is read and stored into a numpy array using the `readAudioFile` function from the `audioBasicIO` module of pyAudioAnalysis.

2. The features of this numpy array are extracted into another numpy array using the `stFeatureExtraction` function from the `audioFeatureExtraction` module of pyAudioAnalysis by setting window size as 20ms and step size as 20ms.

3.2 Principal Component Analysis

Principal Component Analysis (PCA) is a feature extraction technique of dimensionality reduction, used for multivariate analysis of dataset that contains many possibly correlated variables. PCA transformation involves converting the highly correlated feature space into a small set of uncorrelated variables called `principal components`. Principal components can be seen as linear combinations of all variables which aim at retaining the maximum information from the original feature space. The first principal component explains the maximum variance of the dataset and then each successor explains the remaining variance regressively. The resulting PCs are orthogonal to one another which means they are statistically independent of one another [18]. Also there is no assurity that these PCs can be interpreted into actual physical features. PCA does not assume a dependent variable to be specified while fitting and uses a correlation matrix to determine how the original variables are interrelated. This technique is mostly used when we want to retain the slightest possible contribution that a feature might have in predicting the class variable rather than just removing the entire feature, as is done in feature elimination technique of dimensionality reduction.

PCA works by calculating a matrix that defines how our variables relate to each other and this matrix is then broken down into two components: direction and magnitude [18]. Directions are a fusion of the original variables and magnitude defines the significance of

each direction. PCA projects our data into a smaller space by dropping the directions that are least important based on their magnitude. The figure 3.1 shows a sample 2 dimensional data in the x-y plane.

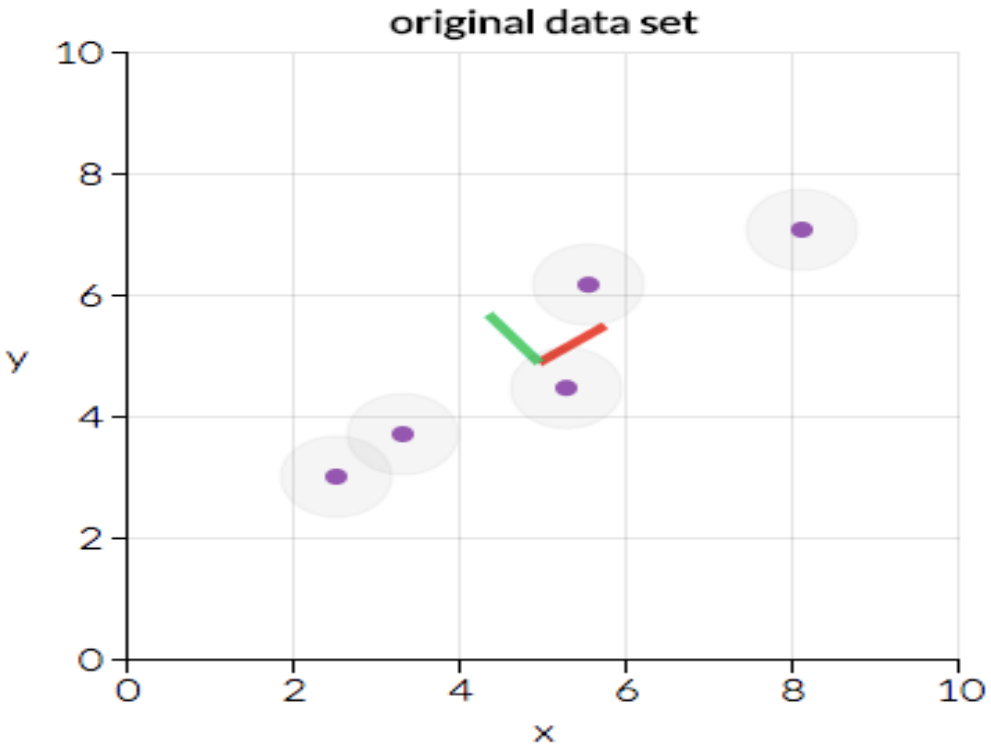

Fig. 3.1: Our original data in the xy-plane [2].

In the Figure 3.1, the purple dots depict the observations in the sample data, and the `red` and `green` lines are the two main directions in this data. The red direction appears to be the most important one as it coincides with the line of best fit to this data. PCA transforms the original data to align with the main directions of this data. The transformation of sample data is shown in figure 3.2.

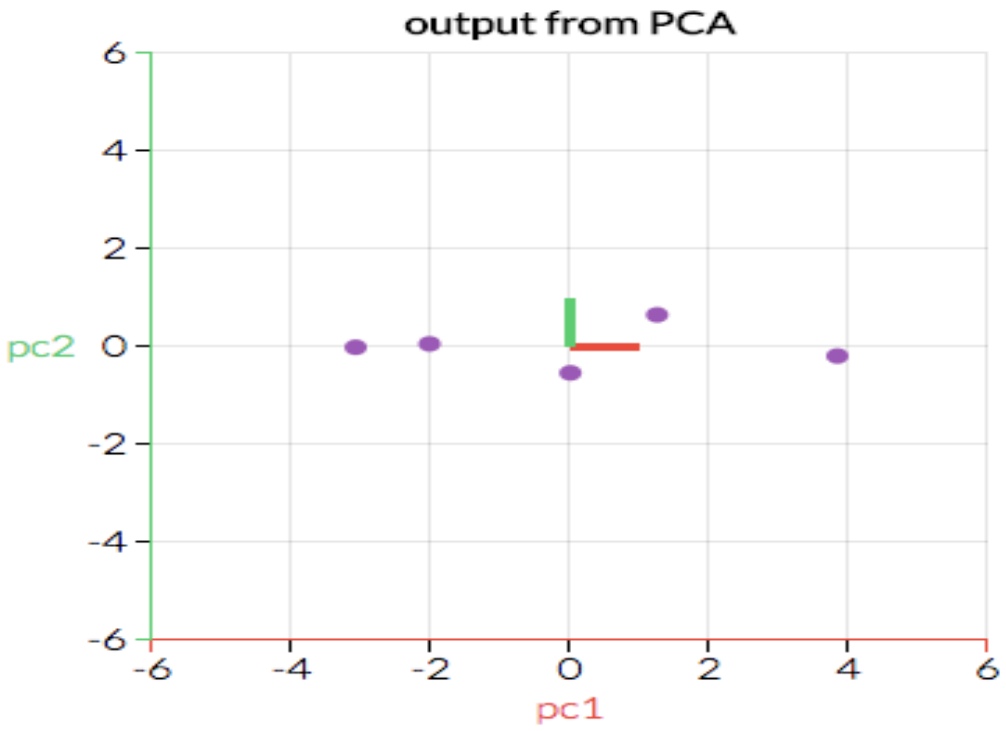

Fig. 3.2: Our original data transformed by PCA [2].

In the Figure 3.2 we can drop the pc2 component as most of the variance in data is explained by pc1. Thus, applying PCA helped in reducing the dimension of sample data.

CHAPTER 4

FEATURE SELECTION

After successfully extracting the features of an audio sample the next step is to analyze the most important features that would contribute most towards classification and also to remove those features which tend to increase noise in our models. This process is called feature selection. Feature selection automatically removes or identifies the redundant, irrelevant or highly correlated variables from the high dimensional dataset at hand without loosing much information. This helps in reducing overfitting, decreases the training time and enhances the performance in some cases. Most of the feature selection techniques scan the entire feature set for the most optimal subset of features. This scan algorithm uses an evaluation metric that ranks all the features and provides the optimal feature subset. Based on the evaluation metric used in the scan algorithm the feature selection techniques can be divided into three categories: Wrapper methods, Filter methods and embedded methods.

4.1 Wrapper Methods

Wrapper methods aim at determining the most `functional` subset of features by training a machine learning model on those subsets. This training process is iterative and after each iteration of training we make a decision to add or remove some features from the feature subset used in previous iteration based on the model performance in that iteration. The iteration stops when the algorithm produces a feature subset with desired number of features. The procedure for selecting features by wrapper methods is shown in Fig. 4.1. The feature subsets are scored according to their cross- validation performance with the machine learning model being trained. Wrapper methods use an extensive greedy search, and can therefore find the best and most useful feature subset for the model being trained. But the features selected from wrappers are best for just this model, and may not perform well if used with models other than the one currently being trained. These features are

more prone to overfitting as they model the current data too closely. The wrapper method is also computationally more expensive as it involves many iterations of model training.

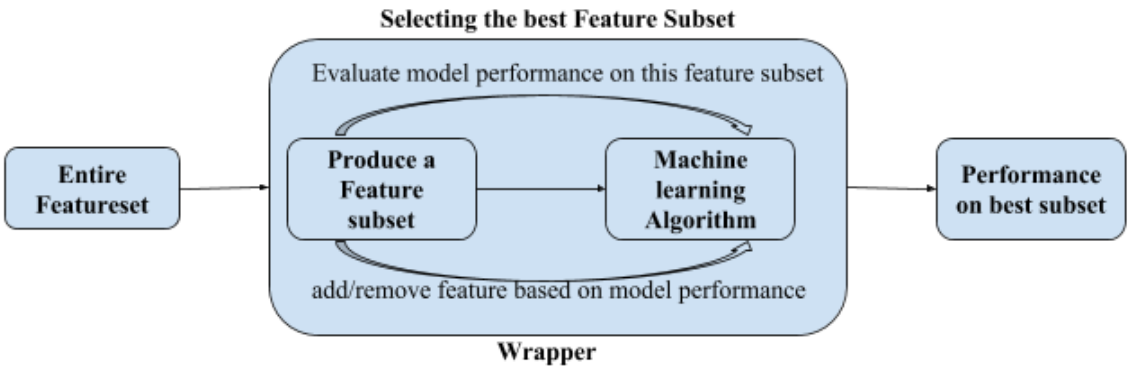

Fig. 4.1: Wrapper Method for Feature Selection [3]

The wrapper methods used in this investigation are Recursive Feature Elimination and Sequential Feature Selection.

4.1.1 Recursive Feature Elimination

This feature selection technique works by eliminating the least useful feature recursively. Usefulness is determined by the scores assigned to these features from the machine learning models being trained recursively. The process involves fitting a machine learning model with the initial set of features and calculating the classification performance at first. This model provides the importance of each feature either through a `coeff_` attribute or through `feature_importances_` attribute. The least important feature is then removed from the current feature set and the model is trained again with the remaining set of features. This process is repeated recursively with the remaining features until we reach a subset with the desired number of features [19].

It should be noted that recursive feature elimination can only be applied to models that expose either a `coeff_` or `feature_importances_` attribute and hence machine learning algorithms such as K-nearest neighbor and Support Vector Machines (other than linear kernels) could not be used here.

4.1.2 Sequential Feature Selection

Sequential feature selection involves a greedy search for determining the best subset of features and is an alternative to the computationally intensive exhaustive search which scans through all possible subsets of features. These techniques work towards determining that subset of features which is most useful for the model classification. SFAs reduce the dimensionality of the feature space by adding or removing one feature at a time based on the classifier performance [20]. The algorithm stops when the desired number of features is reached. There are two types of SFAs:

- Sequential Forward Selection: They start with an empty set of features. Train n models with individual features at first(n is the dimension of feature set). Then calculate their performances and selects the feature that contributes most towards classification performance. This feature is added to the initial empty set and then again n-1 models are trained with each feature and the previously selected feature. This process is repeated until a subset with a predetermined number of features is reached.

- Sequential Backward Selection: This process trains a model on the entire feature set first and then removes one feature at a time. The feature being removed is the one that contributes the most to classifier performance on its removal. This process is repeated until a subset with a predetermined number of features is reached.

In this thesis, Sequential Forward Selection technique is used.

4.2 Filter Methods

Filter methods aim at determining the most `relevant` features by analyzing the relation between features and the target class. Filter methods provide a ranking for all the features rather than selecting the best feature subset. This ranking is based on the scores obtained by each feature/target combination in some statistical test or inter/intra class distances between instances of dataset and can be used to select the top scoring features which can be used with any classifier to help improve its performance and efficiency. Filter methods

do not guarantee the best subset of features, but the results obtained from these methods are more generalized as they are not dependent on any predictive model. Performance of a model on the features selected from these methods may sometimes be less than the performance on wrapper selected subsets. These methods are less computationally intensive and the features selected are more robust towards overfitting as no training of models is required here. The filter methods are sometimes used for preprocessing the data before applying a wrapper method, so that the wrappers can be used for large datasets. The procedure for selecting features by filter methods is clearly shown in Fig. 4.2.

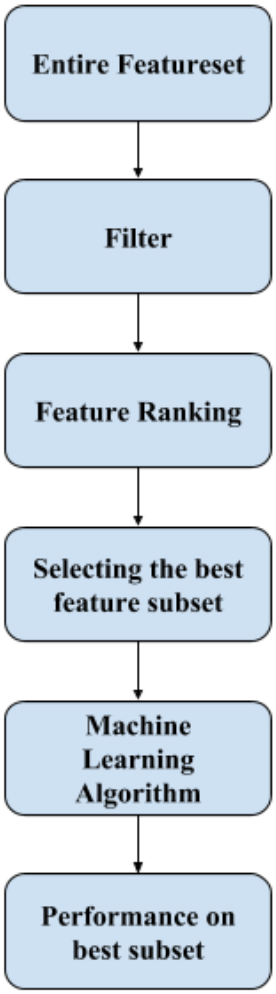

Fig. 4.2: Filter Method for Feature Selection

The filter methods used in this research are Univariate Feature Selection and Relief Based Feature Selection.

4.2.1 Univariate Feature Selection

Univariate feature selection helps in determining the strength of the relationship between each individual feature and the target variable, using some statistical tests such as chi-square, F-test, Mutual Information etc. Features are ranked according to the strength of their relation with the outcome. All features other than a predetermined number of highest scorers, are removed from the current feature space. The remaining features are then used to train, test and validate the machine learning models. Univariate feature selection is used as a preprocessor before applying an estimator model to the dataset [19]. F-test and Mutual Information are the two univariate statistical tests used in our work.

- F-test: It uses hypothesis testing to check if feature and target variable are significantly different. Correlation between feature and target is computed which is then converted to F-score to obtain the p-value. The features with high F-scores are selected for modelling.

- Mutual Information: This test measures the dependence of target variable on the feature variable. If the target is independent of the feature then mutual information value is 0, if target is fully dependent on the feature then the value is 1, otherwise the value is between 0 and 1. The features with high mutual information values are selected for modelling.

4.2.2 Relief Based Feature Selection

Relief based algorithms are a family of algorithms derived from Kira and Rendell's Relief Algorithm [21] which assigns weights/scores to each feature depending on the difference between feature values of `nearest neighbor instances`. The neighboring instances are determined using euclidean distances. If we see a difference in feature values between two

neighboring instances of the same class (i.e. a `hit`), then the feature score decreases. In contrast, if a feature value difference is observed between two neighboring instances belonging to different classes (i.e. a `miss`), then the feature score increases. Original relief algorithms were designed to capture only binary classification problems with discrete features and had no mechanism to handle missing data [4]. Although further extensions of this algorithm are adapted to address these issues and have proved to increase the performance of core Relief algorithm even with noisy datasets that have large feature spaces. The core Relief algorithm is described in Algorithm 1.

Algorithm 1: Relief Algorithm [4].

Input : Feature vector (of length **a**) and target value for each instance in the dataset of size **n**.

m - number of random instances out of n, used to update the feature weight W.

Output: Vector W consisting of feature weights.

Initialization: p length weight vector (W) of zeroes.

for *i = 1 to m* **do**

Select a random target instance R_i.

Find its nearest hit **H** and miss **M** instances through euclidean distance.

for *A = 1 to a* **do**

$$W[A] = W[A] - \frac{(A - H_A)^2}{m} + \frac{(A - M_A)^2}{m} \tag{4.1}$$

end

end

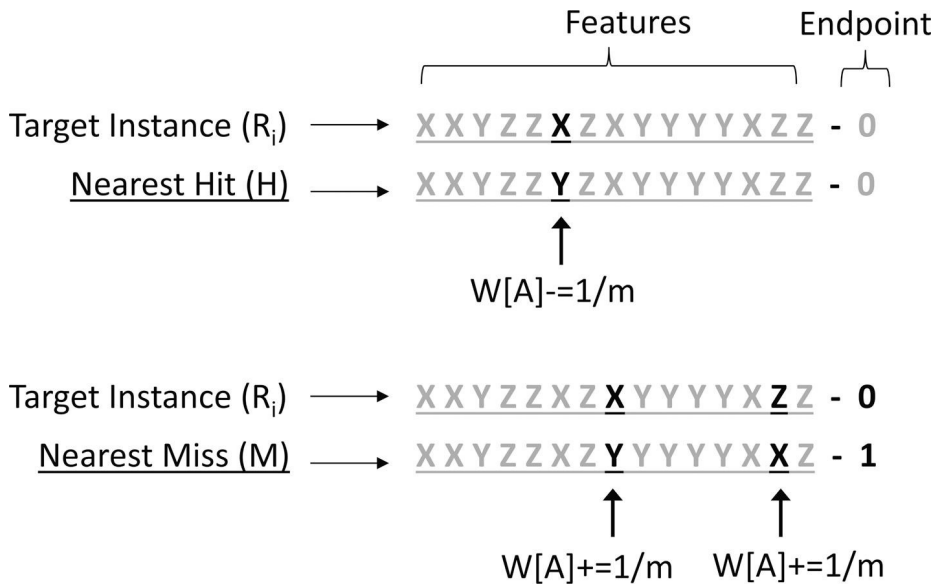

Fig. 4.3: Relief updating W[A] for a given target instance when it is compared to its nearest miss and hit [4].

In this example, features are discrete with possible values of X, Y, or Z, and endpoint is binary with a value of 0 or 1. Notice that when the value of a feature is different, the corresponding feature weight increases by 1/m for the nearest miss, and reduces by 1/m for the nearest hit.

We have used the ReliefF algorithm from the family of RBAs in this thesis. The ReliefF algorithm updates the original Relief algorithm in the following ways:

1. It uses the Manhattan distance rather than Euclidean distance to find the nearest neighbor instances.

2. It finds the absolute difference between A and H_A, and A and M_A rather than the square of those differences.

3. For moderately large datasets, the algorithm would repeat itself for each instance rather than running for m random number of instances.

4. It finds K-nearest neighbors and averages their contribution to the weights of each feature. K can be adjusted according to the specific problem.

5. It extends itself to multinomial classification by finding K nearest misses from each class different from the target class.

6. It can also handle missing data by assigning it with the relative frequency from the dataset.

4.3 Embedded Methods

Embedded methods are a hybrid of filters and wrappers and differ mainly by the learning algorithms from these methods. In contrast to filters and wrappers, these methods use those machine learning algorithms that involve feature selection as a part of their model building process (i.e., feature selection is embedded with the learning algorithm). Embedded methods are very similar to wrapper methods and hence aim at determining the most `functional` subset of features. The difference from wrappers is that here we combine the feature selection step with the model training process. These methods take advantage of the intrinsic structure of the machine learning model and select a feature at each step of the model construction process. The method stops when either all features are covered or the performance of the model does not tend to increase further. The procedure for selecting features by embedded methods is shown in Fig. 4.4. These methods are midway between filters and wrappers, in terms of computational complexity, with wrappers being most expensive. These methods are less prone to overfitting than wrappers, although the subset selected from these methods is more specific for the model being trained as in the case of wrappers.

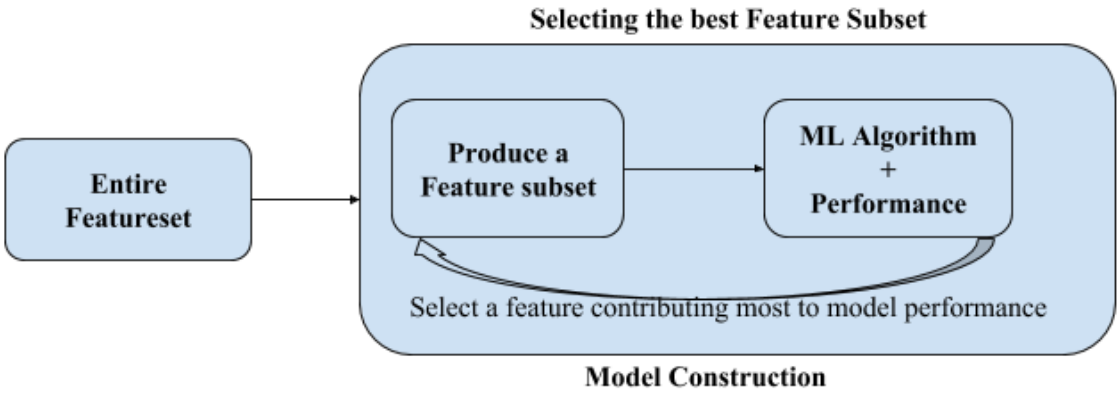

Fig. 4.4: Embedded Method for Feature Selection

The embedded method used in this research is Random Forest which is a decision tree based algorithm.

4.3.1 Random Forest Feature Selection

Random forests are a collection of decision trees constructed by selecting random samples from the dataset. Each tree votes for a target class and the classification which has the majority of votes is chosen amongst all trees in the forest [22]. Decision tree construction process involves splitting the existing dataset into two halves recursively, based on the feature which contributes the most towards increasing the purity of a node. Purity of a node is measured by either Gini-index or Entropy value (we have used Gini-index here). Entropy and Gini-index can be defined as:

$$Entropy = -\sum_{j} P_j log_2 P_j \tag{4.2}$$

$$Gini = 1 - \sum_{j} P_j^2 \tag{4.3}$$

where P_j is the probability of class j in the node. The value for entropy and gini is 0 if all samples of a node belong to the same class, and it is maximal if we have a uniform class distribution [23].

Features which contribute more towards decreasing the impurity of a node are more closer to the root node in the finally constructed tree. Random forests find the mean of impurity decrease from each feature across all its trees to determine the final importance of the feature. The features which are selected at the start of the tree construction are placed at the top in feature ranking and can be selected for model testing and validation.

CHAPTER 5

MACHINE LEARNING MODELS

Supervised Learning is a ML technique that involves predicting the outcome of new data based on the inference from the input data. It learns a function, which helps in mapping the input object (mostly in the form of vectors) to the desired output. Using this mapping function, a mathematical model is constructed on the labeled training data instances provided to the algorithm. The algorithm then uses this model to predict outcomes for future data instances. Supervised Learning uses classification and regression techniques to build these models. Classification algorithms are used when the target variable is discrete or categorical (i.e. restricted to a limited set of values), whereas regression techniques are used when the target variable is continuous and can have any value within a range.

Building a machine learning model requires, properly labelled training data (as sample data) and a machine learning algorithm suitable for our target attribute. This algorithm is then run on the sample data to capture the hidden patterns of the data. The algorithm can be controlled by tuning its parameters, in order to achieve better performance. After the algorithm finishes, the model is finally built. The new data that comes for prediction, is passed to this model to predict the outcome.

We have used four types of machine learning classification models in this research: K-Nearest Neighbor [24], Random Forest [25], Logistic Regression [26] and Support Vector Machines [27].

5.1 K-Nearest Neighbor

K-nearest neighbor (KNN) is a `non-parametric`, `lazy learning` algorithm. Non-parametric means that it makes no assumptions about the distribution of the training data which means that the models are constructed on the underlying data [5]. Lazy learning signifies that the algorithm does not generalize the training data and keeps all or most of

the training data to predict the outcome of new instances.

While dealing with classification problems, KNN models help in predicting the target class of the incoming test object. The major premise behind KNN is that "similar things appear close to each other". Therefore, an object is classified by a majority vote of its neighbors and is assigned to the class most common amongst its k-nearest neighbors [5]. This k is a positive integer parameter passed to the KNN algorithm and can be tuned in order to achieve better prediction accuracy.

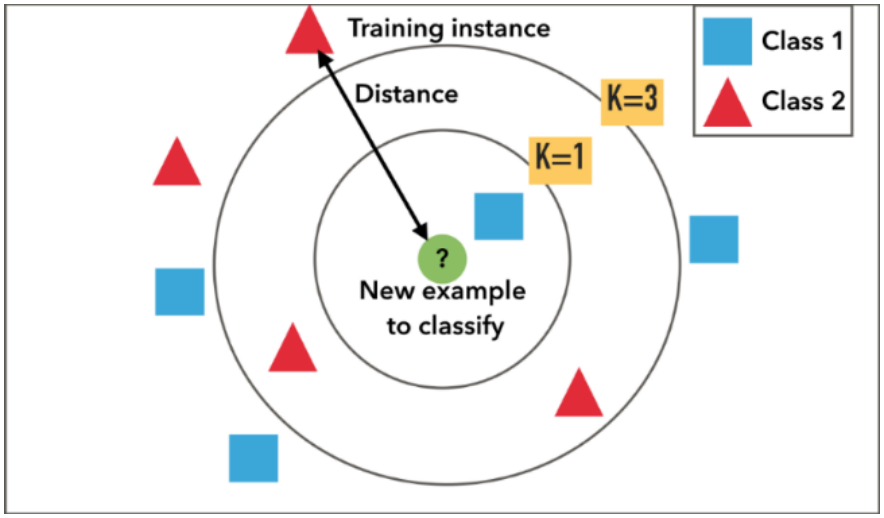

Fig. 5.1: Example of KNN classification [5].

The figure 5.1 explains the notion of KNN classification models. The objects are classified according to their distance to the nearest neighbors. In this example, the green circle can be classified to the blue square class or to the red triangle class. If k = 1, then it is assigned to the blue class as there is a square near to the green object in the inner solid circle. If k=3, it is assigned to the red class (2 triangles and 1 square).

5.2 Random Forest

Random forests algorithms are a type ensemble learning algorithms, which create a number of decision trees from the training dataset to predict the outcomes of test data. When the training data (consisting of target and feature values) is given as input to the

decision tree, it produces a set of rules. These rules are then used to predict the class of the new instances in test data. Decision trees follow a top down approach in which the root node recursively applies a binary split on most predictive feature in order to create new nodes. The recursion continues until the resulting leaf nodes are in their purest form. The impurity of a node in decision trees is calculated using `gini index` or `entropy`. Decision trees predict a target class for each leaf node.

Random forests create a forest of decision tree models, where each tree is created over random sample of data chosen from the training set. This is the reason random forests are called `bagging classifiers`. In contrast to decision trees which take all features into consideration while building a model, these algorithms choose a random sample of features from the feature space to create each individual decision tree. Each tree votes for a specific target class and the test instance is then assigned to the class with a majority vote. Each tree can be assigned a weight according to their prediction accuracy. Trees with high error values are weighted low and finally a mode of the prediction class from trees with higher weights is predicted as the final target class.

5.3 Logistic Regression

Logistic Regression model uses a logistic or sigmoid function to convert its output into a probability value which can then be mapped to two or more target classes [28]. Logistic regression is generally used for binary classification problems but when the outcomes can belong to three or more discrete classes then we use `multinomial logistic regression`. Multinomial logistic regression calculates the probabilities for each target class in the training process and these probabilities are then used to predict the outcome of test instance. The general equation for logistic regression is:

$$y = \frac{e^{b_0 + \sum b_i x_i}}{1 + e^{b_0 + \sum b_i x_i}} \tag{5.1}$$

Here y is the predicted output. b_0 is a constant or bias and b_i is a coefficient associated with each x_i, which is a feature from the feature set X [29].

5.4 Support Vector Machines

Support vector machine (SVM) is used when we have non-linearly separable data. The main premise of SVMs is to find a hyperplane in a an n-dimensional feature space which can be used to categorize the new instances distinctly [30]. In a 2 dimensional space this hyperplane is a line which separates the classes accurately. This line is at an optimum distance from the `support vectors`. Support vectors are the points closest to this separating line. SVMs find many lines and choose the one that is at maximum distance from the support vectors. While dealing with non-linear classification problems, SVMs use polynomial kernels which map the data points into a different plane so as to separate the classes by a clear gap that is as wide as possible [31].

In this thesis SVM classifier is also applied using the One-vs-the-rest (OvR) multi-class/multilabel strategy. OVR strategy builds a binary classifier for each class. This classifier considers the currently chosen class as positive and all the rest classes as negative which means that the class is fitted against the rest of the classes. The multiclass classification problem is thus solved by multiple binary classifiers. Thus we can easily gain insights about each class by accurately inspecting their corresponding classifiers [32].

CHAPTER 6

EXPERIMENTS AND RESULTS

The experimental procedure starts with extracting the features of an audio sample at first, then selecting the best amongst those features and finally training our machine learning models only with the best features. Feature selection was applied only on the BUZZ2 dataset, as it is our benchmark dataset. Similar features were then extracted from both BUZZ1 and BUZZ3 dataset. The best performing models on test data of all datasets were saved using python pickle and then used to predict on the out of sample validation data of each dataset respectively.

6.1 Chosen Models and Evaluation Metrics

Four machine learning models from the sklearn library [33] were used to address the classification problem for all datasets. The following list describes each model along with its parameter values used as a standard throughout our experiments.

- **K Nearest Neighbor**

  Implemented using KNeighborsClassifier class from the neighbors module of sklearn. The number of neighbors was set to 8. The class also takes a parameter `p` which determines the distance metric to be used to find the neighbors. p is set to 3 which means it uses the minkowski distance. For KNN we have scaled our features using the scale routine of sklearn preprocessing module, as this estimator requires the variances to be adjusted to a scale centered around 0.

- **Random Forest**

  Implemented using RandomForestClassifier class from ensemble module of sklearn, using 100 trees. The `max\_features` parameter is set to `log2` which means log base 2 of the total number of features is considered while looking for the best split at each tree.

- **Logistic Regression**

  Implemented using LogisticRegression class from the linear_model of sklearn. The `penalty` parameter of the class, which specifies the rule used for penalization, is set to `l1`. Also the tolerance or stopping criteria pf the algorithm is set to 0.1.

- **SVM using OneVsRest strategy**

  Support vector machines have been implemented by using the SVC class of svm module from sklearn, as an estimator of the OneVsRestClassifier class from multiclass module. Polynomial kernel of degree 3 is used with the `probability` parameter set to True (which enables the probability estimates).

The performance of above models was analyzed using two evaluation metrics.

- **Classification Accuracy:** Accuracy is the number of labels our classifier identifies correctly. It can be defined as:

$$Accuracy = \left(\frac{N}{P}\right) * 100, \tag{6.1}$$

  where N is the number of correct predictions and P is the total number of predictions made by the classifier.

  It is kind of a superficial estimator and does not give much insights about the underlying distribution of the predictions and hence does not identify False Positives and False Negatives.

- **Confusion Matrix:** Also known as error matrix uses a tabular format to describe the performance of our classifier on out of sample data for which true values are known. It provides a clear picture of the errors our classifier makes in prediction. For binary classification with target labels 0 and 1, the confusion matrix is:

Table 6.1: Confusion Matrix.

	Predicted Class 1	**Predicted Class 0**
Actual Class 1	True Positive	False Negative
Actual Class 0	False Positive	True Negative

6.2 PCA over Extracted Features

Principal component analysis is applied using the PCA class from the decomposition module of scikit-learn library [34], over all the 34 features extracted by pyAudioAnalysis library (as shown in 3.1). PCA extracts min(n,F) principal components from a dataset, where n is the size of the dataset and F is the size of feature set. Since in our case number of features (i.e. 34) is significantly lower than the size of datasets, therefore only 34 principal components are obtained from PCA. The figure 6.1 shows a plot of the variance ratios of all 34 principal components. PCA is applied over the features of BUZZ2 dataset.

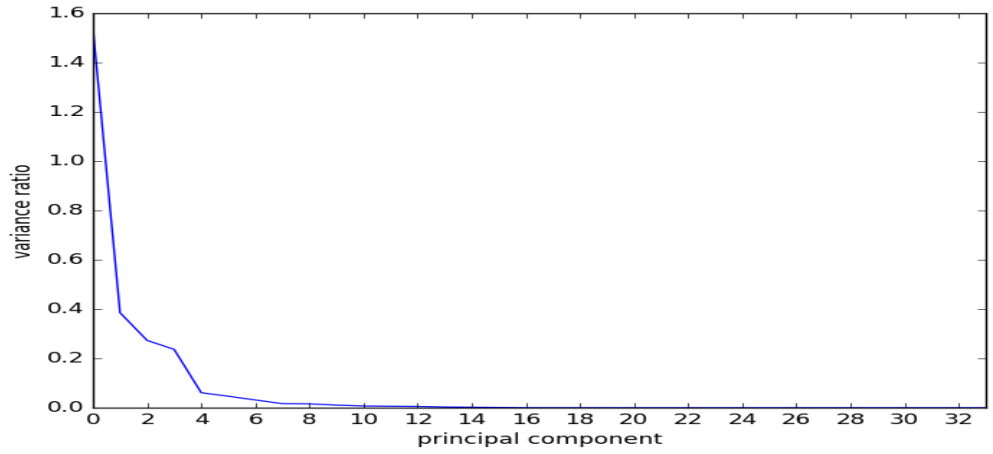

Fig. 6.1: Principal component variance plot

Analyzing the plot above, we decided to run our models on the first 7 and 8 principal components only, as around 99% of the variance of dataset is explained by these components.

The table 6.2 shows the testing and validation accuracies of applying our chosen ML models with 7 and 8 principal components.

Table 6.2: PCA Results on BUZZ2 dataset.

Model	**# principal components**	**Testing Accuracy**	**Validation Accuracy**
KNN	**7**	97.85%	82.1%
	8	97.38%	82.63%
Random Forest	**7**	96.95%	60.46%
	8	97.12%	64.1%
Logistic Regression	**7**	83.70%	82.86%
	8	85.07%	85.39%
SVM using OneVsRest	**7**	92.19%	75.6%
	8	89.49%	78.06%

In the Table 6.2, we can see that the best performance on the validation data of BUZZ2 was achieved by applying logistic regression model with 8 principal components (i.e. 85.39%). KNN (with best validation accuracy of 82.63%) and SVM using OneVsRest strategy (with best validation accuracy of 78.06%) performed reasonably with the principal components. Random Forest models on the other hand did not perform satisfactorily with the principal components as the highest validation accuracy was only 64.1%.

6.3 Feature Selection Results

In this section, the most important features out of the 34 features extracted by pyAudioAnalysis have been selected using Wrappers, Filters and Embedded Methods. Feature selection techniques have been applied on the BUZZ2 dataset. The performance of some

selected subset of features have been analyzed by their classification accuracies over testing and validation data of BUZZ2 dataset. The results also show the number of mel frequency cepstral coefficients (MFCCs) selected by each method. MFCCs lie in the index range of 8 to 20, when the entire feature set (of 34 features) is indexed from 0 to 33. For more details about MFCC refer table 3.1. Number of MFCC is analyzed here as it is a significant feature in most audio classification works. It also compacts the most important characteristics of an audio sample.

6.3.1 Features Selected by Wrappers

Wrapper methods have combined our chosen machine learning models with Recursive Feature Elimination (RFE) and Sequential Forward Selection (SFS) techniques. The following sections show the results of RFE and SFS.

Recursive Feature Elimination

Recursive Feature Elimination has been applied using the RFE class from the feature_selection module of scikit-learn library [34]. It takes two parameters as input `estimator` and `n_features_to_select`. The estimator is a supervised learning estimator with a fit method that provides information about feature importance either through a `coef_` attribute or through a `feature_importances_` attribute [35]. We have used Random Forest, Logistic Regression and SVM (with Linear Kernel) models as the estimators. KNN and other kernels of SVM do not expose `coef_` or `feature_importances_` attributes and hence have not been used as estimators. `n_features_to_select` is another user defined parameter which corresponds to the number of features to select ultimately, as a stopping criterion for the recursion of RFE. We have just tried choosing the best 12, 13 and 14 features for each model, as the best performances were achieved only in this range. The performances within this range with our chosen models are shown in the table 6.4. The `get_support` method of RFE class is used to get the indices of the best features selected. The table 6.3 shows the indices of 14 best features selected by each model (using RFE) along with the number of MFCCs selected.

Table 6.3: Features selected by RFE out of 34 features extracted by pyAudioAnalysis [1]; see Table 3.1 for feature names corresponding to their indices.

Model	**Indices of 14 selected features**	**# MFCCs selected**
Random Forest	[0 3 4 5 7 9 10 13 14 15 28 29 30 32]	5
Logistic Regression	[1 3 5 7 12 13 14 15 18 19 20 23 26 29]	7
SVM (Linear Kernel)	[2 5 9 10 11 12 13 14 15 17 18 19 20 31]	11

Table 6.4: RFE Results.

Model	**# features**	**Testing Accuracy**	**Validation Accuracy**
Random Forest	**12**	99.01%	74.5%
	13	99.09%	75.4%
	14	99.05%	79.06%
Logistic Regression	**12**	85.03%	93.66%
	13	90.65%	94.23%
	14	90.13%	89.3%
SVM (Linear Kernel)	**12**	98.58%	70.39%
	13	98.32%	70.36%
	14	98.11%	70.83%

The results in the table suggest that logistic regression with 13 features is quite efficient when used with RFE as it can achieve an accuracy of 94.23% on the validation data of BUZZ2 dataset. Random forest (with best validation accuracy of 79.06%) performed reasonably, while linear SVM (with best validation accuracy of 70.83%) did not perform satisfactorily when used as estimators in RFE.

Sequential Forward Selection

Sequential Forward Selection (SFS) is implemented using the SequentialFeatureSelector class of mlxtend library [36], by setting the `forward` parameter to True. It also takes an `estimator` parameter as argument, which can be any scikit-learn classifier or regressor. We have used all our chosen models (KNN, Random Forest, Logistic Regression and SVM) as estimators. The `k_features` is a user defined parameter to specify the number of features to select ultimately. It has been set to 12, 13 and 14 integer values as the best performance (in accuracy score) was achieved in this range. The testing and validation accuracies within this range of features, with our chosen models is shown in table 6.6. The k_feature_idx_ attribute of SequentialFeatureSelector class exposes the indices of the best features selected. The table 6.5 shows the indices of 14 best features selected by each model (using SFS) along with the number of MFCCs selected.

Table 6.5: Features selected by SFS out of 34 features extracted by pyAudioAnalysis [1]; see Table 3.1 for feature names corresponding to their indices.

Model	**Indices of 14 selected features**	**# MFCCs selected**
KNN	(0, 1, 4, 5, 10, 12, 13, 14, 15, 16, 18, 19, 20, 31)	9
Random Forest	(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 32)	5
Logistic Regression	(1, 3, 5, 6, 8, 10, 12, 14, 15, 17, 19, 20, 24, 27)	8
SVM (Linear Kernel)	(2, 5, 9, 10, 11, 12, 13, 14, 15, 17, 18, 19, 20, 31)	11

Table 6.6: SFS Results.

Model	**# features**	**Testing Accuracy**	**Validation Accuracy**
KNN	**12**	97.46%	88.3%
	13	98.79%	91.16%
	14	96.09%	86.13%
Random Forest	**12**	95.49%	64.86%
	13	96.26%	65.83%
	14	96.35%	65.83%
Logistic Regression	**12**	91.29%	82.3%
	13	88.46%	84.16%
	14	92.28%	76.53%
SVM (Linear Kernel)	**12**	97.89%	69.96%
	13	98.32%	70.36%
	14	98.11%	70.83%

From the above table we can infer that KNN with 13 features is the best performer on the validation data of BUZZ2 dataset as the highest accuracy achieved is 91.16%. Logistic Regression with best validation accuracy of 84.16% performed reasonably, whereas linear SVM (with best validation accuracy of 70.83%) and Random Forest (with best validation accuracy of 65.83%) did not prove to be efficient estimators for SFS.

6.3.2 Features Selected by Filters

As discussed in previous chapters, the two filter methods we have used are Univariate

and ReleifF feature selection. The results of these methods are shown in the following sections.

Univariate Feature Selection

Univariate feature selection is applied using the SelectKBest routine from the feature selection module of scikit learn library [34]. SelectKBest selects the k highest scoring features from the feature space. The scoring functions used here are `f_classif` and `mutual_info_classif` statistical tests. SelectKBest works as a transformer and reduces the features in our dataset to a subset of selected features. These features are then used to train, test and validate with our chosen machine learning models. Results of univariate feature selection with `f_classif` and `mutual_info_classif` are shown in table 6.7 and 6.8 respectively. The results contain the indices of best 12, 13 and 14 features selected by these methods along with the number of MFCCs selected.

Table 6.7: Features selected by F-test out of 34 features extracted by pyAudioAnalysis [1]; see Table 3.1 for feature names corresponding to their indices.

# features	**Indices of selected features**	**# MFCCs selected**
12	[0 2 8 9 10 13 14 15 28 29 30 32]	6
13	[0 2 8 9 10 13 14 15 23 28 29 30 32]	6
14	[0 2 8 9 10 13 14 15 20 23 28 29 30 32]	7

Table 6.8: Features selected by Mutual Information test out of 34 features extracted by pyAudioAnalysis [1]; see Table 3.1 for feature names corresponding to their indices.

# features	**Indices of selected features**	**# MFCCs selected**
12	[0 3 5 7 8 9 10 13 14 15 16 30]	7
13	[0 3 5 7 8 9 10 13 14 15 16 30 32]	7
14	[0 3 5 7 8 9 10 13 14 15 16 28 30 32]	7

ReliefF Feature Selection

ReleifF feature selection is applied using ReliefF class of scikit-rebate [37]. Scikit-rebate is a scikit-learn-compatible Python implementation of ReBATE, a family of Relief-based feature selection algorithms for Machine Learning [38]. The `n_neighbors` parameter of this class was set to the default value of 100 neighbors. The `top_features_` attribute exposes the order of selecting each feature index according to its importance. In our case, the order of feature index selection by reliefF is:

(13, 15, 14, 18, 10, 20, 16, 9, 0, 17, 19, 30, 12, 5, 28, 11, 7, 4, 23, 29, 33, 32, 31, 2, 24, 21, 3, 25, 8, 27, 26, 22, 6).

It should be noted that out of the first 16 features in this list 12 are MFCCs.

The `feature_importances_` attribute provides the score of each feature. The plot in Figure 6.2 depicts the score for each feature index.

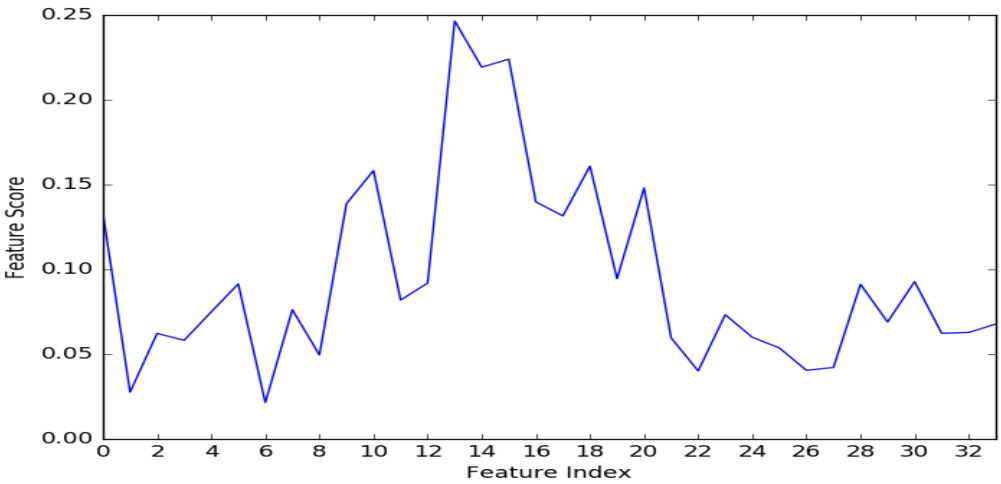

Fig. 6.2: Feature Importance plot from ReliefF method

6.3.3 Features Selected by Embedded Methods

Features are selected using Random Forest feature selection. Implemented Random Forest using the RandomForestClassifier class from the ensemble module of sklearn. 100 trees with a max_features value of "log2" is used. The "feature_importances_" attribute of RandomForestClassifier class exposes the importance of each feature, which is clearly depicted in the figure 6.3.

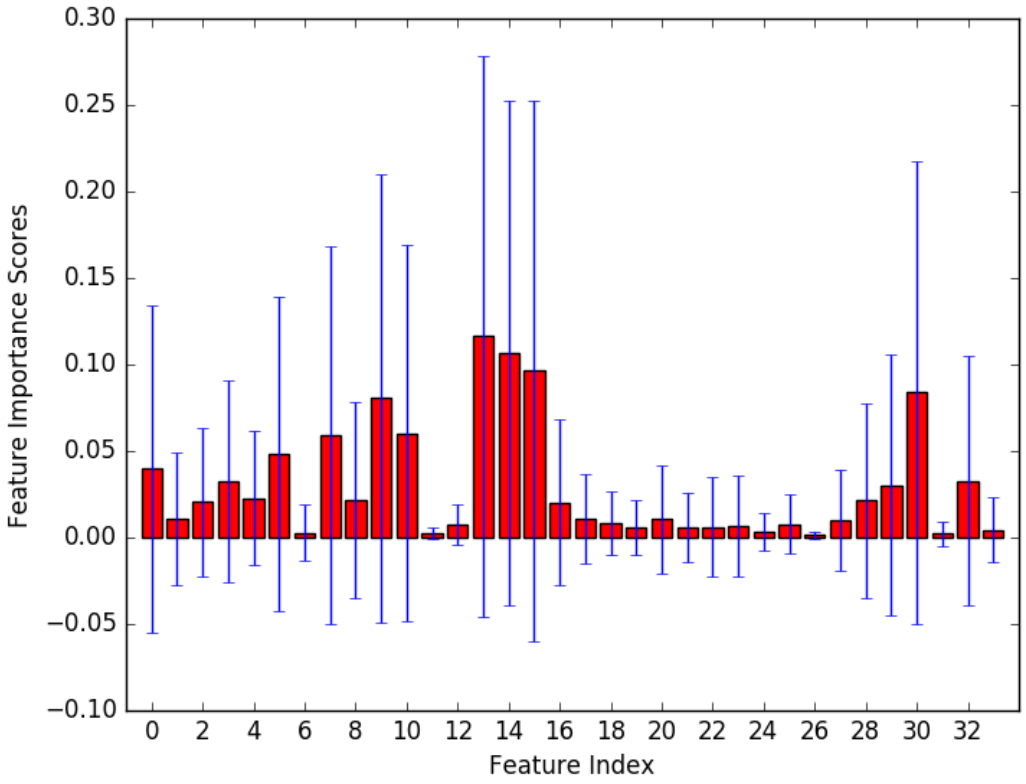

Fig. 6.3: Feature Importance plot using Random Forest Classifier

In the figure above we can see that amongst the top 6 high scoring features 5 are in the index range of MFCCs (i.e between 8 to 20).

In most of the feature selection results, especially the ReliefF results, MFCCs came up to be the most important features amongst all the 34 features extracted by pyAudioAnalysis. Finally the 13 MFCCs were selected and used with our chosen models. The table 6.9 shows the final classification accuracies of our models on the testing and validation data of BUZZ1, BUZZ2 and BUZZ3 datasets.

Table 6.9: Final classification accuracies with 13 MFCCs.

Model	**Dataset**	**Testing Accuracy**	**Validation Accuracy**
KNN	**BUZZ1**	99.58%	83.21%
	BUZZ2	98.84%	90.83%
	BUZZ3	77.49%	86.68%
Random Forest	**BUZZ1**	99.54%	98.43%
	BUZZ2	99.05%	90.46%
	BUZZ3	79.16%	97.91%
Logistic Regression	**BUZZ1**	96.11%	94.78%
	BUZZ2	77.78%	95.33%
	BUZZ3	79.97%	95.32%
SVM using OneVsRest	**BUZZ1**	99.17%	98%
	BUZZ2	97.64%	72.46%
	BUZZ3	85.21%	95.55%

In the table we can see that best performance on validation data was achieved using Random Forest model on BUZZ1 (i.e. 98.43%) and BUZZ3 (i.e. 97.91%) and Logistic Regression on BUZZ2 (i.e. 95.33%). The tables 6.10, 6.11 and 6.12 below show the confusion matrices for the best classification models on the validation data of each dataset.

Table 6.10: Confusion Matrix for Random Forest on validation data of BUZZ1.

	Bee	**Noise**	**Cricket**
Bee	300	0	0
Noise	18	332	0
Cricket	0	0	500

The confusion matrix depicts that the classes Bee and Cricket were classified accurately by Random Forest on BUZZ1 as there were no false predictions for these classes. Although, some instances of the Noise class were incorrectly predicted as Bee.

Table 6.11: Confusion Matrix for Logistic Regression on validation data of BUZZ2.

	Bee	**Noise**	**Cricket**
Bee	950	50	0
Noise	65	926	9
Cricket	0	16	984

The confusion matrix depicts that logistic regression classifies Cricket most accurately as compared to others on BUZZ2 dataset. 50 instances of Bee were classified incorrectly as Noise. On the other hand, 65 instances of Noise were classified as Bee and 9 as Cricket.

Table 6.12: Confusion Matrix for Random Forest on validation data of BUZZ3.

	Bee	**Noise**	**Cricket**
Bee	1142	27	1
Noise	34	1124	11
Cricket	0	0	1169

The confusion matrix depicts that Cricket is classified most efficiently by Random Forest on BUZZ3 dataset as there were no false predictions for this class. 27 instances of Bee were classified as Noise. On the other hand, 34 instances of Noise were classified as Bee and 11 as Cricket.

6.4 Comparison with Deep Learning Models

The best performing machine learning models, i.e. Random Forest on BUZZ1 and BUZZ3 and Logistic Regression on BUZZ2, are compared with the benchmark performance of convolutional neural network RawConvNet (designed by Kulyukin et al. in [17]) on these datasets. The input signal to RawConvNet is a raw audio wav file downsampled to 12kHz and normalized to have a mean of 0 and a variance of 1. The raw audio file is passed as a 20000 x 1 tensor to layer 1 of RawConvNet. The comparison between convolutional neural networks and standard machine learning methods is depicted in the Table 6.13.

Table 6.13: Comparison between deep learning and machine learning methods.

Dataset	**Method Used**	**Testing Accuracy**	**Validation Accuracy**
BUZZ1	**RawConvNet**	99.93%	95.21%
	Random Forest	99.54%	98.43%
BUZZ2	**RawConvNet**	95.67%	96.53%
	Logistic Regression	77.78%	95.33%
BUZZ3	**RawConvNet**	93.04%	96.97%
	Random Forest	79.16%	97.91%

The table above depicts that Random Forest (ML method) has better performance than RawConvNet (DL method) on the validation data of datasets BUZZ1 and BUZZ3. The performance of Logistic Regression (ML method) is 1.2% less than that of RawConvNet (DL method) on validation data of BUZZ2 dataset.

6.5 Model Evaluation on ESC50 Dataset

The procedure used in this research was applied over the ESC50 environmental sound classification dataset [39]. The dataset consists of 5-second long recordings organized into 50 classes, with 40 examples per class. The dataset was prearranged into 5 folds for comparable cross-validation, making sure that fragments from the same original source file are contained in a single fold. Sequential forward feature selection(SFS) technique was applied over this dataset using the train_test_split procedure from the model_selection module of sklearn library [34]. The test_size parameter of train_test_split procedure was set to 0.1 which indicates that the 10% of the entire dataset was randomly set aside for validation and the rest was used for training the standard machine learning models. Table 6.14 shows the testing accuracies of our chosen models, when SFS is used to select 21 features.

Table 6.14: Results of SFS on ESC50 dataset with 21 features.

Model	**Testing Accuracy**
KNN	36.0%
Random Forest	56.99%
Logistic Regression	30.5%
SVM (Linear Kernel)	40.0%

As can be seen in Table 6.14, the highest accuracy of 56.99% was achieved using 21 features with the Random Forest model. Piczak [40] designed a convolutional neural network for classifying the ESC-50 dataset and was able to achieve an accuracy of 64.5%. Thus, Piczak's convolutional neural network performed better than the standard machine learning models with feature engineering in classifying audio samples from the ESC-50 dataset.

CHAPTER 7

CONCLUSION AND FUTURE WORK

Feature selection reduced the original set of 34 features into a subset of 13 features containing MFCCs. These features were used with standard machine learning methods in order to achieve the accurate classification on datasets BUZZ1, BUZZ2 and BUZZ3. Random Forest classifier outperforms convolutional neural networks on the out of sample validation data of datasets BUZZ1 (of 10,260 audio samples) and BUZZ3 (of 15,254 audio samples) with classification accuracies of 98.43% and 97.91% respectively. The performance of Logistic Regression classifier is just 1.2% lower than that of convolutional neural networks on a more challenging dataset BUZZ2 of 12,914 audio samples. Thus we can conclude that the application of feature engineering techniques with standard machine learning methods can help them to perform on par with the deep learning methods and can also reduce the complexity of a classifier. The main difference between standard ML and DL methods is that ML methods require feature engineering while DL methods take longer training times to classify the data. Our investigation proves that with appropriate set of features, ML methods can achieve similar performance as DL methods in comparably shorter training times.

Effectiveness of certain subset of features may vary according to the dataset used for classification. Therefore, in our future work, we propose to automate the method of feature selection and choosing the best ML model for classification of audio datasets. Given a dataset of audio samples, our method would automatically select the best features from the current feature space using appropriate feature selection techniques and can also select the most effective ML model for accurate classification of the given dataset. We also plan to improve the performance of current classifiers by using different techniques to extract features (particularly the MFCCs) from raw audio samples and by adding more data to the training set.

REFERENCES

[1] T. Giannakopoulos, "pyaudioanalysis: An open-source python library for audio signal analysis," *PLOS ONE*, vol. 10, no. 12, pp. 1–17, 12 2015. [Online]. Available: https://doi.org/10.1371/journal.pone.0144610

[2] "Principal Component Analysis". [Online]. Available: http://setosa.io/ev/principal-component-analysis/

[3] N. El Aboudi and L. Benhlima, "Review on wrapper feature selection approaches," in *2016 International Conference on Engineering MIS (ICEMIS)*, Agadir, Morocco, Sep. 2016, pp. 1–5.

[4] R. J. Urbanowicz, M. Meeker, W. LaCava, R. S. Olson, and J. H. Moore, "Relief-Based Feature Selection: Introduction and Review," *arXiv e-prints*, p. arXiv:1711.08421, Nov 2017.

[5] "A Quick Introduction to K-Nearest Neighbors Algorithm". [Online]. Available: https://blog.usejournal.com/a-quick-introduction-to-k-nearest-neighbors-algorithm-62214cea29c7

[6] "Why Bees Are Important to Our Planet". [Online]. Available: https://www.onegreenplanet.org/animalsandnature/why-bees-are-important-to-our-planet/

[7] "Colony Collapse Disorder". [Online]. Available: https://www.epa.gov/pollinator-protection/colony-collapse-disorder

[8] P. Amlathe, "Standard machine learning techniques in audio beehive monitoring: Classification of audio samples with logistic regression, k-nearest neighbor, random forest and support vector machine," Master's thesis, Dept. of Computer Science, Utah State University, Logan, UT, 2018.

[9] Y. Yaslan and Z. Cataltepe, "Audio music genre classification using different classifiers and feature selection methods," in *18th International Conference on Pattern Recognition (ICPR'06)*, vol. 2, Aug 2006, pp. 573–576.

[10] C. C. Lin, S. H. Chen, T. K. Truong, and Y. Chang, "Audio classification and categorization based on wavelets and support vector machine," *IEEE Transactions on Speech and Audio Processing*, vol. 13, no. 5, pp. 644–651, Sep. 2005.

[11] E. Kiktova, M. Lojka, J. Juhr, and A. Cizmar, "Feature selection for audio surveillance in urban environment," *Journal of Electrical and Electronics Engineering*, vol. 7, pp. 69–72, 05 2014.

[12] J. R. Delgado Contreras, J. Garca-Vzquez, R. Brena, C. Galvn Tejada, and J. Galvn Tejada, "Feature selection for place classification through environmental sounds," vol. 37, 09 2014, pp. 40–47.

[13] N. P. Patel and M. S. Patwardhan, "Identification of most contributing features for audio classification," in *2013 International Conference on Cloud Ubiquitous Computing Emerging Technologies*, Pune, India, Nov 2013, pp. 219–223.

[14] "Weka Tool". [Online]. Available: https://www.cs.waikato.ac.nz/~ml/weka/

[15] V. Kulyukin and S. K. Reka, "A computer vision algorithm for omnidirectional bee counting at langstroth beehive entrances." Int'l Conf. IP, Comp. Vision, and Pattern Recognition, 2016.

[16] "BeePi: A Multisensor Electronic Beehive Monitor". [Online]. Available: https://www.kickstarter.com/projects/970162847/beepi-a-multisensor-electronic-beehive-monitor

[17] V. Kulyukin, S. Mukherjee, and P. Amlathe, "Toward audio beehive monitoring: Deep learning vs. standard machine learning in classifying beehive audio samples," *Applied Sciences*, vol. 8, no. 9, p. 1573, Sep 2018. [Online]. Available: http://dx.doi.org/10.3390/app8091573

[18] "A One-Stop Shop for Principal Component Analysis". [Online]. Available: https://towardsdatascience.com/a-one-stop-shop-for-principal-component-analysis-5582fb7e0a9c

[19] "Scikit-learn Feature Selection". [Online]. Available: https://scikit-learn.org/stable/modules/feature_selection.html

[20] "Mlxtend Sequential Feature Selector". [Online]. Available: http://rasbt.github.io/mlxtend/user_guide/feature_selection/SequentialFeatureSelector/

[21] K. Kira and L. A. Rendell, "The feature selection problem: Traditional methods and a new algorithm," in *Proceedings of the Tenth National Conference on Artificial Intelligence*, ser. AAAI'92. AAAI Press, 1992, pp. 129–134. [Online]. Available: http://dl.acm.org/citation.cfm?id=1867135.1867155

[22] M. T. Uddin and M. A. Uddiny, "A guided random forest based feature selection approach for activity recognition," in *2015 International Conference on Electrical Engineering and Information Communication Technology (ICEEICT)*, May 2015, pp. 1–6.

[23] "Gini Index and Entropy Value". [Online]. Available: https://www.bogotobogo.com/python/scikit-learn/scikt_machine_learning_Decision_Tree_Learning_Informatioin_Gain_IG_Impurity_Entropy_Gini_Classification_Error.php

[24] "K Nearest Neighbor-sklearn". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html

[25] "Random Forest-sklearn". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html

[26] "Logistic Regression-sklearn". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html

[27] "Support Vector Machine-sklearn". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.html

[28] "Logistic Regression-ML Cheatsheet". [Online]. Available: https://ml-cheatsheet.readthedocs.io/en/latest/logistic_regression.html#types-of-logistic-regression

[29] "Logistic Regression for Machine Learning". [Online]. Available: https://machinelearningmastery.com/logistic-regression-for-machine-learning/

[30] "Support Vector MachineIntroduction to Machine Learning Algorithms". [Online]. Available: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[31] "Support-vector machine". [Online]. Available: https://en.wikipedia.org/wiki/Support-vector_machine

[32] "OneVsRestClassifier". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.multiclass.OneVsRestClassifier.html

[33] "Scikit-Learn". [Online]. Available: https://scikit-learn.org/stable/

[34] 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," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[35] "Sklearn-Recursive Feature Eliminiation". [Online]. Available: https://scikit-learn.org/stable/modules/generated/sklearn.feature_selection.RFE.html

[36] S. Raschka, "Mlxtend: Providing machine learning and data science utilities and extensions to pythons scientific computing stack," *The Journal of Open Source Software*, vol. 3, no. 24, Apr. 2018. [Online]. Available: http://joss.theoj.org/papers/10.21105/joss.00638

[37] I. Kononenko, "Estimating attributes: Analysis and extensions of relief," in *Machine Learning: ECML-94*, F. Bergadano and L. De Raedt, Eds. Berlin, Heidelberg: Springer Berlin Heidelberg, 1994, pp. 171–182.

[38] "Benchmarking relief-based feature selection methods for bioinformatics data mining". [Online]. Available: https://doi.org/10.1016/j.jbi.2018.07.015

[39] K. J. Piczak, "ESC: Dataset for Environmental Sound Classification," in *Proceedings of the 23rd Annual ACM Conference on Multimedia*. ACM Press, pp. 1015–1018. [Online]. Available: http://dl.acm.org/citation.cfm?doid=2733373.2806390

[40] ——, "Environmental sound classification with convolutional neural networks," *2015 IEEE 25th International Workshop on Machine Learning for Signal Processing (MLSP)*, pp. 1–6, 2015.