Universita J.Selyeho - J.Selye University
Fakulta Economie and Informatiky - Faculty of Economic and Informatics

Voice recognition algorithm for recognizing one-word recordings

Bakalarska praca - Bachelor's Thesis

Kalocsanyi Daniel 2024 Komarno

Universita J.Selyeho - J.Selye University
Fakulta Economie and Informatiky - Faculty of Economic and Informatics

Kalocsanyi Daniel

Voice recognition algorithm for recognizing one-word recordings

Bakalarska praca - Bachelor's Thesis
Study programme: Applied informatics
Študijný program: Aplikovaná informatika
Field of study: Informatics
Študijný odbor: Informatika
Field of study code: 2511
Číslo odboru: 2511
Supervisor: Laszlo Marak,PhD.
Školiteľ: Laszlo Marak,PhD.
Department: Department of Informatics
Školiace pracovisko: KINF - Katedra informatiky
Date: 12 May 2025
Dátum: 12. Mája 2025

Univerzita J. Selyeho
Fakulta ekonómie a informatiky

ZADANIE ZÁVEREČNEJ PRÁCE

Meno a priezvisko študenta: Dániel Kalocsányi
Študijný program: Aplikovaná informatika (Jednoodborové štúdium, bakalársky I. st., denná forma)
Študijný odbor: 18. - Informatika
Typ záverečnej práce: Bakalárska práca
Jazyk záverečnej práce: anglický

Téma: Voice recognition algorithm for recognizing one-word recordings

Anotácia: The author shall train a machine learning algorithm that is able to recognize words in Mozilla Commons' single word voice database. The author trains at least two different words on the database using the SVM algorithm. The authors trains the algorithm and evaluates its accuracy.

Vedúci: László Marák, PhD.
Katedra: KINF - Katedra informatiky
ZOŠP: prof. Annamária Várkonyiné Kóczy, DSc.

Dátum schválenia: 23.10.2024

prof. Annamária Várkonyiné Kóczy, DSc.
osoba zodpovedná za realizáciu študijného programu

Statutory declaration

I Hereby Kalocsanyi Daniel declare that I'm responsible for creating this document completely by myself according to my best knowledge and understanding. I have not used any types of sources without the declaration in the document. Any types of citations or quotation which is not mine are marked in the text. I also mention that I will not take any responsibiility for using or editing the open-source library by replicating methods that are mentioned in the document.

Acknowlegment

First of all I want to thank you for my supervisor Laszlo Marak,PhD. without him, and his help throughout all the process this document would have never been made. I would like to thank you for your hard work and support, and most importantly I want to thank you for not give up on me even though we faced problems, what we hardly could solve. This motivation helped me keep going until the end, even though sometimes it was difficult to solve problems that occurred in this project.Thank you for the motivation, and the continous assistance what you gived me during these times.

Abstract

The project is about creating a machine learning model that can understand words from recordings. To do that, we need to build a model and calculate its accuracy of words from a database of recordings. To build a model and calculate the accuracy we need to extract data from the voice and store it, after extracting all the data we will create the model. In order to make sure that the model understands words we will test it with real-life recordings. All the recordings are from the Mozilla Common Voice database. **Keywords: Discrete Fourier Transform,Open-source, Machine Learning, Artificial Intelligence, Support Vector Machines(SVM)**

Absztract

A projekt célja egy olyan gépi tanulási algoritmus létrehozása, amely képes a szavakat a felvételekből megérteni. Ehhez egy olyan modellt kell építenünk, amely kiszámítja a szavak pontosságát a felvételből. A modell építéséhez és a modell kiszámításához szükségünk volt arra, hogy a hangból adatokat nyerjünk ki és tároljuk, az összes adat tárolása után létrehozzuk a modellt. Annak érdekében, hogy megbizonyosodjunk arról, hogy a modell megérti a szavakat, valós felvételekkel fogjuk tesztelni. Az összes felvétel a Modzilla Common Voice adatbázisából származik.

Kulcsszavak: Diszkrét Fourier-transzformáció,nyílt forráskód,gépi tanulás, mesterséges intelligencia,támogató vektor gépek(SVM)

Abstract

Projekt sa týka vytvorenia algoritmu strojového učenia, ktorý dokáže porozumieť slovám z nahrávky. Na to potrebujeme vytvoriť model, ktorý bude počítať presnosť slov z nahrávky. Na vytvorenie modelu a výpočet modelu sme potrebovali získať údaje z hlasu a uložiť ich, po uložení všetkých údajov vytvoríme model. Aby sme sa uistili, že model rozumie slovám, otestujeme ho s reálnou nahrávkou. Všetky nahrávky sú z databázy Modzilla Common Voice.

Kľúčové slová: Diskrétna Fourierova Transformácia,otvorený zdroj, strojové učenie, umelá inteligencia, podporné vektorové stroje(SVM)

Resume prace

Bakalarská práca zamierá

Contents

Abbreviations

This is the list of the commonly used abbreviations what I used in this document, with explanation of these phrases.
AI - Artificial Intelligence ML - Machine learning DFT - Discrete Furier Transform SVM - Support Vector Machines SISO - Single Input Output JPMS - Java Platform Module System CPU - Central Processing Unit GPU - Graphics Proccessing Unit SWIG - Simple Wrapper Interface Generator MFCC - Mel-Frequency Cepstral Coefficient

Part I
Machine Learning and Speech Recognition

1 Introduction

The goal of this project is to build an algorithm that is capable of recognizing spoken words from different audio recordings. To build the model, we extract the data from the recordings, and use the extracted data to train the classifier. In order to evaluate the accuracy of our model, we classify the recordings and compare the classes with the annotated database.

1.1 What is Machine Learning

Machine learning is a subfield of artificial intelligence, which is broadly defined as the capability of a machine to imitate intelligent human behavior. Artificial intelligence systems are used to perform complex tasks in a way that is similar to how humans solve problems. Machine learning is an application of artificial intelligence that uses statistical techniques to enable computers to learn and to make decisions without being explicitly programmed. The learning process is automated and is improving based on the experiences of the machines throughout the process.

1.2 History of Machine Learning

1642 - Blaise Pascal created one of the first mechanical adding machines as an attempt to automate data processing. It employed a mechanism of cogwheels, similar to those in odometers and other counting devices.

1801 - When looking at the history of machine learning, there are lots of surprises. Our first encounter was a data storage device. The device that could be considered a data storage was, in fact, was used in a weaving loom for storing stitching patterns. The first use of data storage was in a loom created by a French inventor named Joseph-Marie Jacquard, that used metal cards with holes to arrange threads. These cards represented the program that controlled the loom and allowed the procedure to be repeated with the same outcome every time.

1847 - Boolean Logic (also known as Boolean Algebra), all values are either True or False. These true and false values are employed to check the

conditions for flow control. This is how Boolean operators work. George Boole created AND, OR, and XOR operators using this logic, responding to questions about true or false, yes or no, and binary 1s and 0s. These operators are still used in computational logic today. Boolean algebra was introduced in artificial intelligence to address some problems associated with machine learning.

1890 - Herman Hollerith developed the first combined mechanical calculation and punch-card system to compute statistics from millions of individual samples efficiently. It was an electromechanical machine built to assist in summarizing data stored on punched cards.

1949 - In 1949, Canadian psychologist Donald O. Hebb, then a lecturer at McGill University, published The Organization of Behavior: A Neuropsychological Theory. Hebb referred to the combination of neurons that may be regarded as a single processing unit as "cell assemblies." Hebb's model for the functioning of the mind has had a significant influence on how psychologists view stimulus processing in mind. It also paved the way for the development of computational machines that replicated natural neurological processes, such as machine learning.

1950 - In 1950 Alan Turing created the Turing Test to determine if a computer has really been intelligent. To pass the test, a computer must be able to fool a human into believing it is also human.

2 Support Vector Machine (SVM)

A support vector machine (SVM) is a supervised machine learning algorithm that can be used for binary classification problems. After giving the SVM algorithm a set of labeled training data for each category, the trained model is able to categorize new samples. Compared to newer algorithms like neural networks, they have two main advantages: higher speed and better performance with a limited number of samples (in the thousands). The goal is to find the hyperplane in an N-dimensional space (where N is the number of features). In order to separate the data points, we need to choose the best hyperplanes which maximize the distance between data points of the classes. (Distance between points - Maximum margin). The data points are falling on both sides of the hyperplane, and the dimension of hyperplane depends upon the number of features. However, if the number of input features is only 2 then the hyperplane is just a line. As the number of features increases, the hyperplane becomes higher-dimensional.

The major benefit of SVM is that it can handle high-dimensional data. Most of the time the algorithm finds the best fitting hyperplane, and the

algorithm can embed the data into high-dimensional spaces, in that the data points cannot be separated linearly.

SVM is developed in a way that is quite different from how neural networks have evolved. SVM is built on a solid theoretical foundation, which really set it apart when it first came into play, and it was put to practical use and tested in real-world applications. Neural networks, on the other hand, are gaining popularity in applications. It is interesting to note, that SVM can still outperform neural networks and other statistical models in some applications. Particularly when computational capacity is scarce, such as embedded devices. When we talk about the learning framework for SVM, we can think of it this way: there is an unknown and nonlinear relationship described as $y = f(x)$, indicating how a high-dimensional input vector x relates to a scalar output y. The information we have to work with consists of a training dataset $D = \{(x_i, y_i) \in X \times Y\}$, where $i = 1, \ldots, l$ and l tells us how many training samples we have, which gives us the size of the dataset D. In cases, where the classes in the dataset cannot be linearly separated in the original input space, SVM makes use of a smart technique called the kernel trick. This approach allows the model to apply a non-linear transformation, effectively mapping the original input space into a higher-dimensional feature space. This transformation greatly helps the model to find a separating hyperplane that distinguishes between different classes efficiently. Methods like Radial Basis Function (RBF) kernels or Gaussian functions are often used for this non-linear mapping, as they help capture complex relationships in the data. By using these advanced techniques, SVM is well-equipped to handle a variety of classification problems, making them valuable tools in fields such as finance, bioinformatics, and image recognition.

While this perspective on SVM will be consistently used throughout the remainder of our thesis, it is interesting to take a moment to explore how a geometric idea gave rise to the initial algorithms. The support vectors are data points that are closer to the hyperplanes and manipulate the orientation of the hyperplanes. Using support vectors we set the maximum margin of the classifier.

SVM can be used both for regression and classification, but mainly used for classification objectives. Robustness is a key factor when we are talking about SVM.

The main advantage of SVM is its ability to handle high-dimensional data. Typically, the algorithm identifies the ideal hyperplane that separates the data linearly by extending it into high-dimensional space. One of the key advantages of SVM is its ability to handle both linear and non-linear data distributions. Common kernel functions include the Linear Kernel, Polynomial Kernel, Gaussian Kernel and Radial Basis Function (RBF) Kernel, each

suited for different types of problems.

In SVM algorithm we are looking to maximize the margin between data points and the hyperplanes. In order to do that we actually need to calculate the loss function, which will help us to determine the maximum margin.

3 Sound Processing

Sound waves are mechanical longitudinal waves that are propagating in the air. Sound waves have two important characteristics: frequency, which determines the pitch of the sound, and amplitude, which determines the volume of the sound. Using a simple algorithm, the single-input and single-output (SISO), we take the input from the signal and produce the output according to the sample. To process any continuous signal by computers, the signals need to be discretized into to discrete samples. The procedure that transforms the signal from continuous to discrete time is usually called sampling, by picking the values from the continuous-time signal. The phrase called sampling interval is where the signal has multiple quantities.

Java Sound Processing Java has its own API for sound processing, called Java Sound API. The API provides detailed support and documentation for capturing, processing and rendering audio signals. Only uses a small amount of system resources. It can be used to manipulate resources such as digital audio signals, and devices that require sound cards.

Examples of sound processing Sound can be stored as digital files with sound processing methods. WAV, CD and MP3 are the most common audio formats. Some applications of voice processing and speech occur here for example recognition of human voice patterns and phone network voice calls.

3.1 Sound Formats

Sound Format defines the quantity and loss of audio data. They are divided into 3 categories: Uncompressed Formats, Lossy Compressed format, Lossless Compressed Format. There are various types of audio formats each one has its advantages and disadvantages in audio data compression. If we select an audio format it is highly recommended considering some factors such as compatibility, size of the format, and sound quality. These above-mentioned parameters play a big part in finding the best format when it comes to either of these three aspects. We talk about the uncompressed audio format when the raw data is kept in the audio format, although the disadvantage of that

is that the data comes with a larger file sizes. Among the most popular uncompressed formats, we find WAV and AIFF formats. When we talk about compressed audio formats it is quite the opposite of uncompressed formats. Compressed formats solve the issue of large files in exchange for processing power needed for decompressing the signal. It uses a unique algorithm that removes any of the data that is not suitable for the human ear. This action reduces the file size and makes the format more compatible with other formats. There are a lot of compressed formats most popular formats include MP3, M4A, AAC, and FLAC.
Uncompressed file formats are :

- PCM
- WAV
- AIFF

Lossy compressed formats are :

- MP3
- AAC
- WMA

Lossless compressed formats are:

- FLAC
- ALAC.

Sound file formats are digital standards for storing audio information. PCM needs to be organized into a file so we can work with it, or play it back on a system. Different audio file formats use different containers and varying methods of data compression to organize the PCM stream. Depending on which one we choose, each format represents the same target information in different storage sizes or quality.

PCM(Pulse Code Modulation) - Represents raw audio signals in digital forms. To convert analog signal into digital it has to be at different interval. It has sampling rate and bit rate, the number of bits used to represent each sample in the audio.

WAV(WaveForm Audio File Format) - It was developed by Microsoft and IBM. It is a Windows audio format. It is just a wrapper most cases these WAV files contain uncompressed audio in PCM format.

AIFF(Audio Interchange File Format) - Developed by Apple for mac. They can obtain multiple kinds of audio format. This format is also a wrapper for PCM encoding. Containing uncompressed audio in PCM format. It is compatible both with Windows and Mac operating systems.

4 Discrete Fourier Transform

The Discrete Fourier Transform (DFT) is a widely used transformation algorithm. It is used in digital communication, image compression, signal processing, etc. Fourier transforms on discrete signals can be done using DFT, which can be used to switch back and forth between the time and frequency domains. The time signal needs to be discrete to be processed, because the digital computers do not work with continous-time signals. Discrete Fourier Transform is a method, which enables us to find the spectrum of finite-duration signal. There are some conditions, which need to be fulfilled in order to decide the frequency of the time-domain signal. When the signal is periodic, then is possible to use DFT methods for signal representation. In this case the frequancy domain spectrum will be discrete and periodic. On the other hand if the signal is non-periodic or the finite duration of the frequency domain is not continous, then the implementation of DFT is more complex. Using periodicity representation of the finite sequence allows us to convert these signals into frequency domain. DFT can be computed using the discrete-time signal with a period N, $X[k] = \sum_{n=0}^{N-1} x(n)e^{-j\frac{2\pi n}{N}\times k}$, where N is called the size of the DFT, and X(k) and X(n) is a sequance of complex numbers. The frequancy domain spectrum will be transformed into time-domain, which is the main reason why we use Discrete Fourier Transform. The DFT demonstrates continious signal processing using sine and cosine functions, in order to convert the frequancy domain into time-domain.

Using Discrete Fourier Transform for voice recognition Using DFT for voice recognition is quite straightforward you have to first analyze the voice and separate these signals into different segment, which will be used when applying DFT methods. Using this strategy method the voice can be recognized from the file format using DFT technique, which utilizes this sampling frequency segmentation method in order to achieve highly recognized voice results.

5 Voice Recognition Datasets

A database is a repository of data, it is designed to store data conveniently. Various kind of databases exists to satisfy different kind of industry expectations. In the database, you may store binary files, documents, relational data, and so on. Data can be stored in many forms namely tabular, hierarchical, and graphical forms. When data is stored in a tabular form called a relational database. However, when the data is stored in a tree form then it is called a hierarchical database.

5.1 Mozilla Spoken Words Project

Data collection is the process of gathering and measuring information on variables of interest, in an established systematic fashion that enables one to answer stated research questions, test hypotheses, and evaluate outcomes. Data collection is one of the most important stages in conducting research. You can have the best research design in the world but if you cannot collect the required data you will be not be able to complete your project.

Mozilla Common Voice project is designed to help to build a voice database for speech recognition. The project is supported by volunteers worldwide. They record their voices and sentences, also they review other peoples recordings. These recorded sentences are collected in the voice database, which is publicly accessible.

Mozilla Common Voice is a free database that builds voice data sets for speech recognition technology. The project is supported by volunteers worldwide who are prompted to voice record sentences and review recordings made by other users. These transcribed sentences are collected in a voice database so that our technology has access to a diverse subset of voices, represented by all people. For instance, this project may help a machine understand a particular dialect to be used in a GPS, or make technology become more universally available in multiple languages.

Number of Samples, and the Types of Recorded Words The Common Voice dataset contains more than 2000 hours of recorded words. It's a diverse way to represent data samples and has a unique approach to record words worldwide from different users. It is quite remarkable that the words can reach more than 1500 samples per word. Sample size is making your dataset more robust and reliable, it plays an important role in every research project. Finding the right sample from your dataset can be challenging, it requires a particular level of understanding when it comes to selecting the sample from a machine learning dataset. Sometimes you may encounter

different problems while you searching for the right sample for example unbalanced dataset problems with the samples giving you hard times when you want to return a specific sample.

6 Searching for Machine Learning Libraries

6.1 Finding the Library for Machine Learning

First we needed to find the best library for us to create the model. We started by looking on the web for a library that would do this task for us. We found a lot of different libraries compatible with Java, but not all libraries were right for the task. Furthermore, we selected two candidate libraries Weka ML and DLib ML. Finally, we have concluded, that for us the best pick would be DLib ML. It has support for a lot of languages, and with a right wrapper, we can implement to train our model.

6.2 Choosing the Library for Machine Learning

To choose the best machine learning library for this task was quite difficult, because we had so many options in Java. We started with Weka which is a machine learning library for Java, but later we realized that it was not an optimal choice. It was due to the fact, that Weka has too few features SVM training. On the top of that it has issues with optimization as well. Considering these factors and issues with parameter optimization the right choice was to choose DLib. It is reliable, and it also supports for SVM extensively. DLib also has large documentation and an active community, should we encounter any type of compiling error.

7 Java Machine Learning Libraries

7.1 Java and its Modules

Java is a versatile and unique programming language. It has its own modules named packages. Java Modules provide a packaging versatility that allows us to package a Java application or Java API as a separate module. A Java module is packaged as a modular JAR file. It consists of one or more Java packages that are related and belong together. Each module has its unique name, following the same name as Java packages.

Java Platform Module System (JPMS) Java Platform Module System (JPMS) - It is a new feature implemented in Java to sort and manage modules.

Important Modules in Java Java Application Interfaces(APIs) are sorted into methods, classes, and packages. The highest package level is called the module, and it is an essential part of many Java programs. A module can have many parameters, like name, dependency, API, and services. The API comes with these built-in components and modules.

Java machine learning requires using an algorithm to parse data, learn from previous experience and make predictions based on the data. The algorithm target is to learn the function that consist of several input data. Most commonly used ML algorithms are Linear Regression, Logistic Regression, K-Nearest Neighbors and Support Vector Machines. Machine learning has a subset field called Deep learning which describes as artificial neural networks with a huge number of parameters and layers.

7.2 Deeplearning4j

Is a large deep learning library for Java. It has a friendly user interface with great community, which offers decent performance when it comes to machine learning. Also, it is an open source library, and it can be accessible to everyone who seeks to build machine learning based applications. Deeplearning4j library uses Spark engine in order to distribute learning parameters both for CPU and GPU. When it comes to training the library well-equiped with a training user graphics interface, which will simulates the training with a chart, what will show you the end result of the application. The library has several neural network layers with a single input and single output layer. It is a widely used machine learning library with many toolkits which helps to identify different kind of objects and images in order to recognize the smallest details, which used in image recognition.

7.3 Weka

The library was implemented for the purpose to process data within a wide-range of applications, however the machine learning methods and data mining capability of the library gives more opportunity to involve more in different sectors of machine learning. It contains a set of algorithms, including tools for data mining, attribute selection, and regression. The graphical user interface has many panels, which is accessible from the top tab, and data can be loaded either from a file or extracted from a database using SQL query commands.

Important to note that the architecture of the library is well-designed to be modular, which is allowing algorithms to be combined in different way. Most of the algorithm have more options that can be adjusted based on the dataset. To make it more flexible and convenient as possible it has many type of architectures, which allows new classifiers, and clustering algorithms to be implemented and added to the library easily.

8 C++ libraries for machine learning

8.1 DLib Library For Machine Learning

DLib is a cross-platform open source software machine learning library written in the C++ programming language. That means DLib is first and foremost a collection of independent software components. DLib components are supported by documentation and debugging modes. Moreover, the library is designed to be a useful tool for not just research purposes, but also helps us with real world projects too. It has been carefully implemented to make it easy to integrate into any C++ project. The library consists of multiple components : Bayesian Nets, Linear Algebra, Optimization, and Machine Learning Tools.

Part II
Training the Machine Learning Model for Speech Recognition

9 SWIG

SWIG is a software development tool that simplifies the task of interfacing different languages to C++ libraries. In a nutshell, SWIG is a compiler that takes C/C++ declarations and creates the wrappers needed to access for those declarations from other languages including Perl, Python, Tcl, Ruby and Java. SWIG requires the use of interface files for input. SWIG was originally designed to make it extremely easy for engineers to build extensible software without having to write complicated code.

9.1 Why SWIG

Using SWIG, we can replace the main function of a C program with a scripting interpreter from which we can control the application. SWIG allows C and C++ programs to be placed in a scripting environment as well that can be used for testing and debugging. SWIG works by compiling C and C++ header files directly into scripting language wrappers. This process is fully automated and requires few, code modifications. As a result, SWIG can be bound into any project with minimal coding and it allows people to focus on the problem instead rather than the language integration problems. Furthermore, SWIG tries to maintain a clean separation between the application and interpreter.

9.2 Examples of Using SWIG

To build a scripting language interface, the user must create a special file containing SWIG modules and the C++ implementations what they would like to access. However, SWIG is also able to process raw header files, which compilation has been used to embed special SWIG functions. This makes it easier for everyone to integrate SWIG into their application since existing source code can be used as input. A simple example how swig can be used in C++ programs:

```
#include "shockwave.h"
int integrate(int nsteps, double dt);
void setboundary-periodic();
void init(double epsilon, double sigma, double cutoff);
void setoutputpath(char *path);
```

This example also show us that SWIG can hold declarations and wrap it to make the reading easier to the interpreter. Because of this, it gives modularity and allows pieces of software to be used in different kind of interpreter settings.

10 Creating the SWIG Module

To wrap C++, SWIG uses a layered approach to code generation. At the lowest level, SWIG generates a collection of procedural ANSI style of wrappers. These wrappers take care of basic type conversion, type checking, error handling, and other details of the C++ binding. These wrappers are also sufficient to bind C++ into any target language that supports built-in functions. In some cases, you might see this layer of wrapping as providing a

C library interface to C++. On top of that is a low-level function interface, SWIG generates proxy classes that provide a natural object-oriented interface to the code. The proxy classes are typically written in the target language itself. For instance, in C++ a specific class is used to provide a wrapper around the object.

10.1 Building the SWIG module

First thing first we will create the SWIG core module using C++ scripting language. For C++, SWIG creates wrappers that mirror class interfaces in the chosen language. In order to do that we will create an object which will hold attributes from the inherited class. These attributes are represented as a pointer object which both contains the value of the pointer and the tag attributes as well.

10.2 Importing the SWIG module

To import the SWIG module first it is required to have the necessary libraries imported to our project. SWIG has his own library, which we have to import first in order to work with it properly. The front-end consist of a C++ preprocessor and parser. Its primary using interfaces to parse the functionalities of the scripting language. Importing the module involves several steps from creating the interface file to compiling the C++ code and generating the shared library. To import the module first we have to install SWIG in our system by using the `brew install SWIG` command in the terminal. This will download the necessary files from the internet to use the SWIG module and automatically imports into the choose location in our system. In the next step we will write a simple C++ file that will contain the classes and functions for the given interface. After that we will generate the wrapper code for the targeted programming language like in the example shows above. Lastly but not least we have to compile the generated wrapper code along with the C++ code in order to use the library in our project.

11 Training the Sound Model

In this section we are going to train our machine learning algorithm by giving it input data from our database. The learning will improve if the dataset will remain consistent, and it will also improve our prediction rate for the model. The process of running a machine learning algorithm on a dataset, is known as model training. During model training, the algorithm adjusts

its parameters to minimize errors and improve its performance. To ensure the robustness and accuracy of a machine learning model, techniques like cross-validation are deployed. Cross-validation calculates the accuracy of the machine learning model separating the data into two groups these two groups called training set and testing set. After that the data is separated into folds which are subsets of each group. The number of folds depends on the size and data which are defined in our model.

11.1 Normalization

Before training the model it is important to note that first we have to normalize the data which will helps us to process the signal, and improve our model prediction result. Normalization is a specific form of feature scaling that transforms the range of features to a standard scale. Normalized data enhances model performance and improves the accuracy of a model. By normalizing the features, we can ensure that each feature contributes proportionally to the model's learning process. The model can now learn patterns across all features more effectively.

11.2 Training

Training models in machine learning is an essential process that equips machines with the ability to spot patterns, predict outcomes, identify anomalies, and test correlations. This stage is a vital part of the process that allows businesses to leverage machine learning to extract valuable information to enhance experiences and enable better decision-making. Model training in machine learning refers to the process of teaching a machine learning models to recognize patterns, make predictions, or perform specific tasks. During training, the algorithm learns from a set of labeled data and adjusts its internal parameters to minimize errors and optimize its performance. The trained model can then be used to make predictions based on the data it has learned. In this stage we will train the model using Dlib library built-in functions, from the class called `dlib::matrix` we create a `sampletype` variable which we will use to access Dlib functions like train to train our model and compute the model accuracy based on the input datasets. For kernel type decided to use `radialbasiskernel` which is coming from `dlib::decisionfunction` class.

Extracting the Feature Vectors To extract the feature vectors we store the voice recording sample in an array. We created a function for extraction called `ExtractFeatures`, which will extract the features from the database recordings.

11.3 Feature Extraction

We used the sound vector to segment the audio signal at specific milliseconds per frame rate for feature extraction. We chose the Mel-Frequency Cepstral Coefficients (MFCC) as the main feature extraction method for our machine learning model due to their effectiveness in speech recognition and their capability to recognize audio signals. MFCC helped us analyze voice by focusing on how people hear sounds. When we talk about feature extraction the main goal is to reduce the dimensionality of the input vector while still maintaining the information content of the sample.

Mel-Frequency Cepstral Coefficient Mel Frequency Cepstral Coefficients (MFCCs) are audio representation coefficients. The MFCCs are different from cepstrum representation of audio data. The difference between Mel Frequency Cepstrum and standard cepstrum is that the MFCC frequency positioned logaritmically. The coefficients are generated by an unique algorithm, which are representation of signal spectrums with data compression. The main purpose of using these coefficients are audio compression and Automatic Speech Recognition.

Using MFCC for feature extraction Mel-Frequency Cepstral Coefficient(MFCC) is a method based on human hearing behavior. It can recognize different kind of frequencies. In the computation of MFCC the first thing to do is windowing the signal to split the audio signal into frames. Since high-frequency domains have a lower amplitude compared to low-frequency formats, it is important to prioritize high-frequency domains in order to achieve similar amplitude for all formats. After windowing the FFT is applied to find the power spectrum of each frame.

11.4 Framing

Speech samples are segmented into small number of frames(N number). These speech samples determine the framing process, which are segmentation of samples within range of 20-40 ms. The input signal is segmented into frames per ms with an overlap of half the frame size. In case of $\frac{20\text{ms}}{\text{frame}}$ the signal remain consistent and the following equation valid for all N samples: $N = \text{fs} \cdot \text{tframe}$, where fs is a sampling frequency and tframe is a length of the frame which is measured in time. The frame size has to be equal to power of two in order to apply Fast Fourier Transform on it. The frame is shifted between overlapping and frame size to avoid the risk of losing the audio signal.

Hamming, windowing Windowing is used when the audio signal has a limited amount of length. In case of windowing the frame the function is being multiplied. Windowing is most commonly used to minimize each frame of signal in the beginning and in the end of each frame. When using Hamming windowing technique the window is multiplied by each frame to keep constant the first point and the last point in the frame.

Fast Fourier Transform In order to convert all frame samples from time domain into frequency domain we have to apply Fast Fourier Transform to the given dataset. Each frame having N number of samples which are converted into frequency domain, meanwhile also minimizes the calculation number. The input given to the Fourier Transform is a windowed signal and the output for each frequency band is represented as a magnitude of that frequency.

Mel Filter Bank processing Each signal of speech consist of tones with different frequencies and each of these tones are measured through Mel-Scale using the following equation: $f(mel) = 2595 \cdot \log_{10}(1 + \frac{f}{700})$, where f is a frequency. This equation is used when it comes to implement Mel-Filter Banks, and the magnitude coefficient for each Fourier Transform segment.

Discrete Cosine Transform Discrete Cosine Transform is a sequence of finite data points in which the cosine function is moving between different kind of frequencies.

Dimension reduction(PCA) Reduces the dimension of the feature vectors that we obtained through feature extraction method. Uses various techniques to reduce the dimension and plays an important part when it comes to training. When we want to reduce the number of features in our dataset we have to use dimension reduction(PCA), which will solve this problem for us. PCA will try to find the approximate linear space, where are the most of data variance. When PCA finds that approximate space, then it will try to pick a direction where is the maximum variance available.

11.5 Dimension reduction with DLib

To reduce the dimension of the feature vectors DLib has its own built-in function `dlib::vector_normalizer_pca`, which not only apply the dimension reduction, but also normalizes the dataset. We used this method to apply the dimension reduction. This function has a variance parameter, which is

between 0 and 1 , however it has to be adjusted based on experience, in order to get better accuracy result. We tried with different variance numbers, but the best result always was, when the variance was set approximetelly to 0.9. After some research on the topic we also discovered that this number have a limitation which is 0.99, and if we set this over the limit the model just not performs the reduction accurately.Finally in the end we managed to reduce the dimension of the vectors and we displayed the result to the screen. The number of output vectors was strange at first, but in the end everything was calculated according to the plan.

12 Optimizing training parameters

To optimize the parameters have to set the two parameters ν , γ starting from lower number, and increase it based on the testing result accuracy. If the number is too high then the model prediction is not quite accurate. Hyperparameters are set before training the model and their values are affecting the model prediction and accuracy. It is challanging to select the optimal hyperparameters for machine learning models, since if we choose poorly it can lead to performance issues. The difficulty in selecting the optimal hyperparameters for our model is quite hard , since these parameters are more complex, and often they are in a high-dimensional space. There are different techniques what can help us to find the optimal parameters. Before training these hyperparameter values need to be optimized, in order to have the best training result as possible. With the right combination it is possible to achieve great accuracy, but for that we have to try every combination based on our experience and compare the results to find the best optimal parameters for both ν and γ.

Adjusting parameters based on training Based on the training result we have an opportunity to adjust ν and γ in order to make the model more accurate. This involves with changing the parameter values from the initialized value. In our case those parameters were set to an initialized value, which we needed to adjust based on the training result.

13 Description of the functions

In this section we will be describing the functions what we used in order to train our model. These functions are built-in function, which are implemented in DLib, and they have a important purpose when it comes to

training the model. Furthermore it shows that every class of the library have an important role during training stage, because every function what those classes contains described below makes our final model more accurate.

Random Shuffle Using DLib built-in function randomshuffle we will shuffle the positives and negative samples in the dataset. First we have to random shuffle before we normalize the samples in our dataset. We use built-in function for that `dlib::random_shuffle` and it will random shuffle all the samples in our dataset. It is very

Normalize For normalizing the dataset we using DLib built-in function `dlib::normalize` this function will make the nomalization using different kind of techniques. Normalization is kinda neccessary, before training, because with normalized data the model gives more accurate result. With normalization we achieved a more consistent and reliable dataset.

Balancing the dataset For balancing the dataset we created a function called `BalanceDataSet`. This function will equally balance the positives and negatives samples in our dataset. In order to do that we had to calculate the positive and negative samples in the dataset and store it in a count variable, and than we compared these with the input values. After comparison we just pushed back the input parameters using the `.push` function to their original place to balance out our dataset.

Training First and foremost we have to normalize and random shuffle the dataset, before we train our model. After that balancing the dataset is recommended ,since it will improve our model parameters. For training we used a built-in DLib function called `dlib::train`,which we used to train our model and evaluate its accuracy.

14 Training set

Training set is a collection of labels and input data to train the model. For SVM, the training set allows the model to learn patterns. and data quickly. There are a few key components that make the training set more accurate and reliable these component are:Feature Vectors, Labels, Data Distribution, Size of the training set, and Normalization/Scalling. It contains neccessary datas to train our model. The labeling procedure helps us to seperate positive and negative samples in our dataset. The input data also seperated into two

group positives and negatives. In order to work with the data more efficiently the training set have to be scaled and normalized between a certain treshold.

14.1 Collecting training data from dataset

In order to train our model first we had to collect the data from our dataset. We downloaded recording from the database and stored it in a folder. Then we extracted the samples from the dataset , and stored it into a raw file. We created a simple .sh program that will extract every recordings from the database. When we done with the extraction we stored these data into our project and we had the rigth set for training our model.

14.2 Model Prediction

The model receives feature as input and predicts the output based on the model. During training the model utilizes patterns and predicts parameters to enhance the accuracy of the model. When using predictions in machine learning models we focus on improving the decision-making and predict values from previous experience. To make predictions in our model based on past data we use different kind of techniques and algorithms, to find relationships between input data and outcomes during training phase. In our case we used SVM algorithms to predict model output based on training dataset. When we are predicting models it is essential to have enough data in the dataset, since its enable to us predict model more accurate and make decisions more frequently.

15 Training

15.1 Training with DLib

For training the model training the library DLib was used to randomize samples and train the feature vectors. After the extraction of the feature vectors the model needed to be balanced and shuffled randomly. Training the model is a process of teaching the machine learning algorithm to recognize speech and make predictions based on the dataset. Using Dlib library this approach is pretty straightforward since the library has extended functions for training large amount of datasets. To perform these tasks there are built-in functions like `dlib::train` which is used to train our model and evaluate the accuracy.

15.2 Training with Parallel Solution

For faster execution imported the library QtConcurrent threads which helped us to evaluate the model faster. The model training time reduced from 20 min to 10 min which is a huge improvement compared to the previous version. It is import to note that this only reduces the time of debugging, while using maximum threads available in our system. We decided to use QtConcurrent since it is a faster and more reliable when it comes to fast debugging. To use it effectively in our code we used 8 threads, because in this situation we don't use all the threads and also improve the model evaluation time. In this way we achieved result faster and more frequent, which was us great news since we had problems with model evaluation time.

16 ROC Accuracy

A Receiver Operating Characteristic (ROC) curve represents the relationship between true positive, and the false positive, over various classification score thresholds. The area under the ROC curve determines the overall performance of a classifier by calculating the area under the curve. AUC values range from 0 to 1, with higher values identifying better classification result.

16.1 Confusion Matrix

A confusion matrix is a contingency table that is used for describe the performance of a classifier. Each column and row represents the number in the predicted class while each row represents the number in the true class. There are four numbers in the matrix: true positives, true negatives, false positives, and false negatives. Each row of the matrix represents the number of samples of the predicted class , and each column represents the number of samples of the actual class. The confusion matrix calculates the model performance, and helps,if the model makes mistakes during classification.

16.2 ROC parameters

Accuracy Accuracy is a useful parameter, especially when the classes in the dataset are balanced, which means they have equal number of positive and negatives samples in the dataset. In inbalanced datasets the accuracy can be misleading, since it will give high percent result, because it will predict the majority class in the dataset. Accuracy shows us how much the model understands words from the recording. It is a fraction of the correct and all prediction, and it can be calculated using the confusion matrix numbers.

Precision It is a fraction of the correctly predicted positive results. This parameter is kinda usefull when false positives outnumber false negatives. When this number is high, that means the model has a low number of false positives, and its only makes positive predictions. In case of low precision value the model has high number of false positives, which results that the model makes more incorrect prediction. There are some cases, where is better to measure precision of the model.

Recall This parameter measures the actual positives predicted correctly. It is focuses on the actual values predicted by the model. Recall is used when we need to minimaze the false negative samples. A high value of recall tells us that the model has a low false negative samples. The low value of recall indicates otherwise, and have high number of false negative samples.

17 Hyperparameter tuning

There are two parameters in the SVM algorithm with the RBF kernel. Both play important role in pattern recognition and audio extraction. These parameters play key role when it comes to tuning. First one is called γ which gives the value of how deeply we want to embed the samples into the kernel space, while the ν number is controls the number of tolerated outliers in the dataset. To achieve better result of accuracy these parameters need to be tuned for our case γ needed to be approximately 1e-6 and ν not more than 0.5 to achieve a better accuracy result. The max ν can be found by calling the model max ν function from the DLib library. This function will return the max ν value, and it can be used when we are tuning these parameters inside the for loop.

Setting Parameters ν and γ must be initialized in the beginning we set those values to a number first then in the later stage we will adjust it. Setting those parameters gives different type of results so its crucial to set those values between a certain threshold. This treshold can be in a range of low-high value. In our situation setting those first to a lower number is makes more sense, because after training we still can adjust those parameters if the result are not satisfying.

Adjusting Parameters γ and ν parameters are need to be adjusted based on training, since these parameters are responsible for better accuracy result. In our case we had to adjust the parameters, because if we leave those number at constant or the value they were initialized, then we never reaching better

accuracy result. When it comes to adjust these parameters it is nice to know the maximum ν , and adjust the parameters based on that. We can return the max ν using DLib built-in function max ν.

18 Testing Results

After training our model we can say that adjusting hyperparameters made a huge difference, since it improved our whole model performance. First we tested with testing dataset, because we wanted to make sure that the model working properly. We achieved a great accuracy result with the testing dataset, and it was time now to test it again but now with the real dataset. We trained the model again and evaluated the results. We also compared these two result, we wanted to know if there is any difference in term of performance or accuracy. We were surprised to find that there is not much difference between these two testing results. Our next step was to test our dataset with the best possible parameters.

18.1 Testing with the Best Possible Parameters

To test with the best (ν) and (γ) we created parameters called Bestν and Bestγ initialized it to 0 first and then we used this variable in our cross-validation training. First and foremost we created a gamma and nu parameters inside the training function and then used these values to train the model. After training we set these previous gamma and nu values to be equal to Bestν and Bestγ . We trained the model again but now with the best possible parameters. The result were way better with Bestν and Bestγ , than with γ and ν, since those parameters are giving us the best possible outcome result, when it comes to training.

18.2 Final Testing Result

The final training result were calculated with the best possible parameters and accuracy as well. In order to achieve that result we tuned the hyperparameters to a threshold where the accuracy is over 0.85, so we can reach the best possible result in the end of training. We also set the variance of the PCM parameter to 0.9, in order to get better accuracy. Despite everything first it gives us lower accuracy result, but when we set the variance to 0.95 it clearly gived us better result. We realized that with higher variance the model seems to give way better results than before, so we keep that number as a constant for a future use. At the end results were suprisingly great with

over 85 percent accuracy , which is not that bad , since there are always some background noices in every recording. Overall we succeded and was pretty happy with the results.

19 Summary

In this thesis we created a machine learning algorithm, which is capable of performing voice recognition using SVM algorithms. When we started this project we had no idea how we going to find solution for problems that may occur during model training. After some research and documenation we found some solution which we implemented immediatelly, but then we faced problems which were difficult to solve. Later we realized that cannot find perfect solution for every problem that occur during model training. It was a long road and sometimes I had a feeling that we getting nowhere because we could not find a solution for the current problem in the implementation, but we never give up, which was the key in our success in the end. First we started implementing the model with DLib library and slowly but surely we read all the mp3 files into our project and used a feature extraction method which helped us to extract the features from samples. After we done that we started to calculate how much samples we have in the dataset. We realized that the number of positive and negative samples in the dataset was not equal so we had to came up with the solution that solves that problem. For solving that we created a function BalanceDatasSet, which will balance the dataset and make sure that we have equal number of positive and negative samples. After that we had balanced dataset the next step was to train the model using DLib library. During training we managed to achieve better accuracy result by using cross-validation techniques, and adjustment of hyperparameters in our project. Next we calculated the ROC parameters and accuracy using our function which we implemented in order to display the parameters in our project. When we finished to implementing that the next step was to set the hyperparameters to a specific value, so we can achieve higher accuracy result based on training. It was pretty straightforward to do that but we had to do some research,before we set those parameters, since if we set too hight then it gives lower accuracy result. After some research we decided to set the parameters to a lower value, in order to achieve higher accuracy. In the end we manage to get the desired accuracy what we wanted. Pretty sure that machine learning will improve everyday , and in the future we will have more opportunity to implement these algorithms. Lastly but not least as a final note I want to leave this as a motivation, that everything is possible it just a matter of time.

References

[1] Available online: https://www.lightsondata.com/the-history-of-machine-learning/

[2] Available online: https:https://dataconomy.com/2022/04/27/the-history-of-machine-learning/

[3] Available online: https://mitsloan.mit.edu/ideas-made-to-matter/machine-learning-explained

[4] Available online: https://www.mygreatlearning.com/blog/what-is-machine-learning/

[5] Ingo Steinwart Support Vector Machines [book] 2008, ISBN: 978-0-387-77241-7 Available from:

[6] Available online:https://monkeylearn.com/blog/introduction-to-support-vector-machines-svm/

[7] Available online: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47

[8] Available online: https://www.masteringthemix.com/blogs/learn/audio-file-formats-explained

[9] Available online: https://immersiveaudioalbum.com/an-introduction-to-audio-file-formats/

[10] Available online: https://www.geeksforgeeks.org/audio-format/

[11] Available online:

[12] Available online: https://towardsdatascience.com/learn-discrete-fourier-transform-dft-9f7a2df4bfe9

[13] Available online: https://medium.com/@kamil2000budaqov/discrete-fourier-transform-75d9a9f37821

[14] Available online: https://uconntact.uconn.edu/event/6149354

[15] Available online: https://www.mrowe.co.za/blog/2018/10/mozillas-common-voice-project/

[16] Available online: https://riptutorial.com/swig

[17] Available online: https://www.swig.org/Doc3.0/Introduction.html

[18] Available online: https://blog.landr.com/audio-file-formats/

[19] Available online: https://emastered.com/blog/audio-file-formats

[20] Davide Rocchesso Introduction to Sound Processing [book]
ISBN: 88-901126-1-1

[21] Available online: https://indietips.com/audio-format-guide/

[22] Available online: https://jmlr.csail.mit.edu/papers/volume10/king09a/king09a.pdf
Voice recognition algorithm

[23] Available online : https://medium.com/@kamil2000budaqov/discrete-fourier-transform-75d9a9f37821

[24] Introduction to sound processing book

[25] Available online: https://www.developer.com/guides/introduction-to-the-java-sound-api/

[26] Available online: https://jenkov.com/tutorials/java/modules.html

[27] Available online: https://howtodoinjava.com/java9/java-9-modules-tutorial/

[28] Available online: https://dev.java/learn/modules/intro/

[29] Available online: https://www.geeksforgeeks.org/jpms-java-platform-module-system/

[30] Available online: https://blog.mozilla.org/en/mozilla/news/sharing-our-common-voices-mozilla-releases-the-largest-to-date-public-domain-transcribed-voice-dataset/

[31] Available online: https://medium.com/@224vinod/java-machine-learning-libraries-a-guide-to-choosing-the-right-one-cd8229ce0f4a

[32] Support Vector Machines: Theory and Applications[book] 2005
ISBN: 978-3-540-32384-6

[33] https://scikit-learn.org/stable/modules/svm.html

[34] https://www.geeksforgeeks.org/auc-roc-curve/

[35] https://www.mathworks.com/help/deeplearning/ug/compare-deep-learning-models-using-ROC-curves.html

[36] Mun Articles https://medium.com/@mun.articles/svms-in-practice-applications-and-use-cases-for-machine-learnings-most-effective-model-2ae25f4207ef

[37] Support Vector Machines: Theory and Applications Springer Berlin Heidelberg New York
ISBN-10 3-540-24388-7 ISBN-13 978-3-540-24388-5

[38] Poggio, T. and Girosi, F. (1990). A theory of networks for approximation and learning. Proc. ISBN-13 978-3-540-24388-5.

[39] Available online: https://www.qualtrics.com/experience-management/research/determine-sample-size/

[40] https://www.investopedia.com/terms/s/sample.asp

[41] Xu, Min and Duan, Ling-Yu and Cai, Jianfei and Chia, Liang-Tien and Xu, Changsheng and Tian, Qi. (2004). HMM-Based Audio Keyword Generation. 3333. 566-574. 10.1007/978-3-540-30543-971.

[42] Martinez, Jorge and Perez-Meana, Hector and Escamilla-Hernandez, Enrique and Suzuki, Masahisa. (2012). Speaker recognition using Mel Frequency Cepstral Coefficients (MFCC) and Vector quantization (VQ) techniques. 10.1109/CONIELECOMP.2012.6189918.

[43] Ali, Shalbbya and Tanweer, Safdar and Khalid, Syed and Rao, Naseem. (2021). Mel Frequency Cepstral Coefficient: A Review. 10.4108/eai.27-2-2020.2303173.

[44] Gupta, Shikha and Jaafar, Jafreezal and Wan Ahmad, Wan Fatimah and Bansal, Arpit. (2013). Feature Extraction Using Mfcc. Signal and Image Processing : An International Journal. 4. 101-108. 10.5121/sipij.2013.4408.

[45] Janvale, Ganesh and Deshmukh, Ratnadeep and Gambhire, Supriya. (2010). Speech Feature Extraction Using Mel-Frequency Cepstral Coefficient (MFCC).

[46] Available online: https://oden.io/glossary/model-training/

[47] Available online: https://www.databricks.com/glossary/machine-learning-models

[48] Available online: https://www.coursera.org/articles/what-is-cross-validation-in-machine-learning

[49] Available online: https://www.datacamp.com/tutorial/normalization-in-machine-learning

[50] Available online: https://www.rudderstack.com/learn/data-analytics/machine-learning-model-training/

[51] Alim, Sabur and Alang Md Rashid, Nahrul Khair. (2018). Some Commonly Used Speech Feature Extraction Algorithms. 10.5772/intechopen.80419.

[52] Available online: https://mit-crpg.github.io/OpenMOC/devguide/swig.html

[53] Beazley, David. (2003). Automated scientific software scripting with SWIG. Future Generation Comp. Syst.. 19. 599-609. 10.1016/S0167-739X(02)00171-1.

[54] Available online: https://uconntact.uconn.edu/event/6442682

[55] Abeel, Thomas and Van de Peer, Yves and Saeys, Yvan. (2009). Java-ML: a Machine Learning Library. Journal of Machine Learning Research. 10. 931-934. 10.1145/1577069.1577103.

[56] Frank, Eibe and Hall, Mark and Holmes, Geoffrey and Kirkby, Richard and Pfahringer, Bernhard and Witten, Ian and Trigg, Len. (2010). Weka-A Machine Learning Workbench for Data Mining. 10.1007/978-0-387-09823-466.

[57] Svetoslav Zhelev, Anna Rozeva; Data analytics and machine learning with Java. AIP Conf. Proc. 10 December 2018; 2048 (1): 060020.

[58] Beazley, David. (2003). Automated scientific software scripting with SWIG. Future Generation Comp. Syst.. 19. 599-609. 10.1016/S0167-739X(02)00171-1.

[59] B, Jagan and Babu.N, Ramesh. (2014). Speech recognition using MFCC and DTW. 10.1109/ICAEE.2014.6838564.

[60] Tripathy, B.K. and Anveshrithaa, S and Ghela, Shrusti. (2021). Principal Component Analysis (PCA). 10.1201/9781003190554-2.

[61] Ahmad, Noor and Nassif, Ali. (2022). Dimensionality Reduction: Challenges and Solutions. ITM Web of Conferences. 43. 01017. 10.1051/itmconf/20224301017.

[62] Fawcett, Tom. (2006). Introduction to ROC analysis. Pattern Recognition Letters. 27. 861-874. 10.1016/j.patrec.2005.10.010.

[63] Jha,R.K.,Bag,S.,Koley,D.,Bojja,G.R.,and Barman,S. (2023). An appropriate and cost-effective hospital recommender system for a patient of rural area using deep reinforcement learning. Intelligent Systems with Applications, 18, 200218.

[64] Yang, Shengping and Berdine, Gilbert. (2017). The receiver operating characteristic (ROC) curve. The Southwest Respiratory and Critical Care Chronicles. 5. 34. 10.12746/swrccc.v5i19.391.

[65] Gupta, Ruchi and Sharma, Anupama and Alam, Tanweer. (2024). Building Predictive Models with Machine Learning. 10.1007/978-981-97-0448-43.

[66] Alves de Souza, Vinícius and Rossi, Rafael and Batista, Gustavo and Rezende, Solange. (2017). Unsupervised active learning techniques for labeling training sets: An experimental evaluation on sequential data. Intelligent Data Analysis. 21. ISBN -061-1095. 10.3233/IDA-163075.

[67] Brown, Williams and Noah, Asher and John, Ada. (2024). Optimizing Hyperparameters in Machine Learning Models: Techniques and Best Practices. ISBN-13. 978-1484265789

[68] Swaminathan, Sathyanarayanan and Tantri, B Roopashri. (2024). Confusion Matrix-Based Performance Evaluation Metrics. African Journal of Biomedical Research. 27. 4023-4031. 10.53555/AJBR.v27i4S.4345. ISBN: 1119-5096

[69] Tripathy, B.K. and Anveshrithaa, S and Ghela, Shrusti. (2021). Principal Component Analysis (PCA). 10.1201/ ISBN- 9781003190554-2.

[70] Islam, Saiful and Islam, Md. (2019). A Comparative Study on Discrete Fourier Transformation for Digital Signal Analysis. 01. 18-26.

[71] Schatzman, James. (1970). Accuracy Of The Discrete Fourier Transform And The Fast Fourier Transform. SIAM Journal on Scientific Computing. 17. 10.1137/ ISBN-1064-827593247023.