UNIVERSITY OF THE PELOPONNESE &

NCSR "DEMOCRITOS"

MSC PROGRAMME IN DATA SCIENCE

Instrument playing technique recognition

by

Paraskevoudis Konstantinos

243213133546

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Giannakopoulos Theodoros

Athens, January 2020

UNIVERSITY OF THE PELOPONNESE &

NCSR “DEMOCRITOS”

MSC PROGRAMME IN DATA SCIENCE

Instrument playing technique recognition

by

Paraskevoudis Konstantinos
243213133546

A thesis submitted in partial fulfillment
of the requirements for the MSc
in Data Science

Supervisor: Giannakopoulos Theodoros

Approved by the examination committee on January, 2020.

Giannakopoulos Theodoros	Klampanos Iraklis	Platis Nikolaos
………………..	………………..	………………..

Athens, January 2020

UNIVERSITY OF THE PELOPONNESE &

NCSR "DEMOCRITOS"

MSC PROGRAMME IN DATA SCIENCE

Declaration of Authorship

(1) I declare that this thesis has been composed solely by myself and that it has not been submitted, in whole or in part, in any previous application for a degree. Except where states otherwise by reference or acknowledgment, the work presented is entirely my own.

(2) I confirm that this thesis presented for the degree of Bachelor of Science in Informatics and Telecommunications, has

(i) been composed entirely by myself
(ii) been solely the result of my own work
(iii) not been submitted for any other degree or professional qualification

(3) I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or processional qualification except as specified.

Konstantinos Paraskevoudis

Athens, January 2020

ABSTRACT

Instrument Playing Technique recognition is a growing research field of Music Information Retrieval. Regarding stringed instruments, an instrument playing technique can be defined as any particular motive of the instrument players' fingers applied on either the strings of the neck or on the strings of the body or sound hole of the instrument. In this work, the automated recognition of instrument playing techniques in solo recordings of the Greek stringed instrument bouzouki is examined. Towards this, a Dataset comprising of 336 recordings and 5 different playing techniques (slurring, trembling in one string, trembling in two strings, chord play, 2 strings play) is generated. The signal of each recording is firstly broken into short-term frames and the audio features (34) for each frame are extracted. In addition, the mean and standard deviation are extracted for each mid-term segment by applying them per sequence of short-term feature sequence for each segment. In total, there are 68 audio features for each recording. 8 different combinations of short and mid term windows and sizes have been selected in order to extract features and train models. Five Machine Learning models (SVM, K-NN, Gradient Boosting, Extra Trees, Random Forest) are trained on the extracted features, resulting in a total of 40 trained models. The trained models are evaluated in a generated test set comprising of 5 popular songs. Besides the five techniques, a 'None' technique is introduced, as a segment can either have a technique or be a simple melody (absence of technique). In order to address this, we experiment with different confidence levels for the classifications of the models. If the confidence (probability of classification) made by a model for a specific segment is lower than the selected confidence, then the 'None' class is assigned to this segment. The best performing model was SVM with mid-term window 0.9, mid-term step of 0.6, short-term window of 0.06 and short-term step of 0.02. This model achieved the highest F1 score (0.5880) at Confidence value 0.4. Recall drops and precision rises as the confidence limit rises. An open-source standalone python script was developed which classifies each segment of a given solo bouzouki instrument recording as of the playing technique using the best 6 (pre-trained) models of this Thesis.

Keywords: Music Information Retrieval, Instrument Playing Technique recognition, Machine Learning, Audio Features Extraction, Signal Processing

LIST OF TABLES

LIST OF FIGURES

Contents

1. INTRODUCTION

The current Thesis investigates the application of Machine Learning models in the task of the automated recognition of instrument playing techniques of the Greek stringed instrument bouzouki. An instrument playing technique is defined as any extended technique that is used by an instrument player on either the strings of the neck (thin part of the instrument, usually used with bare hands) or on the strings of the body or the sound hole of the instrument. The examination of automatic recognition of playing techniques of other instruments can be of great importance, as instrument learning draws off from classical methods such as conservatoires and teacher help. New beginners are now using the web more and more (online tutorials etc.) in order to study and learn how to play an instrument. As a result, the need for automated tools that can help the music community process music material grows.

The goal of this task is to develop a Machine Learning model which identifies a playing technique in a given recording. In total, there are 5 general playing techniques. The techniques in the table below are general categories, which can have more sub-techniques. However, they generalize well all the instrument's techniques. Of course, besides these 5 categories, the category 'None' is introduced for characterizing all parts of a song with absence of any technique. These parts are often called simple melodies or normal string strikes.

Techniques
Trembling in one string (tremoulo 1)
Trembling in two strings (tremoulo 2)
Slurring (kalopismos)
Two strings play (2 xordes)
Chord (sigxordia)

Table 1: General techniques of Bouzouki

The current work is a subject to Music Information Retrieval (MIR). MIR is the interdisciplinary field of Information Technology and Music Science, which applies algorithms on musical Data in order to extract valuable and meaningful information out of it [1,2]. The continuous generation of large amounts of digital music Data highlighted the need of methodologies which can efficiently exploit it. Taking advantage of great advances in Machine Learning and Deep Learning, researchers from various fields such as Music Science, Engineering and Computer Science have cooperatively proceeded to the development of several MIR applications [2].

2. RELATED WORK

Audio Features Extraction

In order to make use of musical Data through application of Machine Learning or Deep Learning, the music signal has to be transformed into an algorithm-friendly structure. The process of analyzing a recorded musical signal and extracting important audio features of it is called Audio Features Extraction or Audio Analysis [3]. Several methodologies are proposed throughout the last years, in order to extract short and mid term audio features of a raw signal [3-5]. A key factor in analyzing a raw signal is partition of the signal into small pieces and then the calculation of short-term feature vectors for each of these pieces, in order to take advantage of the stability of signal features in these small parts [6]. Such features can be based on Fast Fourier Transform (FFT), MPEG Filterbank Analysis, Linear Predictive Coding (LPC) and Mel-Frequency cepstrum coefficients (MFCC) [3-6]. A common methodology is to compute the mean and variance of these features over a larger time window [3]. Extracting audio features is a prerequisite in any MIR task and therefore results may differ based on the type and the number of features extracted [7]. Audio features can then be used for classification or regression tasks, such as classification of music genre and emotion classification [7]. Audio Features Extraction over large streams or batches of music Data can be computationally inefficient. This issue has also been addressed, as existing methodologies also propose a distributed process for extracting audio features in parallel, thus saving time and efforts significantly [8].

Music Genre Classification

One of the most researched applications of MIR is the music genre classification. A music genre is an abstract term which is assigned to a music song as a label by humans, based on the instruments used, the rhythm and the audio content [9]. The audio features play a key role in this task. Training Machine Learning models on a music Dataset’s audio features with human labeled music genres and validating them in unlabeled songs results in accuracies comparable to humans classification [9]. Beyond Machine Learning, Deep Learning has also been applied for music genre classification. Experiments showed that convolutional neural networks (CNNs) are a good performing alternative for automatic feature extraction and music genre classification [10]. A combination of convolutional neural networks (CNNs) for extracting short term audio features and recurrent neural networks (RNNs) for maintaining temporal information out of the extracted features perform well on music genre classification [11]. Another combination of convolutional neural networks (CNNs) and Robust Local Binary Pattern achieved an accuracy of 92% in the Latin Music Database (LMD) [12,13]. Finally, music genre classification can also be approached with combination of textual features such as lyrics and audio features [14].

Audio Fingerprinting

Audio fingerprinting is another highly researched area of MIR. Each audio has its own content-based signature, called the audio fingerprint, which is a unique compact low-level representation of an audio signal [15,16]. Therefore, any song can be identified by its audio fingerprint. Audio fingerprinting techniques rely on a Database with known songs and their respective fingerprints and metadata. The fingerprint of the unknown song (to be identified) is used as a query in the Database. With use of similarity measures, the most similar song (in terms of the fingerprint) is returned along with its metadata [16].

Music recognition and audio event detection

Music recognition and audio event detection has also captured MIR's community interest. Trained Machine Learning classifiers can classify audio features as either music or all other types of audio in video streams such as movies, while their results achieved high accuracies in the detection of music even in extreme cases, such as mixes of multiple types of audio in the recording [17]. More advanced approaches with Fisher linear semi Discriminant Analysis and Hidden Markov Models can achieve even better results in detecting music parts [18]. Such applications are found for music detection in movies, the Television industry and streaming platforms and are very helpful for content review and digital media rights [17-19]. Other approaches related to music recognition and complex audio event detection propose content-based segmentation and classification of audio or video files. Via this, the classification of the segments into labels of interest such as violent content is possible [20].

Speech vs Music discrimination

In addition to binary MIR classification tasks, Speech/Music discrimination in audio recordings can help in the characterization of audio content. A Speech vs Music discrimination methodology was applied on various radio recordings achieving an accuracy of 96% [21]. Specifically, the proposed technique consists of 3 stages. In the first stage, the audio features of the signal are extracted and predictions are made only in the segments with either high music or high speech precision. The remaining unclassified parts are given as input to the second stage. Here, the proposed algorithm seeks the sequence of segments and respective class labels that give the maximum product of posterior class probabilities for all the initial signal using a Bayesian Network in order to estimate the posterior probabilities [21,22]. Finally, possible errors at the boundaries of generated segments are corrected. In order to limit the complexity and computational needs of such an approach and its application on real time data, a Fisher Linear Discriminant Analysis methodology with use of only one extracted audio feature was proposed and demonstrated to give good results [23]. Another Speech vs Music discrimination approach was validated on a large Dataset, giving an accuracy of 97% in segmentation and an accuracy of 95% in classification (speech vs music) [24]. Deep Learning models are also applied and proposed for classifying an audio segment as either music or speech. Specifically, convolutional neural networks (CNNs) were used in order to learn visual feature dependencies from images such as the audio spectrograms [25]. It was shown, that these models managed to learn meaningful deep structures and correlations and outperformed simpler Machine Learning models. Finally, recurrent neural networks (RNNs) have also been used for discriminating speech vs music [26].

Music recommendation systems

Music recommendation systems involve MIR techniques, in order to extract audio features and recommend content related and similar music to users according to their preferences and musical habits. Current music recommendation approaches can be divided several categories: collaborative filtering, content based, hybrid, statistical [27, 28]. Collaborative fingering techniques rely on a large collection of preferences of other users and propose automatic predictions based on them when analyzing a users' preferences [29]. In addition, content-based techniques treat the recommendation as a user specific classification task, where the goal is to recommend music similar to the music that a user listened to in the past or is currently listening to [30]. Such recommendations can also be hybrid and are not only based on user profiling and analysis of their history (Metadata) but also from audio related features (content-based recommendation) [27]. Deep Learning and especially convolutional neural networks (CNNs) play a key role in music recommendation systems based on only Metadata and can outperform traditional approaches [31]. Recommendation systems which make use of audio features have also been used in order to extract traditional audio features (such as rhythm, tonal strength and spectral characteristics) and recommend based on a high-level music similarity metric [32]. However, these features might not capture important relevant information [33]. In addition, acoustic-based music recommendation systems combining audio features with textual social media data can also be effective [34].

Music Synthesis

Music Synthesis is yet another field of research in MIR which has captured the interest of researchers especially during the last years and the growth of Deep Learning field [35]. Recurrent neural networks (RNNs) can capture valuable long-term dependencies and preserve past information in the signal, which are essential in such a sequential task, where the goal is to «recommend» something about the next sequence based on previous information [36]. Early neural network research on music generation used mostly recurrent neural networks (RNNs) and long short-term memory models (LSTMs) [37,38]. However, convolutional neural networks (CNNs) are also capable of generating realistic musical sequences (melodies) and therefore be used as a Generator [37,39]. In addition, this can be generalized to a Generative Adversarial Network (GAN) with use of a discriminator which learns the distributions of true melodies and tries to distinguish between fake and real in the generated melodies from the Generator [37].

Music emotion classification

Audio features extracted from raw audio signals can also be used for emotion classification. Support Vector Machines (SVMs) can be trained in a multiple binary classification approach on rhythmic contents, timbres, textures and other audio features and then classify a musical song as of its emotion [40]. Deep Learning models are also performing well in detecting emotions in music [41, 42]. Typically, there are two variables of interest in such tasks that need to be predict before classifying an audio signal to an emotion: arousal and valence. These two parameters can be either have a continuous value between -1 and 1, which is a regression task or a categorical value (positive, negative) which is a classification task. In both cases, each song will have its own tuple of valence and arousal, which can be placed in a coordinate system with arousal being the y-axis and valence being the x-axis [42]. Points in the first quadrant (positive arousal and valence) are related to happy emotions, whereas points in the second quadrant (positive arousal and negative valence) are related to angry or nervous emotions. Points in the third quadrant (negative arousal and negative valence) refer to sad emotions and points in the fourth quadrant (negative arousal and positive valence) refer to relaxed and peaceful emotions. The above categorization is known as Russel's circumplex model of emotions [43]. Via this, the emotion recognition task can be either treated as a regression task (arousal and valence continuous value between -1 and 1) or as classification task (high/low arousal and valence or other categorical values for emotion). Support Vector Machines (SVMs), which treat music samples as single point in the arousal-valence system combined with Principal Components Analysis (PCA) to reduce correlations between arousal and valence and advanced feature engineering techniques perform well [44]. Support Vector Machines (SVMs) are also used for respective classification tasks (categorical values in arousal and valence) and can be applied on recommendation systems based on the emotion [45]. Such approaches can also be used for tracking emotion variations in a song [46].

Software and libraries

As MIR research grew rapidly in the last years, various related software, open-source libraries and frameworks were developed in order to support further research, commercial products and users' or musicians' needs. Marsyas is one of the earliest open source frameworks, which supports audio processing techniques such as feature extraction and Machine Learning [47]. PyAudioAnalysis is another open-source library, written in Python, which supports audio features extraction both in short and mid term basis [48]. In addition, pyAudioAnalysis is a helpful tool for classification and regression on musical Data, both for black box use and manual tuning [48]. Specifically, pre-trained models are offered for various popular MIR tasks, such as speech-music classification, music genre classification and audio event detection [48]. Finally, audio segmentation and audio visualization are also some easy-to-use tools offered by pyAudioAnalysis. YAAFE, also written in python, offers audio features extraction tools [49]. Another python library for signal processing and audio analysis is LibROSA, which is focused on audio features extraction [50]. Essentia is another open-source library, written in C++, which is focused mainly on audio features extraction [51]. The MATLAB audio analysis library offers MATLAB solutions for general audio handling, audio processing, audio feature extraction, classification and segmentation of audio [52]. It is also specialized for MIR tasks, as users can implement audio content characterization with segmentation and classification techniques and furthermore experiment on music thumbnailing, music content visualization and tempo induction [52]. Regarding MATLAB, MIRtoolbox also offers functions related to the extraction of music audio features and statistical analysis, segmentation and clustering tools [53]. Graphical user interface (GUI) applications such as jAudio and OpenSmile offer user-friendly functions for audio feature extraction, general audio handling, statistical analysis and Machine Learning [54, 55].

Music Instrument Recognition

Music Instrument Recognition is a subfield of MIR, which applies statistical and Machine Learning techniques in order to identify separate instruments involved in a musical composition. A pattern-recognition technique on acoustic features was applied on a Dataset with 15 different orchestral instruments being able to achieve a 70% accuracy on the identification of individual instruments [56]. An application of linear prediction analysis on a database of more than 5000 solo tunes from 29 different western orchestral instruments achieved an accuracy of 35% for individual instrument recognition and 77% for instrument families, when trained in mel-frequency cepstral coefficients and other extracted audio features [57]. A pitch independent musical instrument recognition approach on spectral and temporal properties of 1498 samples with 30 different orchestral instruments correctly identified 94% of test set instances as of the instrument family and 80% of test set instances as of the individual instrument [58]. A study on a Dataset with 10 different instruments and training on more than 150 audio features showed that higher accuracies can be achieved when splitting the feature set into two pairwise optimized subsets and training with Support Vector Machines (SVMs) [59]. Regarding feature engineering in such tasks, another study proved that transforming the audio features (mel-frequency cepstral coefficients and their first derivatives) to a feature set with maximal statistical independence using independent component analysis (ICA) can improve instrument recognition accuracy by 9% with use of hidden Markov models (HMM) [60]. Deep Learning models have also been examined for instrument recognition tasks. A simple feedforward Neural Network correctly identified 97% of test instances in a test set including 4 instruments (piano, marimba, accordion, guitar), which represent the major music instrument families [61]. However, slightly better accuracies were achieved using a k-nearest neighbors (KNN) algorithm in the same Dataset [61, 62].

Instrument Playing Technique recognition/classification

The current Thesis deals with the identification/recognition of instrument playing techniques in audio recordings with solo playing of the Greek instrument Bouzouki. The playing technique of instruments in a song is a field which has a lot of interest for professional instrument players or amateur hobbyists. A lot of people turn their focus on understanding playing patterns or even explicit techniques of famous instrument players. However, the automatic detection or identification of instrumental playing technique is underdeveloped [63]. Beyond signal processing and MIR, the automatic identification of instrument playing technique is an important research field also for biomechanics and gestural interfaces [64]. An evaluation of machine learning models for this task on the Sound Online Dataset (comprising of 16 musical instruments playing more than 25000 isolated notes in total) achieved a precision of 99.7% for individual instrument recognition and 61% for instrument playing technique recognition [63, 65]. Specifically, k-nearest neighbours algorithm (kNN) with an euclidean distance as metric was applied on the mel-frequency cepstral coefficients of audio recordings of 143 different instrument playing techniques of 16 individual instruments [63]. Current work on instrument playing technique identification or classification focuses piano, guitar, violin, percussion, erhu and flute [63, 66-70]. In this work, an instrument playing technique of a stringed instrument (bouzouki) is proposed. Many playing techniques are technically similar to techniques of the electrical guitar. A guitar specific playing technique recognition validated on a dataset of 42 electric guitar solo tracks (without accompaniment), achieved a 74% accuracy on identifying 5 main electrical guitar playing techniques (bend, slide, vibrato, hammer-on and pull offs) [68]. Unlike previous work, the aforementioned approach does not focus on identifications based on single notes. Instead, the recognition of the playing technique is achieved for solo instrument recordings and has two stages [68]. Firstly, the audio recording is processed and candidate segments are identified via analysis of the melody contour. Finally, a pre-trained Support Vector Machine (SVM) classifier is used to classify the playing technique of the identified candidates of the first step based on extracted timbre and pitch features [68]. Another study on 6 electrical guitar playing techniques correlated individual audio features such as Spectral Flux and Amplitude with each playing technique by applying correlation analysis [71]. In addition to actual techniques (slide, bend, hammer on, pull off, harmonic muting and palm muting), this approach also introduced another two categories (normal and mute) in order to not only cover playing technique recognition but also parts of the recording where there is no technique applied [71]. Classification techniques based on the magnitude spectrum, cepstrum and phase derivatives such as instantaneous frequency deviation achieved an average F1 score of 71.7% for 7 different playing techniques of the electrical guitar

[72]. Furthermore, it was shown that cepstral and phase features are crucial in the discrimination between more similar techniques such as pull-offs, hammer-on and bend [72]. Regarding stringed instruments and besides the electrical guitar, instrument playing technique recognition has also been studied for the electrical bass guitar. Respective classification approaches achieved a 93.25% classification accuracy for plucking techniques [73].

3. DESCRIPTION OF THE PROBLEM

The 5 general techniques (Table 1) chosen, were selected by 2 professional bouzouki players and 3 amateur players as the 5 categories which can generalize and include all existing instrument techniques. The goal was to minimize the classes of this problem. After finalization, the techniques are:

1. Trembling in one string
2. Trembling in two strings
3. Slurring
4. Two strings play
5. Chord

Trembling in one string refers to a bowed string instrument playing technique, in which the player strikes the string alternating downward and upward in a continuous fashion. In high speeds, the technique is also called “tremolo picking” or “double picking” [74, 75]. This technique usually lasts more than one second. In other words, this technique is used to cover longer parts of a song or enrich existing simple plays through notes. In general, due to the need of consecutively striking one string downward and upward, trembling needs more than 0.5 seconds to be performed. A recorded Graphics Interchange Format animation (GIF) of Trembling in one string can be seen by clicking here. Trembling in one string is a popular technique also among other bowed string instruments such as the electric guitar and the banjo.

Trembling in two strings is the exact same technique as trembling in one string but this time the continuous downward and upward strike is performed in two strings. It is a very common and popular technique in folk Greek music and is deployed mainly to cover parts of the songs where vocals exist in order to accompany the voice of the singer. As with trembling in one string, also here there is a significant amount of time (>0.5 seconds) in order to perform one complete trembling in 2 strings. Both tremblings can occur repeatedly one after another. In other words, a trembling can follow another one, which is usually the case. A recorded Graphics Interchange Format animation (GIF) of Trembling in two strings can be seen by clicking here.

Slurring is general technique in which a player brings sharply and rapidly a finger down on the fingerboard behind a fret, causing a note to sound [76]. Slurring can have sub-categories such as Legato and Hammer-ons. This technique is mainly used to enrich parts of a song with more notes, without interfering or changing the melody or the rhythm. Slurring is a fastly performed technique, which requires high familiarity with the instrument. Therefore, this technique can be completely performed just in less than 0.5 seconds. Of course, slurring can also be consecutive, which means that many complete performances of it occur one after another even with moving between different notes and places in the instrument's neck. Slurring is also popular among the electrical and classical guitar and the banjo. A recorded Graphics Interchange Format animation (GIF) of Slurring can be seen by clicking here.

Two strings play is the playing (not trembling) of two notes together. In a bowed string instrument such as bouzouki, in order to achieve this, one has to play together two strings. Two strings play is used widely in folk greek music especially on specific genres, such as xasaposervika and rempetika. Two strings plays can replace simple one string melodies giving a richer sound. A recorded Graphics Interchange Format animation (GIF) of Two strings play can be seen by clicking here.

The technique Chord refers to set of notes played together. This technique is mostly used to accompany vocals or at the end of songs for closures. When played to accompany vocal, different Chords are usually played consecutively one after another. In the case of using Chords to end a song, usually a maximum of 3 different Chords are played consecutively. A recorded Graphics Interchange Format animation (GIF) of Chord can be seen by clicking here.

In addition to the above techniques, another class is introduced for describing all musical segments that do not belong to any of the aforementioned techniques. This class ("None") consists of simple playing through notes that do not have any of the techniques.

4. METHODOLOGY

i) The Dataset

Five Machine Learning models (SVM, K-NN, Random Forests, Gradient Boosting, Extra Trees) will be trained on a Dataset consisting of 336 recordings of length varying from 1 second to 4 seconds. The training samples were recorded by the same player and instrument using the Audacity software and under the exact same environment (microphone, room). In all cases, the noise was limited to low values. In addition, all audio files are mono.

The instrument's techniques here can be very fast, meaning techniques can last from some milliseconds, depending on the speed of the track. The training samples distribution among classes is presented in Table 2 and the total length of training material per class is shown in Figure 1. All recordings of the training set are solo tracks without accompaniment.

Technique	**Training Samples**
2 strings	69
chord	67
slurring	65
trembling 1 string	68
trembling 2 strings	67

Table 2: Training samples per technique

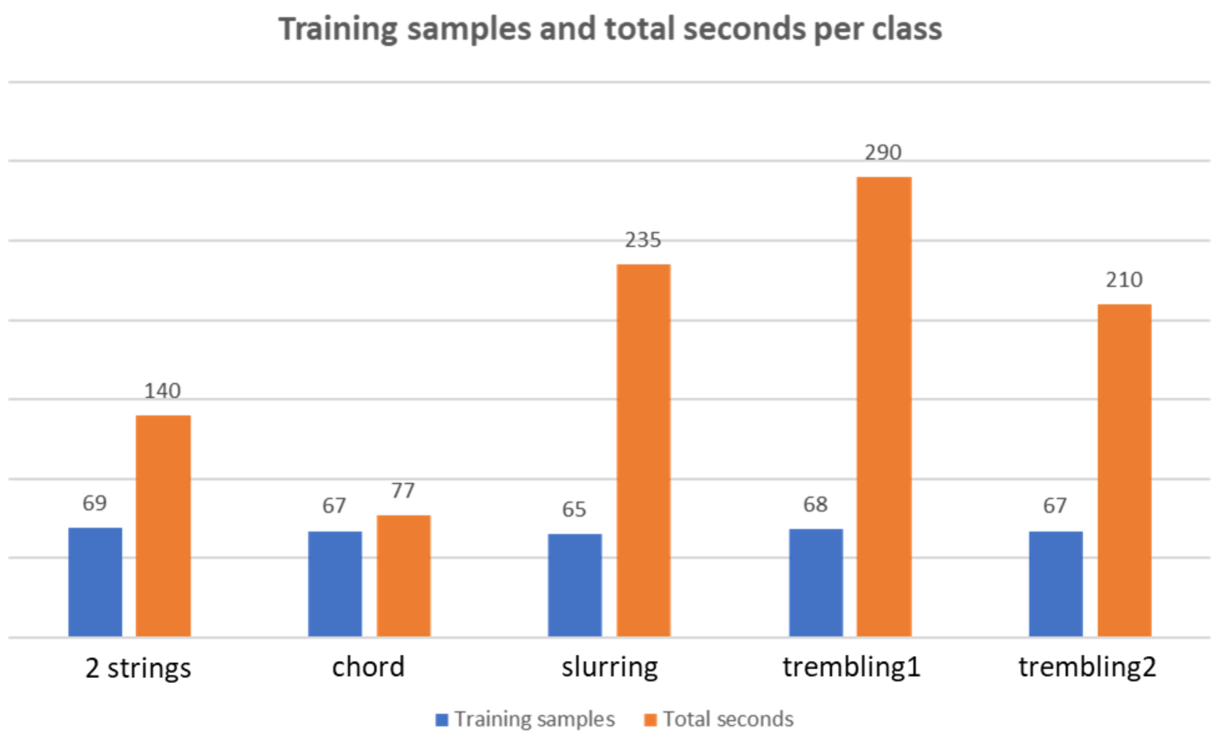


Figure 1: Training samples and total training length for each class

Regarding the distribution of total length (sec) of each class in the training set, there were in total 140 seconds of “Two strings play”, 77 seconds of “Chord”, 235 seconds of “Slurring”, 290 seconds of “Trembling in one string” and 210 seconds of “Trembling in 2 strings”. The high variance of these values is justifiable, as the technique with lowest length in training set (Chords) cover only small parts of any song (mainly at endings of songs or during parts of vocals). In other words, this technique appears much less than the other 4 (Slurring, Trembling in one string, Trembling in two strings, Two strings play). Therefore, there is a need to have much more training Data for the most appeared techniques in bouzouki solo tracks.

The test Dataset consists of 5 recordings, which samples of popular bouzouki songs (Table 3) [77-81]. They were selected due to their rich content in different techniques that one can play in them. The first releases of these songs were not that rich in techniques. Therefore, these songs were studied in order to evaluate the parts of them, in which technique could be introduced. After appropriate adjustment, the final samples were recorded with as many techniques as possible. All samples are solo tracks without accompaniment.

Test recording	Length (seconds)
Xoros sakaina + Minore tis avgis [77]	134
Mh mou ksanafigeis pia (intro) [78]	55
Minore tou teke [79]	59
Taksimi moraiti [80]	81
Prin to xarama [81]	92

Table 3: The test set and its length

For each one of these recordings, a ground truth file with labels for each segment is created, in order to compare our findings and compute evaluation metrics. The ground truth files are simple comma separated values files (CSV) and were created using the software Audacity. In these files, the start of each segment, the end of it and the corresponding class (Technique or None) are stored. The ground truths for each test recording are presented in Figures 2-11 for mid-term and short-term windows 0.5 and 0.01 respectively and mid-term and short-term steps 0.25 and 0.005 respectively. Ground truth values on the time (x) axis signify a None technique (absence of techniques) in the respective segment. The total length of each technique (class) in seconds in the test set is presented in Figure 12.

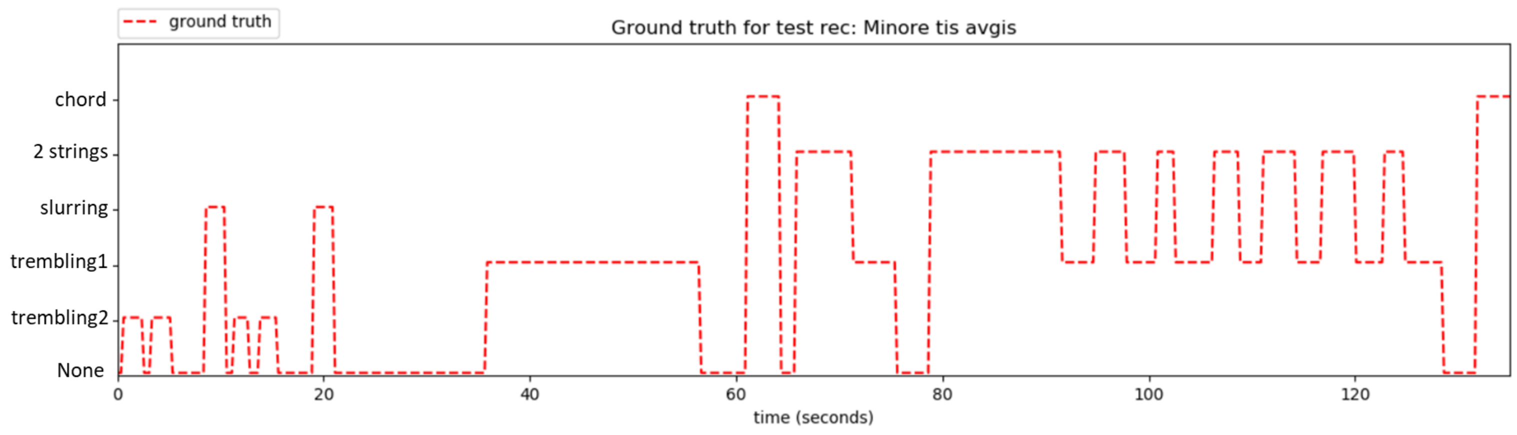


Figure 2: Ground truth for test rec “Minore tis avgis”

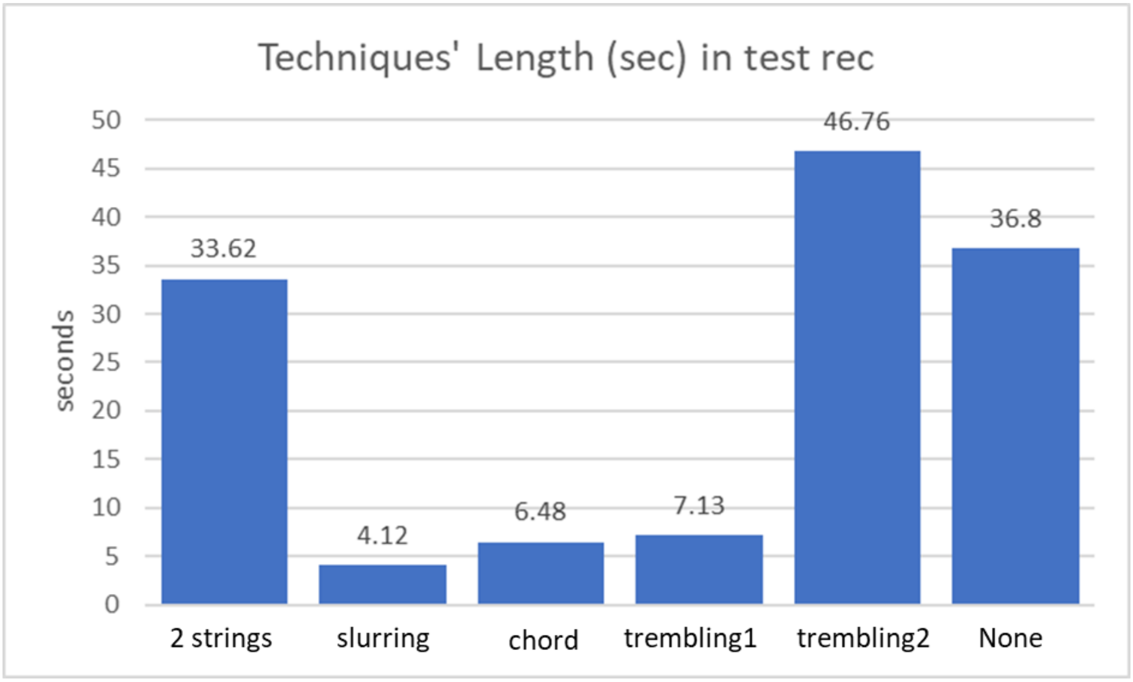


Figure 3: Length (seconds) of each class in the test recording “Minore tis avgis”

As presented in Figures 2-3 the first test recording, consists of 33.62 seconds of “Playing in two strings”, 4.12 seconds of “Slurring”, 6.48 seconds of “Chord play”, 7.13 seconds of “Trembling in one string” and 46.76 seconds of “Trembling in two strings”. The remaining part of the song (36.8 seconds) does not contain any Technique. The first recording of the original song can be listened by clicking here. The test recording can be found here.

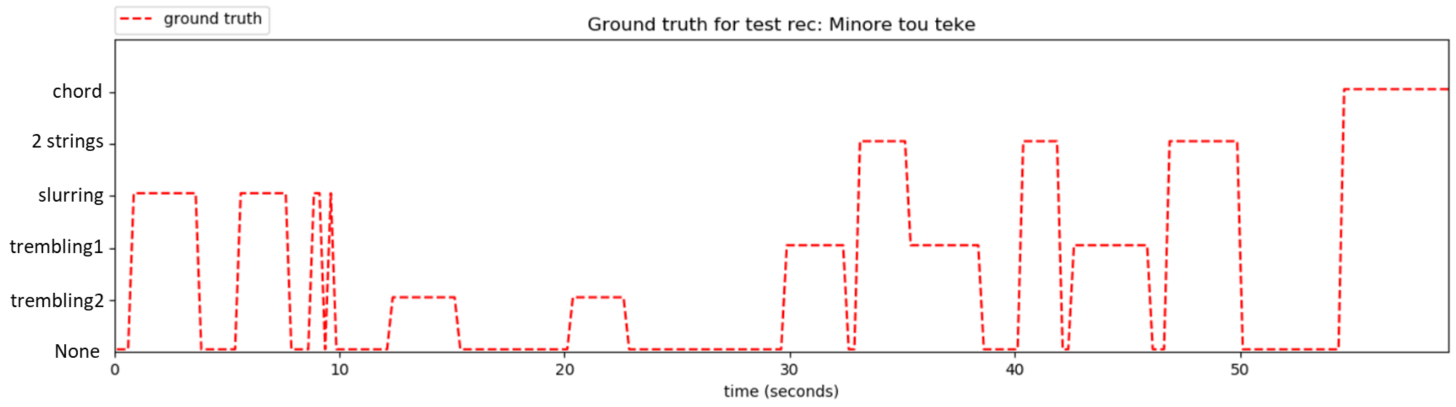


Figure 4: Ground truth for test rec “Minore tou teke”

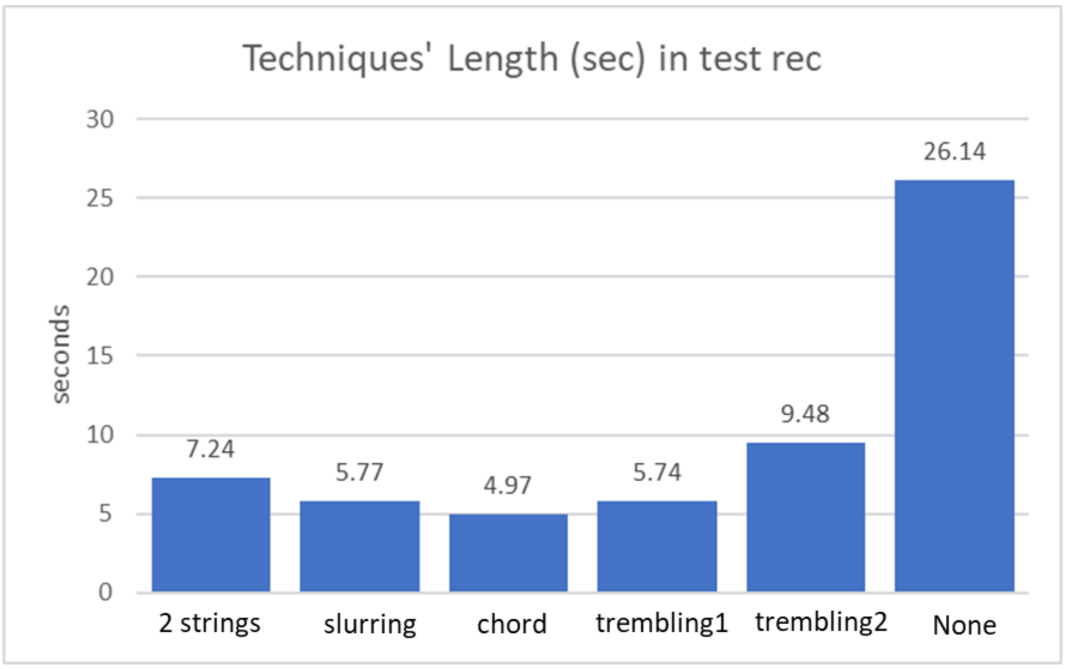


Figure 5: Length (seconds) of each class in the test recording “Minore tou teke”

As presented in Figures 4-5 the first test recording, consists of 7.24 seconds of “Playing in two strings”, 5.77 seconds of “Slurring”, 4.97 seconds of “Chord play”, 5.74 seconds of “Trembling in one string” and 9.48 seconds of “Trembling in two strings”. The remaining part of the song (26.14 seconds) does not contain any Technique. The first recording of the original song can be listened to here. The test recording can be found here.

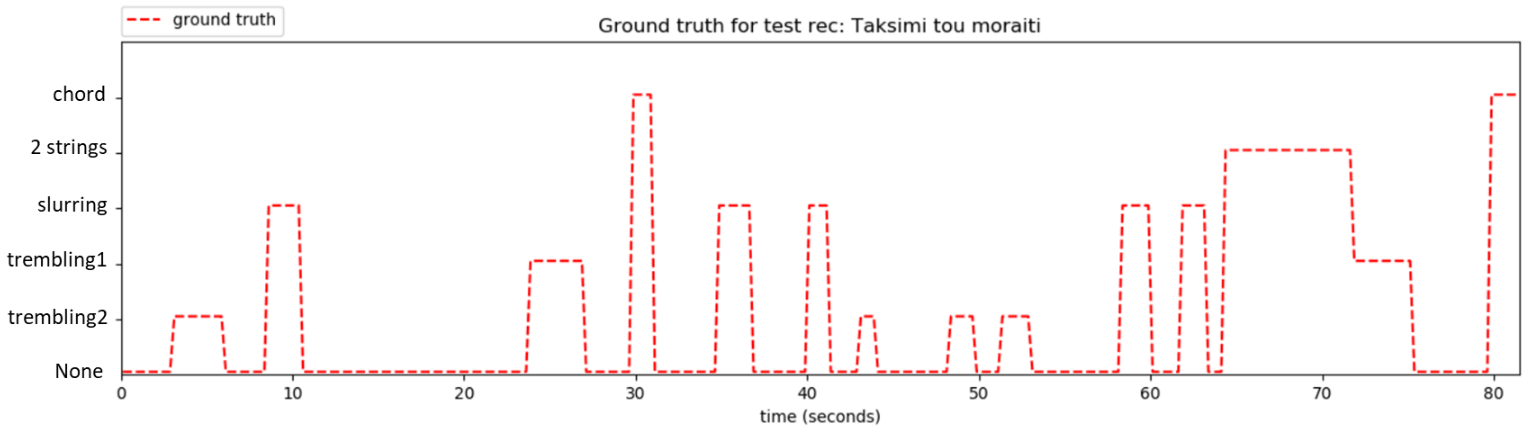


Figure 6: Ground truth for test rec "Taksimi tou moraiti"

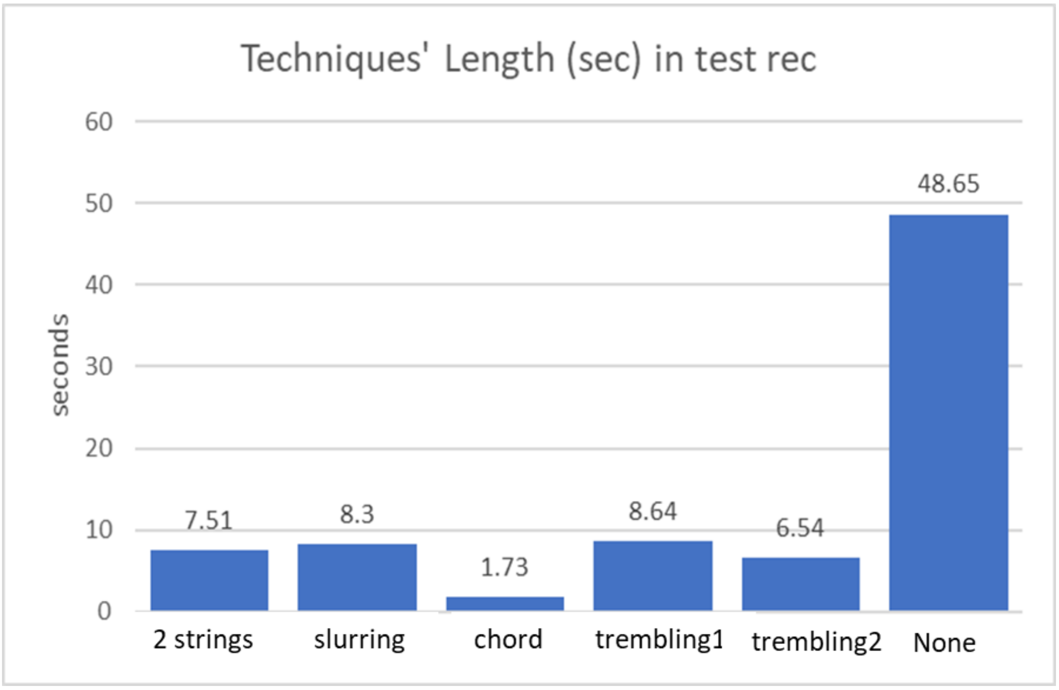


Figure 7: Length (seconds) of each class in the test recording "Taksimi tou moraiti"

As presented in Figures 6-7 the first test recording, consists of 7.51 seconds of "Playing in two strings", 8.3 seconds of "Slurring", 1.73 seconds of "Chord play", 8.64 seconds of "Trembling in one string" and 6.54 seconds of "Trembling in two strings". The remaining part of the song (48.65 seconds) does not contain any Technique. The first recording of the original song can be listened to here. The test recording can be found here.

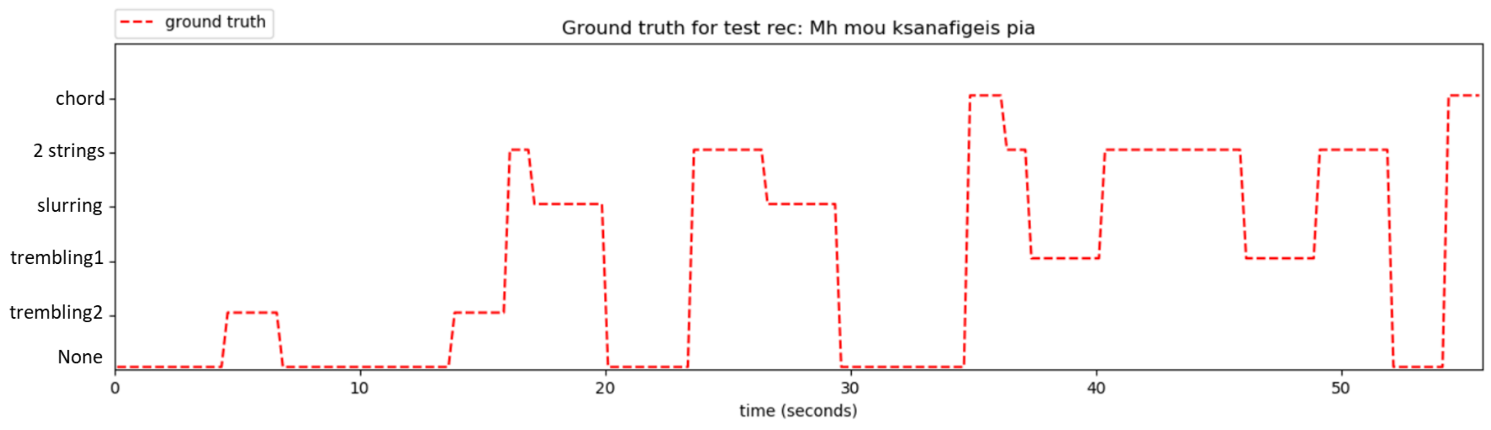


Figure 8: Ground truth for test rec "Mi mou ksanafigeis pia"

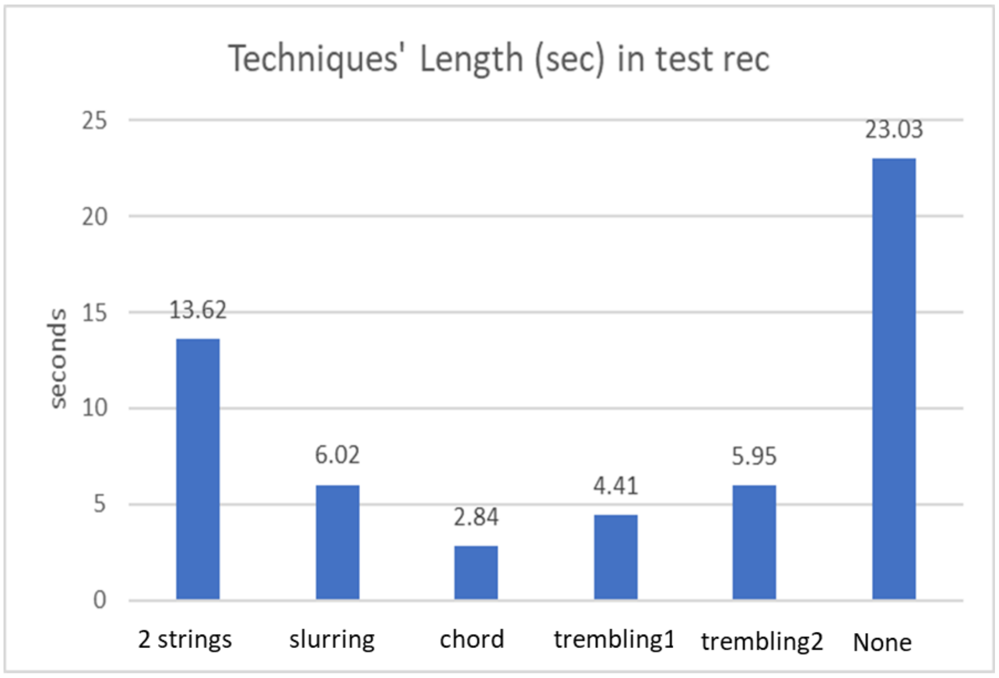


Figure 9: Length (seconds) of each class in the test recording "Mi mou ksanafigeis pia"

As presented in Figures 8-9 the first test recording, consists of 13.62 seconds of "Playing in two strings", 6.02 seconds of "Slurring", 2.84 seconds of "Chord play", 4.41 seconds of "Trembling in one string" and 5.95 seconds of "Trembling in two strings". The remaining part of the song (23.03 seconds) does not contain any Technique. The first recording of the original song can be listened to here. The test recording can be found here.

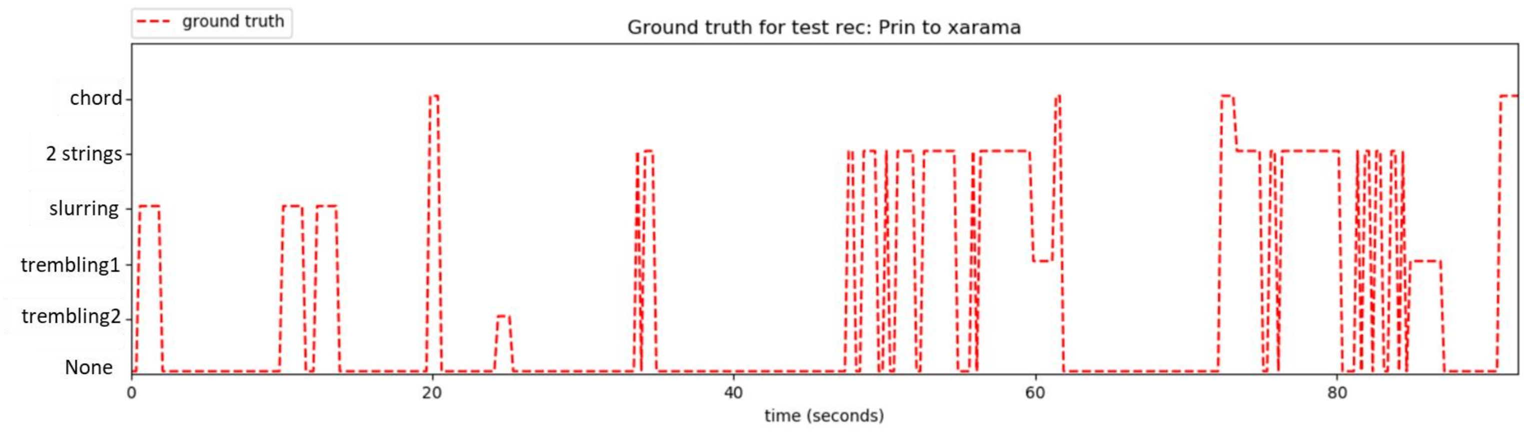


Figure 10: Ground truth for test rec “Print to xarama”

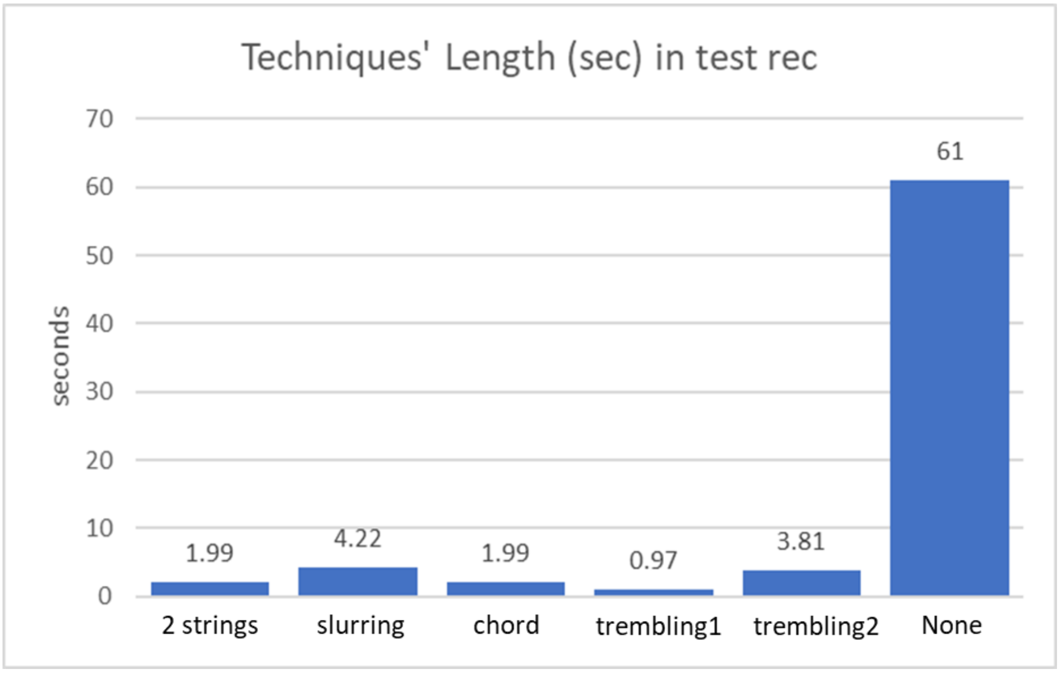


Figure 11: Length (seconds) of each class in the test recording “Prin to xarama”

As presented in Figures 10-11 the first test recording, consists of 1.99 seconds of “Playing in two strings”, 4.22 seconds of “Slurring”, 1.99 seconds of “Chord play”, 0.97 seconds of “Trembling in one string” and 3.81 seconds of “Trembling in two strings”. The remaining part of the song (61 seconds) does not contain any Technique. The first recording of the original song can be listened to here. The test recording can be found here.

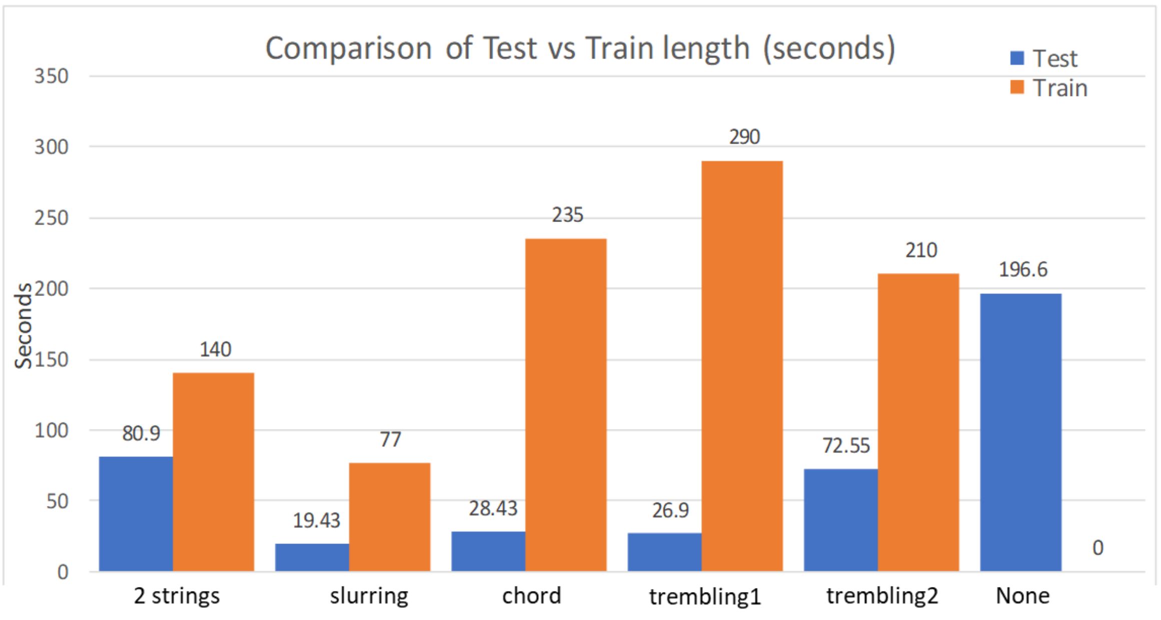


Figure 12: Length distribution per class in total test set and total training set

As presented in Figure 12, “2 strings play” is 35.45% (80.9 seconds) of the total test set and 14.71% (140 Seconds) of the training set. Slurring is 8.51% (19.43 seconds) of the total test set and 8.09% (77 seconds) of the total train set. Chords cover 12.46% (28.43 seconds) of the test set and 24.68 % (235 seconds) of the training set. “Trembling in one string” is 11.79 % (26.9 seconds) of the test set and 30.46% (290 seconds) of the training set. Trembling in two strings covers 31.79 % (72.55 seconds) of the test set and 22.06 % (210 seconds) of the training set.

Specific combinations mt-window/step and st-window/step have been selected in order to extract features and train the models. With bouzouki being a solo instrument, fast playing speeds and rapid switch between techniques or between a technique and a simple melody is very common. For instance, Slurring can occur in just 0.2 seconds depending on the speed of the player. In addition, changes between three techniques can occur in less than 2 seconds. Therefore, small short-term steps and windows are also selected. The models selected are SVM, K-NN, Extra Trees, Random Forests and Gradient Boosting. In total, we get 8 mt-window/step and st-window/step and 5 models, resulting in a total of 40 models. These are presented in Table 4.

mid term window	**mid term step**	**short term window**	**short term step**
0.5	0.25	0.01	0.005
0.6	0.2	0.015	0.005
0.8	0.1	0.02	0.005
0.8	0.2	0.04	0.01
0.8	0.04	0.02	0.01
0.9	0.6	0.006	0.02
1.6	1.6	0.06	0.06
2.0	0.8	0.1	0.04

Table 4: The windows and steps selected (mid and short-term)

ii) Audio Feature Extraction

In order to analyze the recordings, the signal of each recording has to be first broken into short-term (short) frames. These frames can be either overlapping or not, depending on the values of the frame window (length) and the step selected. For each of these frames (segments), the audio features are then extracted resulting in 34 audio features for each frame. Then, the signal is broken again into frame, but this time into mid-term segments. Then, again the short-term feature extraction is repeated. The feature sequence extracted from each mid-term frame is processed and the audio feature statistics are extracted. In other words, the statistics (mean, standard deviation) are extracted for each mid-term segment by applying them per sequence of short-term feature sequence for each segment. At the end of the above procedure, there are in total 68 features for each recording, which correspond to the mean and standard deviation of the audio features of Table 5, which are described below. In Figure X, the process of analyzing a signal and extracting the audio features is presented.

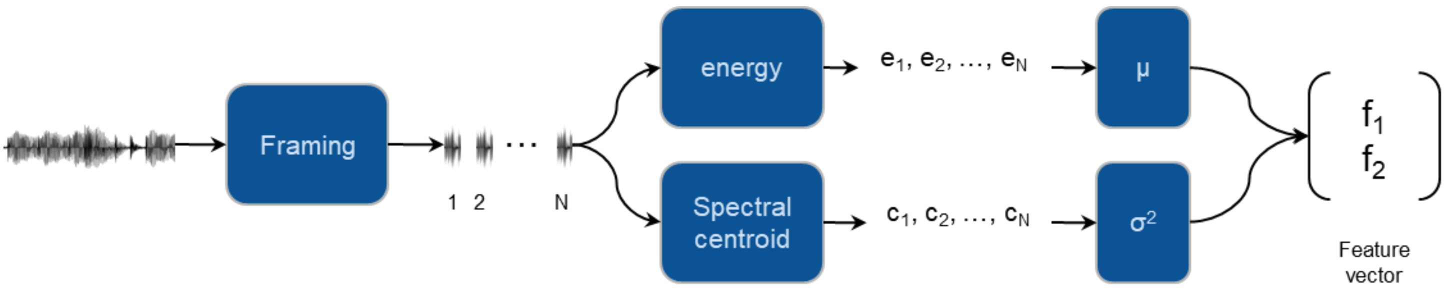


Figure 13: The process of analyzing a signal and extracting its audio features through segmentation [82]

The short term features extracted in the current work are presented in Table 5. In general, these 34 audio features provide valuable and adequate information in order to describe a signal and discriminate it among other [83]. The features can be divided in four broader categories:

Time-domain features

Energy is a measure to describe how fast (or slow) the sign changes. It can be described as the rate of sign-changes of the signal during the duration of a particular frame [83]. Mathematically, it can be computed by summing the squared signal values, normalized by the window length selected for the analysis. Given N frames of length L, the short-term energy of frame i is calculated by:

$$E(i) = \frac{\sum_{n=1}^{W_L} |x_i(n)|^2}{W_L}$$

, where xi(n) (n=1,…,WL) is the sequence of audio samples of the i-th frame.

Zero Crossing Rate (ZCR) is defined as the rate of sign-changes divided by the duration of the frame. ZCR is a measure to describe the amount of noise of a given signal. It is calculated by rate of sign changes during the frame:

$$Z(i) = \frac{1}{2W_L} \sum_{n=1}^{W_L} |sgn[x_i(n)] - sgn[x_i(n-1)]|$$

, where sgn is the function:

$$sgn[x_i(n)] = \begin{cases} 1, & x_i(n) \geq 0 \\ -1, & x_i(n) < 0 \end{cases}$$

Entropy of Energy captures and measures the abrupt changes of the Energy. First, K sub-frames are created by the original frame and the normalized sub-energies (Esubframe_k) of each K-frame are computed. The energy of each sub-frame is also computed and divided by the total Energy of the signal. The entropy of the normalized sub-energies (computed in previous step) is then calculated. The sub-frame energy values ej (j=1,…,K) are calculated by:

$$e_j = \frac{E_{subFrame_j}}{E_{shortFrame_i}}$$

, where:

$$E_{shortFrame_i} = \sum_{k=1}^{K} E_{subFrame_j}$$

The entropy H(i) of the sequence ej is then calculated by:

$$H(i) = -\sum_{j=1}^{K} e_j^2 \log(e_j^2)$$

Frequency Domain (Spectral) Features

The spectral centroid is the center of gravity of the spectrum. It belongs to the frequency domain features. It can be computed by:

$$C_i = \frac{\sum_{k=1}^{W_L}(k+1)X_i(k)}{\sum_{k=1}^{W_L}X_i(k)}$$

Spectral spread is the 2nd central moment of the spectrum and can be calculated by

$$S_i = \sqrt{\frac{\sum_{k=1}^{W_L}((k+1) - C_i)^2 X_i(k)}{\sum_{k=1}^{W_L} X_i(k)}}$$

Spectral entropy is of a signal is a measure of its spectral power distribution. It can be computed in a similar way as the entropy of energy, but for the frequency-domain. Specifically, the spectrum is divided into L sub-bands and the normalized sub-band energies (Ef) are computed.

$$n_f = \frac{E_f}{\sum_{f=0}^{L-1} E_f}, \text{ where } f = 0, \dots, L-1$$

Finally, the entropy is computed by:

$$H = -\sum_{f=0}^{L-1} n_f \, log_2(n_f)$$

Spectral flux is a is a measure of how quickly the power spectrum of a signal is changing, calculated by comparing the power spectrum for one frame against the power spectrum from the previous frame (Spectral change between two successive frame) [3, 83, 84]. At first, the ENi of each frame i and for a value k are computed. ENi(k) is the k-th normalized DFT coefficient at the i-th frame and can be computed by:

$$EN_i(k) = \frac{X_i(k)}{\sum_{l=1}^{W_L} X_i(l)}$$

Then, the flux is calculated by:

$$Fl_{(i,i-1)} = \sum_{k=1}^{W_L} (EN_i(k) - EN_{i-1}(k))^2$$

Spectral rolloff is the frequency below which a certain percentage of the magnitude distribution of the spectrum is concentrated.

Cepstral Domain

Mel-frequency cepstral coefficients (MFCCs) are coefficients that collectively make up an MFC [84]. They are calculated using a cepstral representation of the signal. Specifically, MFCCs are the discrete cosine transform coefficients on the mel-scaled log-power spectrum [3, 48, 82]. Firstly, the DFT is computed and the mel-scale filter bank is applied. The Mel scale relates perceived frequency, or pitch, of a pure tone to its actual measured frequency [83] and can be computed by:

$$f_w = 1127.01048 \log(\frac{f}{700} + 1)$$

Than, Ok is computed, which represents the power of the output of each of the above filters. Finally, the MFCCs are the the discrete cosine transform coefficients of the mel-scaled log-power spectrum:

$$c_m = \sum_{k=1}^{L} \left(\log \widetilde{O_k}\right) \cos[m\left(k - \frac{1}{2}\right)\frac{\pi}{L}]$$, where $m = 1, \dots, L$

Chroma Vector

Chroma vector is a 12-element frequency representation of an audio signal. Each k value is computed by:

$$v_k = \sum_{n \in S_k} \frac{X_i(n)}{N_k}$$, where $k = 1, \dots, 12$

Index	**Name**	**Description**
1	Zero Crossing Rate	The rate of sign-changes of the signal during the duration of a particular frame.
2	Energy	The sum of squares of the signal values, normalized by the respective frame length.
3	Entropy of Energy	The entropy of sub-frames' normalized energies. It can be interpreted as a measure of abrupt changes.
4	Spectral Centroid	The center of gravity of the spectrum.
5	Spectral Spread	The second central moment of the spectrum.
6	Spectral Entropy	Entropy of the normalized spectral energies for a set of sub-frames.
7	Spectral Flux	The squared difference between the normalized magnitudes of the spectra of the two successive frames.
8	Spectral Rolloff	The frequency below which 90% of the magnitude distribution of the spectrum is concentrated.

9–21	MFCCs	Mel Frequency Cepstral Coefficients form a cepstral representation where the frequency bands are not linear but distributed according to the mel-scale.
22–33	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).
34	Chroma Deviation	The standard deviation of the 12 chroma coefficients.

Table 5: Audio features [3, 48, 82, 83]

iii) Classification Methodology

Besides the 5 general classes of techniques, we introduce also a new class (called None) which corresponds to parts of a given sample with absence of technique. A segment of a song can be either one of the above-mentioned techniques or it could be a simple melody without any technique. In order to address this, we experiment with different confidence levels for the classifications of the models. If the confidence of the classification made by a model for a specific segment is lower than the selected confidence, than we “do not allow” any classification by the model. Instead, the classification of the model is set to “None”. Respectively, in the ground truth of a test recording, segments not containing any technique, are assigned a “None” value.

At first, the 40 aforementioned models (5 models X 8 combinations) are trained. Then, for each test recording (with its ground truth) the below steps are followed:

For every confidence level in the range of [0.4,1) with step 0.025 we let every model make a prediction for each segment of the test recording. If the confidence of the classification of the model for the specific segment is lower than the confidence we examine, than the model classifies the segment as “None” technique. If the classification of the model for the segment is greater than the selected confidence, we accept the predicted value. This continues for all segments of each test recording.

To evaluate accuracy, we have introduced weighted errors for misclassifications, as some techniques are more similar to each other. As a result, some misclassifications cannot be considered totally false. These weighted errors have been discussed and chosen by 2 professional music players and 3 amateur players and are presented in Table 6. According to the weighted errors matrix, the normalized accuracy is calculated.

	2 strings	chord	slurring	trembling 1 string	trembling 2 strings	None
2 strings	0	0.5	1	1	0.5	1
chord	0.5	0	1	1	0.8	1
slurring	1	1	0	1	1	1
trembling 1 string	1	1	1	0	0.5	1
trembling 2 strings	0.5	0.8	1	0.5	0	1
None	1	1	1	1	1	0

Table 6: Matrix of normalized weighted errors for misclassifications

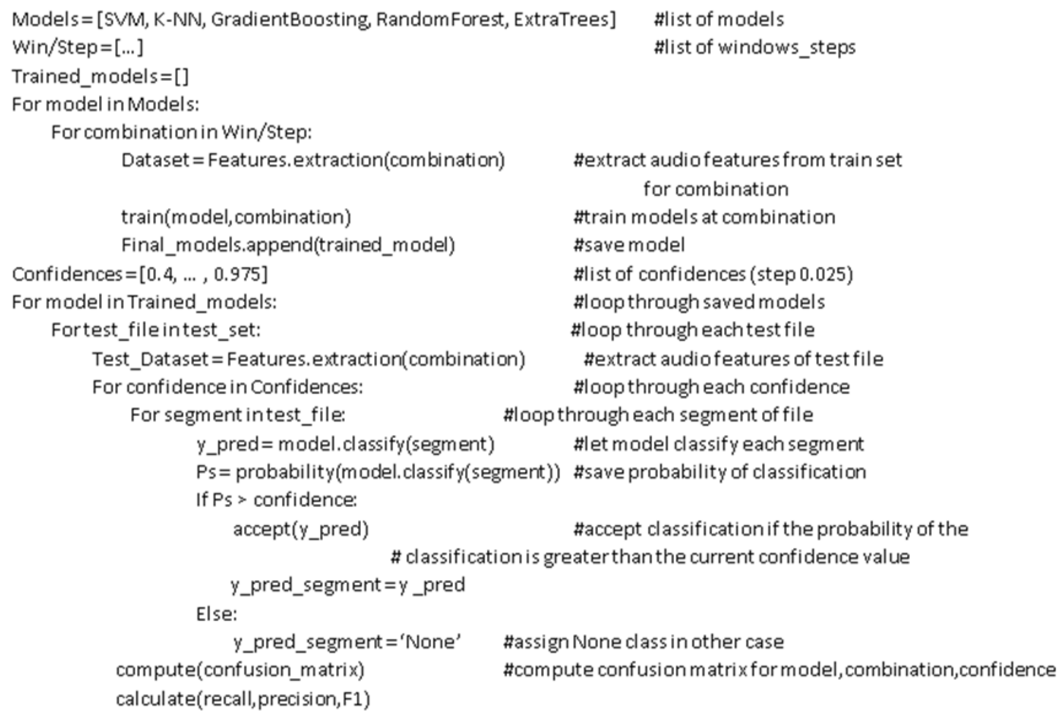
Two chords play consists of a strike in two strings but not in a continuous fashion (as Trembling in two strings). Two chords play is more like a simple melody in two strings, whereas Trembling in two strings is the continuous strike (downward and upward) of two strings. However, the two techniques have in common the strike in two strings. Therefore, a half unit error is introduced for the misclassification of technique “2 strings play” and “Trembling in one chord” and the opposite.

In addition, Chords are the combination of strike of 3 or 4 strings. Chords and “2 string play” have therefore two things in common. The first is the strike of 2 (or more in the case of Chords) strings. The second one is the not continuous fashion of play. For instance, both techniques do not refer to trembling, as they are simple strikes tempowise. As a result, an error of half unit is introduced for such misclassification. In the opposite hand, “Chords” and “Trembling in two strings” have a 0.8 error, as they have in common a significant parameter (two or more strings play) but differ significantly in the playing tempo (Trembling in two strings is a much faster technique than Chords).

Trembling in one string and Trembling in two strings have in common the necessity of fast play. More specifically, both techniques require a continuous strike (downward and upward). However, they differ significantly in the number of strings played. Therefore, these two techniques have a 0.5 error value for the case of misclassification. All other misclassification cases have a typical error, which cannot be normalized, as these techniques have nothing in common. All normalized errors have been decided unanimously by the 2 professional and 3 amateur bouzouki players.

In Figure 14, the algorithm of the approach is presented. At first, a list of all models to be trained is initialized (SVM, K-NN, Gradient Boosting, Extra Trees, Random Forest). Then, the combinations of mid-term windows, mid-term steps, short-term windows and short-term steps are also initialized. For each model (SVM, K-NN, Gradient Boosting, Extra Trees, Random Forest) and each window/step combination, the audio features extraction is the next step: The training recordings are processed (audio signal processing) and the features of Table 5 are extracted for a specific combination (Table 4). These features are used as training Dataset. After training a model on a specific window/step combination (model, combination), the model is stored for later use.

The next step is to initialize a range of confidence values to check. Then, for each stored model the following methodology is followed: For each test file, the audio features are extracted for the given mid-term window, mid-term step, short-term window and short-term step combination are extracted and given as input to the model for classification. The next step if to loop through confidence values. The confidences values are all values between 0.4 and 0.975 with step 0.025. For each of these values, the model classifies each segment (y_pred). For a given confidence c, we accept the classification of a model for a specific segment only if its probability is greater than c. If the probability of the classifier's decision is greater than the investigated confidence value, then this classification is accepted, otherwise the class 'None' is assigned to the segment. After classifying all segments of all test files for a specific model, combination of windows/steps and confidence, the evaluation metrics are computed (confusion matrix, Recall, Precision, F1, accuracy, normalized accuracy). The ROC curves are also plotted and evaluated. The above procedure results in 1440 different tests (36 confidence X 8 window/step combinations X 5 models) to compare efficiency (Recall, Precision, F1-score, Accuracy, Normalized Accuracy, ROC curves).



```
Models=[SVM, K-NN, GradientBoosting, RandomForest, ExtraTrees]      #list of models
Win/Step=[...]                                                      #list of windows_steps
Trained_models=[]
For model in Models:
     For combination in Win/Step:
               Dataset=Features.extraction(combination)          #extract audio features from train set
                                                                          for combination
               train(model,combination)                          #train models at combination
               Final_models.append(trained_model)                #save model
Confidences=[0.4, ... , 0.975]                                   #list of confidences (step 0.025)
For model in Trained_models:                                     #loop through saved models
     For test_file in test_set:                                  #loop through each test file
          Test_Dataset=Features.extraction(combination)           #extract audio features of test file
          For confidence in Confidences:                         #loop through each confidence
               For segment in test_file:                 #loop through each segment of file
                         y_pred= model.classify(segment)         #let model classify each segment
                         Ps= probability(model.classify(segment)) #save probability of classification
                         If Ps > confidence:
                              accept(y_pred)                     #accept classification if the probability of the
                                                 # classification is greater than the current confidence value
                              y_pred_segment=y _pred
                         Else:
                              y_pred_segment='None'      #assign None class in other case
               compute(confusion_matrix)                 #compute confusion matrix for model,combination,confidence
               calculate(recall,precision,F1)
```

Figure 14: The methodology of the approach for the recognition of instrument playing technique

iv) Model Training

The models trained were evaluated using the function evaluateClassifier of pyAudioAnalysis in order to perform cross validation and select the optimal classifier parameter. For SVM, the parameter studied is the soft margin parameter C. For k-NN classifier, the number of nearest neighbors was examined whereas for Random Forest classifier the number of trees was studied. For Gradient Boosting, the amount of boosting stages was optimized during training. Finally, the Extra Trees classifier was studied as of the number of trees.

For SVM, the parameter under study (C) is the penalty parameter C of the error term. Depending on its value, C can influence the balance between low training errors and low testing errors. In other words, C is a crucial parameter for the model's generalization on unseen data. For Gradient Boosting, the boosting stages (number of estimators) are the number of trees in the forest. The parameter tuning is done in order to find a good balance point between low values (which can cause the model to overfit) and high values (which can slow down the training significantly).

After training and testing on the test recordings, the 8 models that achieved the best F1-scores and accuracies (>0.5) were selected. These are presented and discussed in Figures 15-30 in the section of results. Regarding parameter tuning during training, the optimezed parameters (C for svm and number of estimators for Gradient Boosting) of the best 8 models are presented in Table 7 below.

Model	**Parameters**
svm (0.9, 0.6, 0.06, 0.02)	Soft margin parameter **C**: 0.01
gradient boosting (0.9, 0.6, 0.06, 0.02)	Number of Estimators (***n_estimators)***: 200
svm (0.8, 0.4, 0.02, 0.01)	Soft margin parameter **C**: 0.01
svm (0.8, 0.2, 0.04, 0.01)	Soft margin parameter **C**: 20.0
svm (0.6, 0.2, 0.015, 0.005)	Soft margin parameter **C**: 0.01
gradient boosting (0.6, 0.2, 0.015, 0.005)	Number of Estimators (***n_estimators)***: 100
svm (0.5, 0.25, 0.01, 0.005)	Soft margin parameter **C**: 0.01
gradient boosting (0.5, 0.25, 0.01, 0.005)	Number of Estimators (***n_estimators)***: 500

Table 7: Selected Parameters of 8 best models

Besides C for SVM and boosting stages for Gradient Boosting, the rest of the parameters are in common for all models, are they were note finetuned. These parameters are presented in Table 8. For SVC, the kernel parameter can be either ‘linear’, ‘poly’, ‘rbf’, ‘sigmoid’ or ‘precomputed’ and refers to the kernel type to be used. Kernels are mathematical functions which take the input data and transform it into another dimension. This dimensionality transformation is done in order to achieve a clear dividing margin between classes. In this case, the kernel type selected was linear. This means, that the Data is separated by a straight line. The degree refers to the degree of the polynomial kernel function (‘poly’). In the ‘linear’ kernel case, this value is ignored by taking the default value 3. The parameter gamma is a coefficient used for the kernel types ‘rbf’, ‘poly’, and ‘sigmoid’. In the case of ‘linear’ kernel type, gamma has the value ‘auto_deprecated’ which is used to indicate that no explicit value of gamma was passed. Coef0 is another irrelevant parameter with kernel type ‘linear’, which is used only in ‘poly’ and ‘sigmoid’ kernel functions. Shrinking refers to the shrinking heuristics which are used to speed up the optimization and effect the runtime. In this case, shrinking was se to true, enabling the shrinking heuristics to be used in the optimization. The parameter probability is set here to true, in order to also compute probability estimates. However, enabling the calculation of probability estimates can slow down the training process. The tol parameter refers to the tolerance for the stopping criteria (allowed in the optimization process). Once this value is achieved, the search for the optimum is stopped. Here, the tol value is set to the default 0.0001. Cache_size is the size of the kernel cache (in MB) and has a strong impact on runtime of the training. Here, cache_size is set to default value 200. The parameter class_weight is used to assign weights to classes. In this case, it is set to None, signifying that all classes have equal weights (1). Verbose is a parameter that enables or disables detailed logging information. Here, it is set to False. Max_iter refers to a hard limit on iterations within solver. Specifically, the value of max_iter is the maximum number of iterations allowed in order to optimize. In this case, it is set to -1, signifying no limit of iterations. The parameter decision_function_shape is used for specifying the methodology of treating the different classes. In this case, it is set to ‘ovr’, which refers to One vs Rest approach. This approach models each class against all of the other classes independently and trains a classifier for each situation. Random_state specifies the seed of the pseudo random number generator used when shuffling the data for probability estimates. Here, it is set to None which uses the random state generator of Numpy.

For Gradient Boosting, the parameter criterion refers to the function to be called to measure the quality of a split. The 'friedman_mse' selected here uses the mean squared error with an improvement score by Friedman. The loss parameter refers to the loss function to be optimized. Here, it is set to 'deviance', which signifies a logistic regression approach with probabilistic outputs. Learning_rate value specifies the value, by which the contribution of each tree is shrinked. Here, it is set to the default value of 0. 1. Subsample indicates the fraction of samples that are used for fitting the individual ensembles. The value of subsample here is 1. Min_sample_split is the minimum number of samples required to split an internal node. Assigning high values to this parameter can cause the tree to become more constrained as it has to consider more samples at each node. Here, min_samples_split is set to the default value 2. Min_samples_leaf is the minimum number of samples required to be at a leaf node. Here it is set to the default value 1, meaning that one sample is required at least to be at a leaf node. Min_weight_fraction_leaf refers to weights of samples. In this case, it is set to 0, resulting in an equal weight of the samples. Max_depth refers to the deepness of the tree to be built. Higher values result in deeper trees and more splits necessary. In addition, higher value can capture more information about the data and generalize better. There is, however, a risk of overfitting. In this case, max_depth was set to the default value 3. Min_impurity_decrease is the value, above (or equal to) which any node will be split regarding the induced decrease of impurity. Min_impurity_decrease was set here to the default value 0, meaning there is no test regarding splitting and impurity decrease. The parameter min_impurity_split indicates a threshold for early stopping of the tree growing. Any node will split if the impurity it induces is higher than this threshold. In the opposite case, the node will be a leaf node. Here, the parameter value is the default value 'None', meaning there is no such threshold. Init is an estimator for the computation of the initial predictions. It is set to None, as there is no need for initial predictions. The parameter random_state refers to the random number generator used as in SVC. It is set to 'None' and thus the Numpy random generator is used. Max_features is the number of features that need to be considered for the best split. It is set to 'None', resulting in max_features equal to the number of features. Verbose (just as in SVC) enables or disables detailed logging information and here it is disabled (value 0). Max_leaf_nodes specifies the maximum amount of leaf nodes in the tree allowed. Here, it is set to None, meaning there is no limit in the number of leaf nodes. Warm_start is used to recall and reuse aspects of the model learnt from previous parameter values. Here it is set to False, which erases the previous solution. Presort is used in order to pre-sort the data and speed up the search of the best splits. Here, it has the default value 'auto', which enables the pre-sorting on dense data and default to normal sorting on parse data. The parameter validation_fraction indicates the proportion to training data to be considered and left as validation set for early stopping. The default value of 0.1 was used. The value of n_iter_no_change decided whether early stopping will be used in the case that validation score shows no improving. Here, it was disabled by assigning to it the default value None. The parameter tol (as in SVC) refers to the tolerance for the stopping criteria (allowed in the optimization process). Once this value is achieved, the search for the optimum is stopped. Here, the tol value is set to the default 0.0001.

Model	Parameters	
SVC	**kernel**	'linear'
	degree	3
	gamma	'auto_deprecated'
	coef0	0
	shrinking	TRUE
	probability	TRUE
	tol	0.0001
	cache_size	200
	class_weight	None
	verbose	FALSE
	max_iter	-1
	decision_function_shape	'ovr'
	random_state	None
	loss	'Deviance'
	learning_rate	0.1
	subsample	1
	criterion	friedman_mse'
	min_samples_split	2

GradientBoosting	**min_samples_leaf**	1
	min_weight_fraction_leaf	0
	max_depth	3
	min_impurity_decrease	0
	min_impurity_split	None
	init	None
	random_state	None
	max_features	None
	verbose	0
	max_leaf_nodes	None
	warm_start	FALSE
	presort	‘auto’
	validation_fraction	0.1
	n_iter_no_change	None
	tol	0.0001

Table 8: Parameter values for models SVC and Gradient Boosting

5. IMPLEMENTATION ISSUES

Firstly, 8 different combinations of mid and short term windows and steps are selected (Table 4). Making use of the Machine Learning models (5 in total) offered in pyAudioAnalysis library for audio classification results in a total of 40 models (8 combinations X 5 models) [48]. These models are SVM, K-NN, Extra Trees, Random Forests and Gradient Boosting. The models trained were evaluated using the function evaluateClassifier of pyAudioAnalysis in order to perform cross validation and select the optimal classifier parameter. For SVM, the parameter studied is the soft margin parameter C. For k-NN classifier, the number of nearest neighbors was examined whereas for Random Forest classifier the number of trees was studied. For Gradient Boosting, the amount of boosting stages was optimized during training. Finally, the Extra Trees classifier was studied as of the number of trees.

The function train(mt_size,mt_step,st_size,st_step) is the first function called. It takes as argument the mid-term size and step and short-term size and step. The function featureAndTrain of module audioTrainTest of pyAudioAnalysis is then called 5 times in total, one time for each model. The arguments passed in featureAndTrain are the paths to the folders containing the techniques, the windows and sizes, the preferred model and a path to save the model trained. This function then analyzes the training Data (feature extraction) with respect to the passed mid and short term parameters and trains the models. The training Dataset can be found by clicking here [85].

Then, the test(filepath,segmentpath,confidence,test_name) function is called for each file of the test set. The test files can be found by clicking here [86]. The first argument passed is the path to the test file. The path to the ground-truth segment file (manually annotated) is also passed for later comparison between ground truth and classifications. In addition, this function is called for every confidence level in the range of [0.4,1) with step 0.025. A test_name argument is passed in order to save respective results. This function then searches for all models stored after train function. For each model (out of the 40 in total) and for each confidence the function mtFileClassification(input_file, model_name, model_type, confidence, test_name, plot_results=False, gt_file="") of the module audioSegmentation of pyAudioAnalysis is called. This function is adjusted to the needs of the current work as following: We let every model make a prediction for each segment of the test recording. If the confidence of the classification of the model for the specific segment is lower than the confidence we examine, than the model classifies the segment as “None” technique. If the classification of the model for the segment is greater than the selected confidence, we accept the predicted value. This continues for all segments of each test recording. The resulted confusion matrices of each classification of each test file on each model and confidence are stored all together in subfolders per test file. Additionally, mtFileClassification was altered as following: To evaluate accuracy, we have introduced weighted errors for misclassifications (Table 6), as some techniques are more similar to each other. As a result, some misclassifications cannot be considered totally false. These weighted errors have been discussed and chosen by 2 professional music players and 3 amateur players. The normalized accuracy is calculated according to the weighted errors matrix, which is also used for all next steps.

Afterwards, a sort_folders() is called in order to store the confusion matrices created in previous step firstly per test name and then per model in respective subdirectories. Then the create_folder_models() is called in order to store confusion matrices firstly per model, then in subdirectories per test file. Then, calculate_partial_f1s() is called which analyzes the stored confusion matrices of previous step, computes recall, precision and f1-score per model, confidence and test file and stores them. Function calculate_total_f1s is called then, which analyzes the classification results of previous step and creates a single file for each model (40 models) . This file is a table containing all possible confidence and respective average precision, recall and f1-score of all test files. Additionally, it computes the recall vs precision graphs for each model and saves them only if the f1-score is greater than 0.5.

Then, the files stored from create_folder_models (separate confusion matrices per model, test file and confidence) are passed in the function create_partial_folders_per_confidence(), which saves the confusion matrices first per model, then per confidence and then per test file. Later, create_total_folders_per_confidence() is called. This function takes the files from last step and calculates the total confusion matrices for all test files per model and then per confidence. Finally, the stored files are analyzed by function precision_vs_recall()

which computes the total recall, precision and f1-scores for all test files and for each model. It also plots the recall vs precision graphs with respect to the confidence (Figures 15, 17, 19, 21, 23, 25, 27, 29) for the 8 best models.

Finally, function roc_curves(model) is called 40 times in total (one time for each model). It takes as argument the model name and searches in the directory of lastly stored (by function create_total_folders_per_confidence()) confusion matrices for the respective confusion matrix of the model. It calculates True Positive Rate (TPR) and False Positive Rate (FPR) for each model (all test files and for each confidence level) and plots the Roc Curves (Figures 16, 18, 20, 22, 24, 26, 28, 30) for the best 8 models. Figures 33-42 were created with application of the normalized weighted error matrix on mtFileClassification and plotSegmentationResults of pyAudioAnalysis. The methodology of the approach is also presented in Figure 14. The code described above can be found by clicking here [87]. Besides pyAudioAnalysis, the libraries numpy, pandas and matplotlib were used [88-90].

For the purposes of this Thesis, an open-source standalone python script was developed which classifies each segment of a given solo bouzouki instrument recording as of the playing technique using the best 6 (pre-trained) models of this Thesis. The repository with the code and the pre-trained models of the project can be found in github [91]. Chapter 9 of this work covers a detailed description of the repository and the code.

6. RESULTS AND DISCUSSION

Precision, Recall and F1 were calculated for all test recordings together. In order to evaluate the results, the means of Precision, Recall and F1 for the non None techniques were computed. Precision grows and recall shrinks as confidence level grows for the non None techniques. On the other hand, for the None technique, precision shrinks and recall grows. In addition, the roc curves for each model on the whole training Dataset (only for the techniques) are also computed. The graphs of Precision vs Recall and the ROC curves of the best 8 models are presented in Figures 15-30.

I) Multiclass Roc Curves, AUC Score and Confusion Matrices

A receiver operating characteristic curve (ROC) is a graphical plot that shows the ability of a classifier to discriminate two classes depending on the confidence of the classifications. Specifically, the ROC curve shows the efficiency of the model to distinguish two different classes with respect to the probabilities of the predictions. In general, the Y axis of the ROC curve is the true positive rate of a class whereas the X axis represents the false positive rate. As a result, a false positive rate of zero and a true positive rate of one (which is the best discrimination that can be achieved) is on the top left corner of the plot. The confidence threshold that gives ROC value closer to this point is typically a good threshold compared to the rest. As mentioned above, ROC curves are illustrations that are used in binary classifications. In order to apply ROC curves for multi-class tasks, the label must be first binarized. In addition, one ROC curve is plotted per class. The micro-averaging ROC curve can be computed if considering each label as binary prediction (One vs Rest). Macro-averaging ROC curve is computed by assigning equal weight to the classification of each label. The AUC score refers to “area under the ROC curve” and signifies the area underneath the ROC curve. It is an aggregate metric of the performance with respect to all classification thresholds (confidences). AUC can be related to the probability of the model ranking a random positive (true) instance highly than a random negative (false) instance.

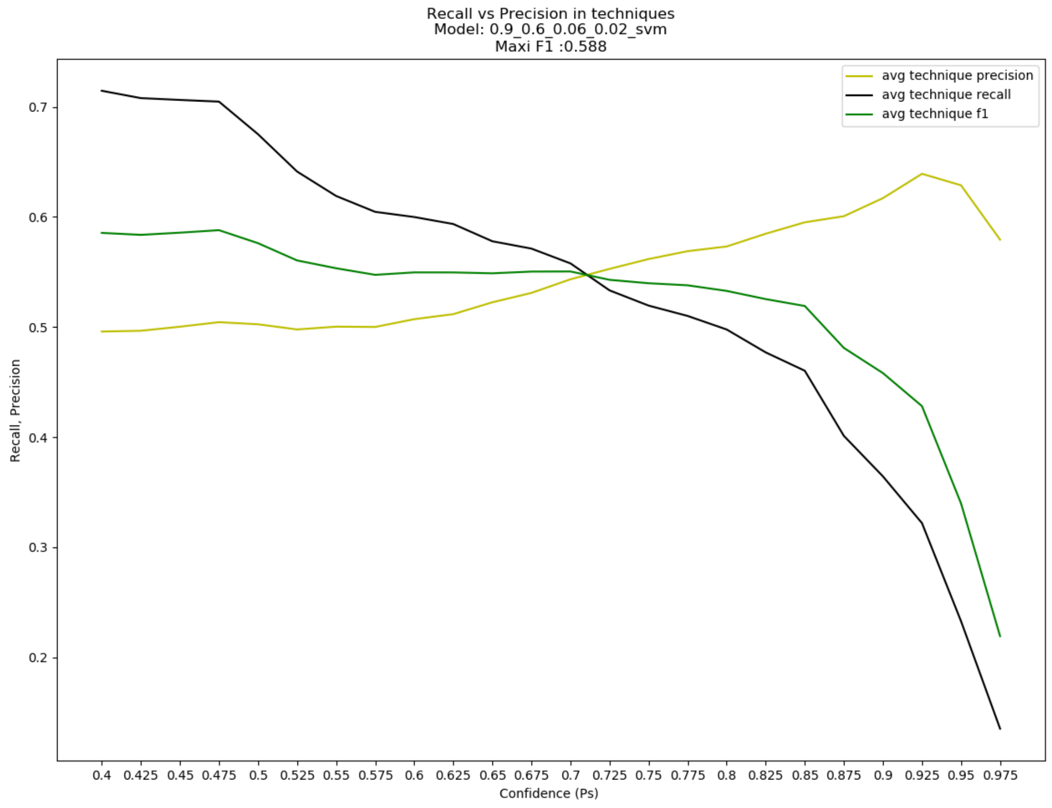


Figure 15: Recall vs Precision for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model svm for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved the highest F1 (0.5880) score of all models at Confidence value 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of 0.7. At the point of Precision, Recall and F1-Score equality (confidence 0.7), their values are 0.55. The confusion matrix of the model for all test recordings and confidence 0.7 is presented on Table 9 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	14	7	0	0	1
trembling2	0	22	0	0	0
slurring	3	2	36	6	1
2strings	4	20	2	99	1
chord	0	0	0	0	25

Table 9: Confusion Matrix of Techniques for model svm (mid-term window: 0.9, mid-term step: 0.6, short-term window: 0.06 and short-term step: 0.02

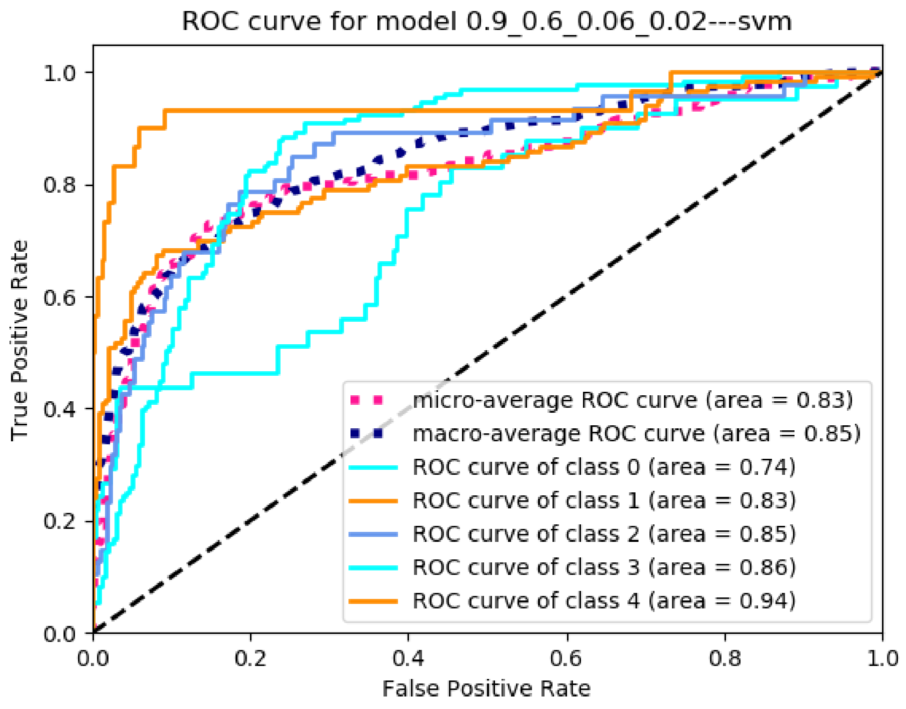


Figure 16: Roc curve for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model svm for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved a micro average ROC curve area of 0.83 and a macro-average ROC curve area of 0.85. The class “Trembling in one string” achieved a ROC curve area of 0.74 and “Trembling in two strings“ achieved a ROC curve area of 0.83. The class “Slurring” achieved a ROC curve area of 0.85 and the class “2 strings play” a ROC curve area of 0.86. Finally, the class “Chord” achieved a ROC curve area of 0.94. The AUC scores are presented in Table 10.

Technique	**AUC score**
Trembling in one string	0.74
Trembling in two strings	0.83
Slurring	0.85
2 strings play	0.86
Chord	0.94
Micro-average	0.83
Macro-average	0.85

Table 10: AUC scores for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

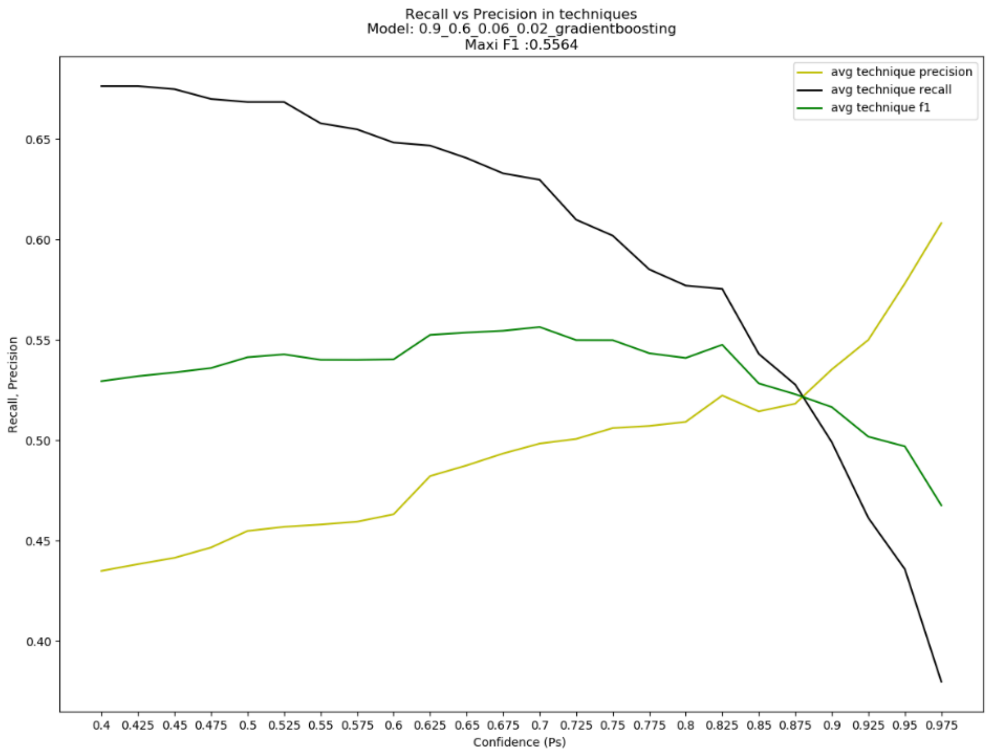


Figure 17: Recall vs Precision for model gradient boosting and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model gradient boosting for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved the highest F1 score (0.5564) at Confidence value 0.7. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of 0.9. At the point of Precision, Recall and F1-Score equality (confidence 0.9), their values are 0.52. The confusion matrix of the model for all test recordings and confidence 0.9 is presented on Table 11 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	17	13	0	0	1
trembling2	0	19	0	0	0
slurring	3	6	33	7	1
2strings	2	26	2	74	1
chord	3	1	0	0	22

Table 11: Confusion Matrix of Techniques model gradient boosting and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

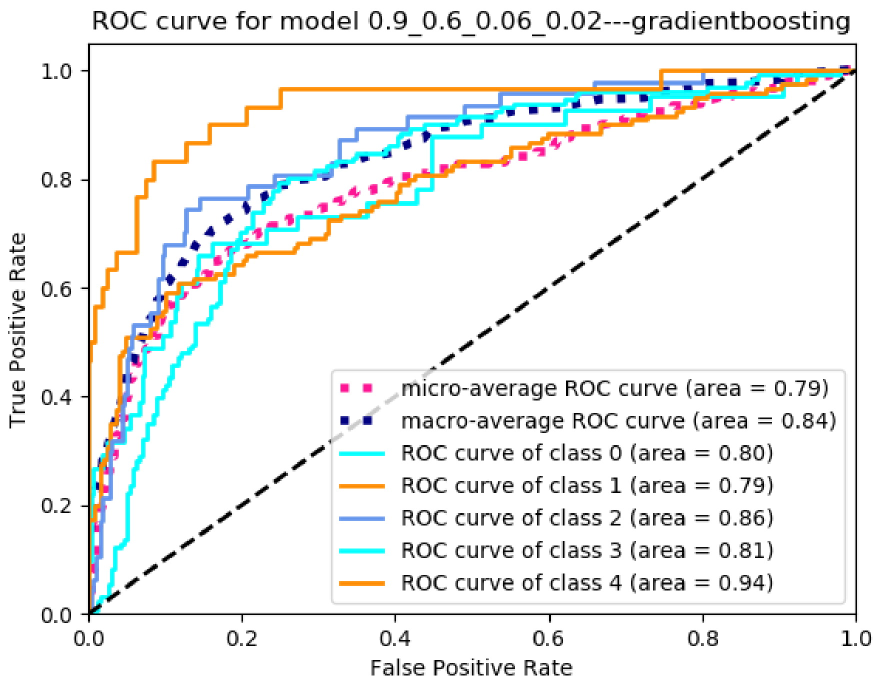


Figure 18: Roc curve for model gradient boosting and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model gradient boosting for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved a micro average ROC curve area of 0.79 and a macro-average ROC curve area of 0.84. The class “Trembling in one string” achieved a ROC curve area of 0.80 and “Trembling in two strings” achieved a ROC curve area of 0.79. “Slurring” achieved a ROC curve area of 0.86 and the class “2 strings play” a ROC curve area of 0.81. The class “Chords“ achieved a ROC curve area of 0.94. The AUC scores are presented in Table 12.

Technique	**AUC score**
Trembling in one string	0.80
Trembling in two strings	0.79
Slurring	0.86
2 strings play	0.81
Chord	0.94
Micro-average	0.79
Macro-average	0.94

Table 12: AUC scores for model gradient boosting and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

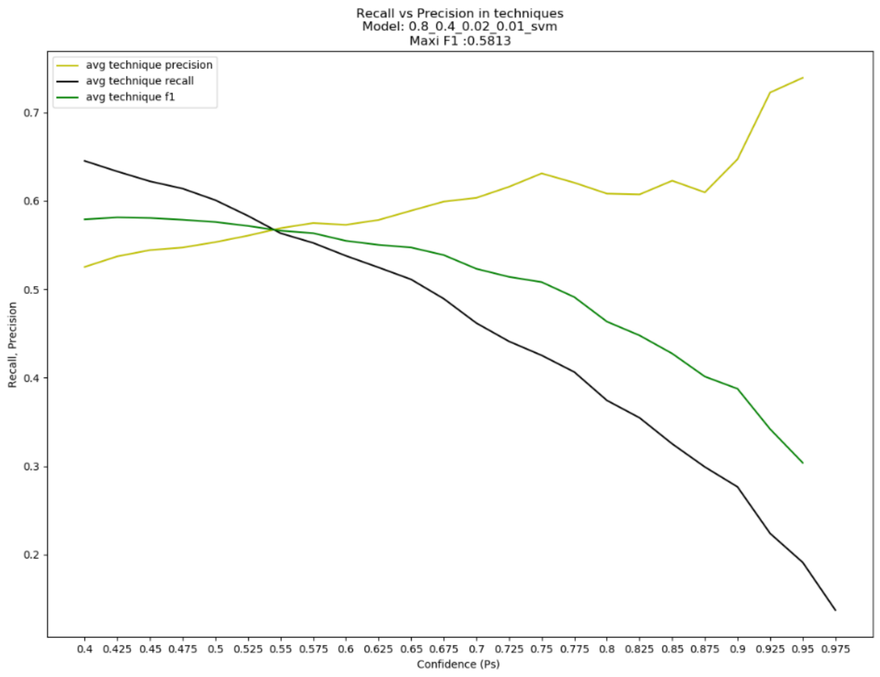


Figure 19: Recall vs Precision for model svm and combination 0.8 (mt window), 0.4 (mt step), 0.02 (st window), 0.01 (st step)

The model svm for mid-term window 0.8, mid-term step 0.4, short-term window 0.02 and short-term step 0.01 achieved the highest F1 score (0.5813) at Confidence value 0.7. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of 0.525. At the point of Precision, Recall and F1-Score equality (confidence 0.525), their values are 0.55. The confusion matrix of the model for all test recordings and confidence 0.525 is presented on Table 13 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	21	5	0	0	1
trembling2	0	62	0	0	2
slurring	13	19	54	22	3
2strings	12	45	11	166	4
chord	7	0	1	0	36

Table 13: Confusion Matrix of Techniques model model svm for mid-term window 0.8, mid-term step 0.4, short-term window 0.02 and short-term step 0.01

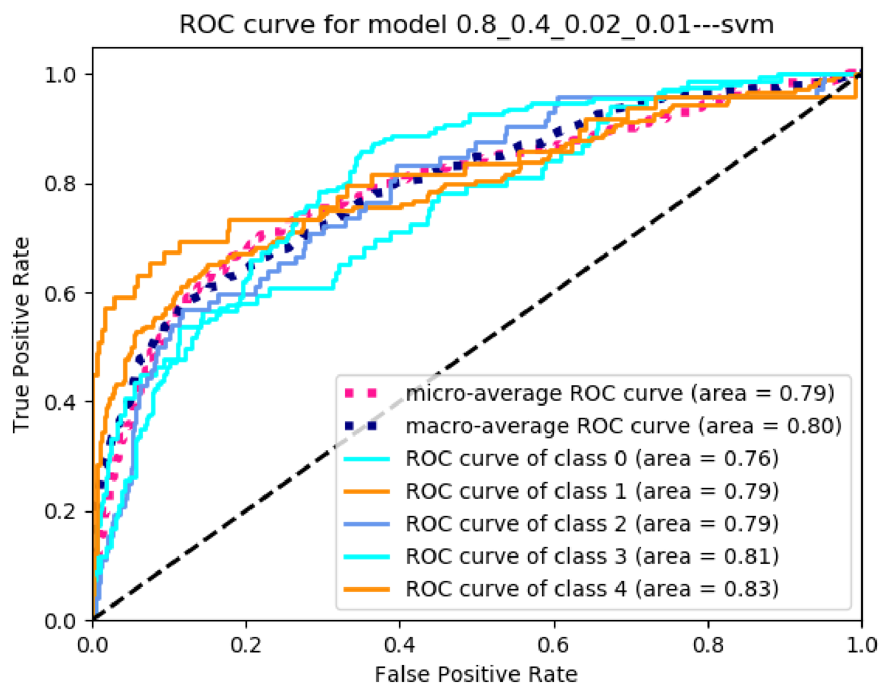


Figure 20: Roc curve for model svm and combination 0.8 (mt window), 0.4 (mt step), 0.02 (st window), 0.01 (st step)

The model svm for mid-term window 0.8, mid-term step 0.4, short-term window 0.02 and short-term step 0.01 achieved a micro average ROC curve area of 0.79 and a macro-average ROC curve area of 0.80. The class "Trembling in one string" achieved a ROC curve area of 0.76 and "Trembling in two strings" achieved a ROC curve area of 0.79. The class "Slurring" achieved a ROC curve area of 0.79 and the class "2 strings play" a ROC curve area of 0.81. Finally, the class "Chord" achieved a ROC curve area of 0.83. The AUC scores are presented in Table 14.

Technique	**AUC score**
Trembling in one string	0.76
Trembling in two strings	0.79
Slurring	0.79
2 strings play	0.81
Chord	0.83
Micro-average	0.79
Macro-average	0.80

Table 14: AUC scores for model svm for mid-term window 0.8, mid-term step 0.4, short-term window 0.02 and short-term step 0.01

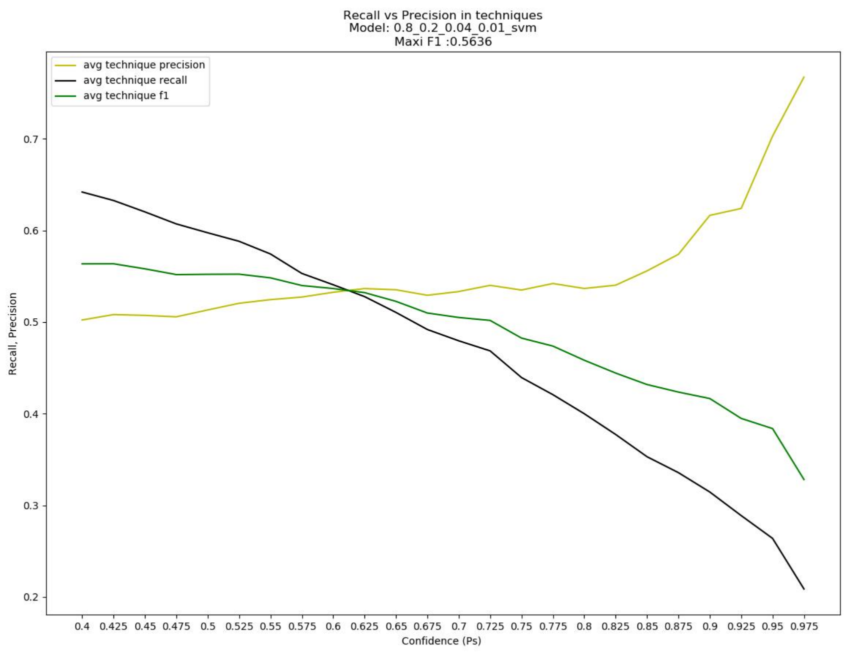


Figure 21: Recall vs Precision for model svm and combination 0.8 (mt window), 0.2 (mt step), 0.04 (st window), 0.01 (st step)

The model svm for mid-term window 0.8, mid-term step 0.2, short-term window 0.04 and short-term step 0.01 achieved the highest F1 score (0.5636) at Confidence value 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of around 0.625. At the point of Precision, Recall and F1-Score equality (confidence 0.625), their values are 0.54. The confusion matrix of the model for all test recordings and confidence 0.625 is presented on Table 15 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	25	12	0	0	3
trembling2	0	70	0	0	3
slurring	34	16	116	42	6
2strings	2	74	4	282	7
chord	6	10	7	10	77

Table 15: Confusion Matrix of Techniques model svm for mid-term window 0.8, mid-term step 0.2, short-term window 0.04 and short-term step 0.01

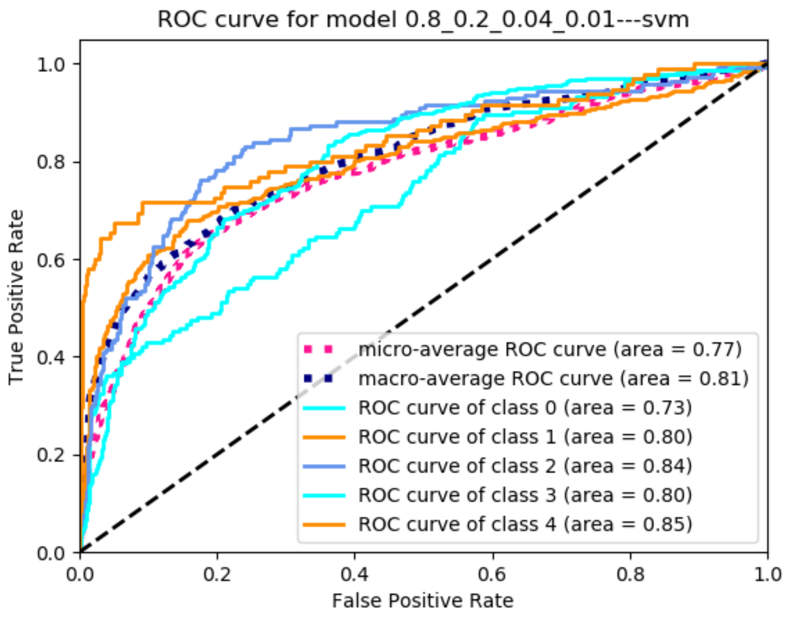


Figure 22: Roc curve for model svm and combination 0.8 (mt window), 0.2 (mt step), 0.04 (st window), 0.01 (st step)

The model svm for mid-term window 0.8, mid-term step 0.2, short-term window 0.04 and short-term step 0.01 achieved a micro average ROC curve area of 0.77 and a macro-average ROC curve area of 0.81. The class “Trembling in one string” achieved a ROC curve area of 0.73 and “Trembling in two strings” achieved a ROC curve area of 0.80. The class “Slurring” achieved a ROC curve area of 0.84 and the class 2 strings play” a ROC curve area of 0.80. Finally, the class “Chord” achieved a ROC curve area of 0.85. The AUC scores are presented in Table 16.

Technique	**AUC score**
Trembling in one string	0.73
Trembling in two strings	0.80
Slurring	0.84
2 strings play	0.80
Chord	0.85
Micro-average	0.77
Macro-average	0.81

Table 16: AUC scores for model svm for mid-term window 0.8, mid-term step 0.2, short-term window 0.04 and short-term step 0.01

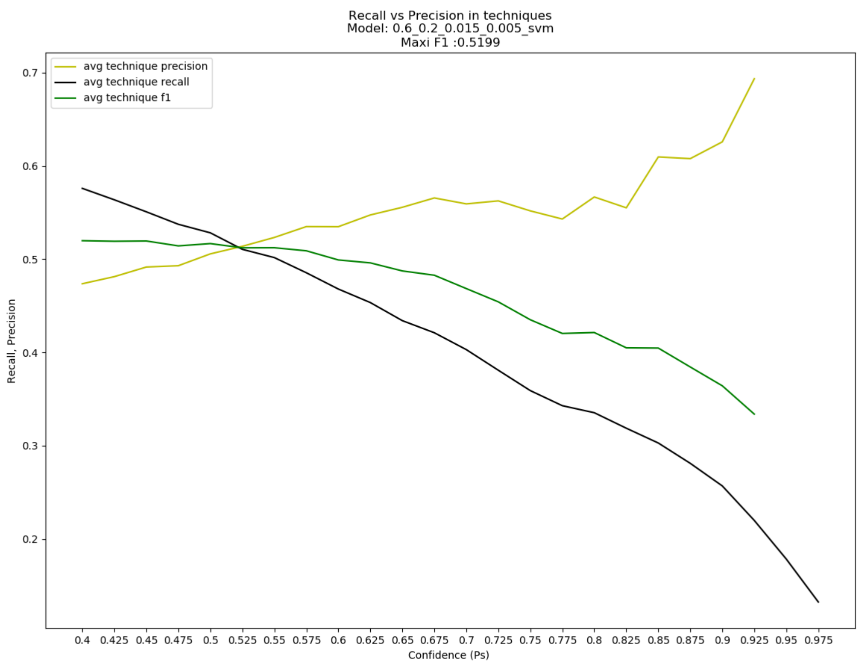


Figure 23: Recall vs Precision for model svm and combination 0.6 (mt window), 0.2 (mt step), 0.015 (st window), 0.005 (st step)

The model svm for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005 achieved the highest F1 score (0.5199) at Confidence value around 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of around 0.525. At the point of Precision, Recall and F1-Score equality (confidence 0.525), their values are 0.51. The confusion matrix of the model for all test recordings and confidence 0.525 is presented on Table 17 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	37	0	0	0	4
trembling2	0	128	1	8	5
slurring	9	16	76	53	6
2strings	25	114	42	315	14
chord	13	6	2	0	66

Table 17: Confusion Matrix of Techniques model svm for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005

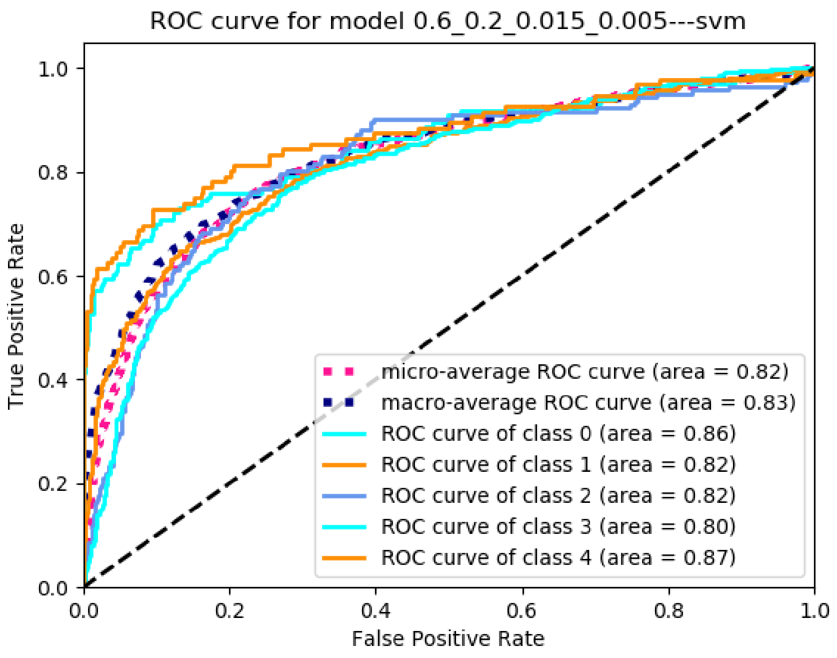


Figure 24: Roc curve for model svm and combination 0.6 (mt window), 0.2 (mt step), 0.015 (st window), 0.005 (st step)

The model svm for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.05 achieved a micro average ROC curve area of 0.82 and a macro-average ROC curve area of 0.83. Specifically, the class “Trembling in one string” achieved a ROC curve area of 0.86 and “Trembling in two strings” achieved a ROC curve area of 0.82. The class “Slurring” achieved a ROC curve area of 0.82 and the class “2 strings play” a ROC curve area of 0.80. The class “Chord” achieved a ROC curve area of 0.87. The AUC scores are presented in Table 18.

Technique	**AUC score**
Trembling in one string	0.86
Trembling in two strings	0.82
Slurring	0.82
2 strings play	0.80
Chord	0.87
Micro-average	0.82
Macro-average	0.83

Table 18: AUC scores for model svm for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.05

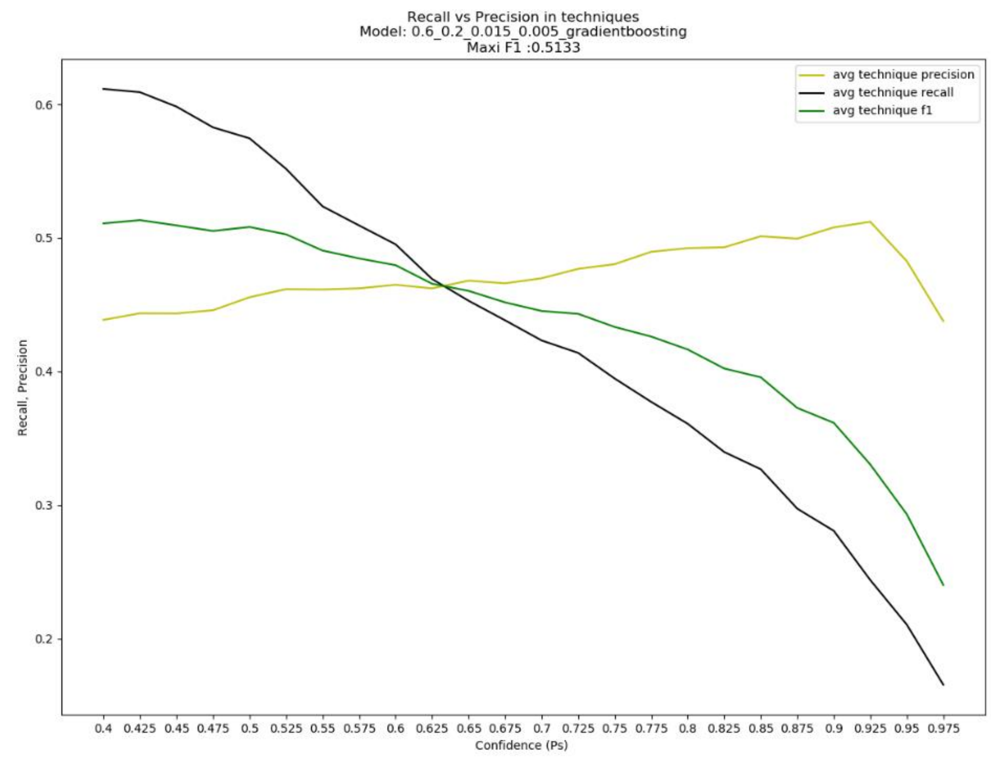


Figure 25: Recall vs Precision for model gradient boosting and combination 0.6 (mt window), 0.2 (mt step), 0.015 (st window), 0.005 (st step)

The model gradient boosting for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005 achieved the highest F1 score (0.5133) at Confidence value 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of around 0.625. At the point of Precision, Recall and F1-Score equality (confidence 0.625), their values are 0.46. The confusion matrix of the model for all test recordings and confidence 0.625 is presented on Table 19 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	38	1	0	2	4
trembling2	2	132	0	2	8
slurring	21	21	73	85	10
2strings	3	108	9	200	3
chord	10	15	5	14	75

Table 19: Confusion Matrix of Techniques model gradient boosting for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005

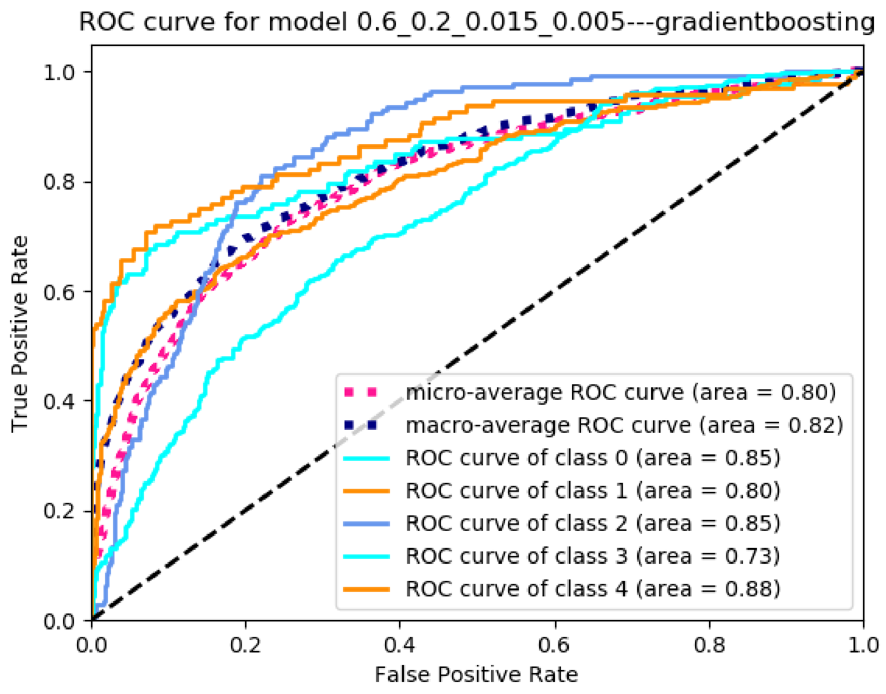


Figure 26: Roc curve for model gradient boosting and combination 0.6 (mt window), 0.2 (mt step), 0.015 (st window), 0.005 (st step)

The model gradient boosting for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005 achieved a micro average ROC curve area of 0.80 and a macro-average ROC curve area of 0.82. The class “Trembling in one string” achieved a ROC curve area of 0.85 and “Trembling in two strings” achieved a ROC curve area of 0.80. “Slurring” achieved a ROC curve area of 0.85 and the class “2 strings play” a ROC curve area of 0.73. The class “Chord” achieved a ROC curve area of 0.88. The AUC scores are presented in Table 20.

Technique	**AUC score**
Trembling in one string	0.85
Trembling in two strings	0.80
Slurring	0.85
2 strings play	0.73
Chord	0.88
Micro-average	0.80
Macro-average	0.82

Table 20: AUC scores for model gradient boosting for mid-term window 0.6, mid-term step 0.2, short-term window 0.015 and short-term step 0.005

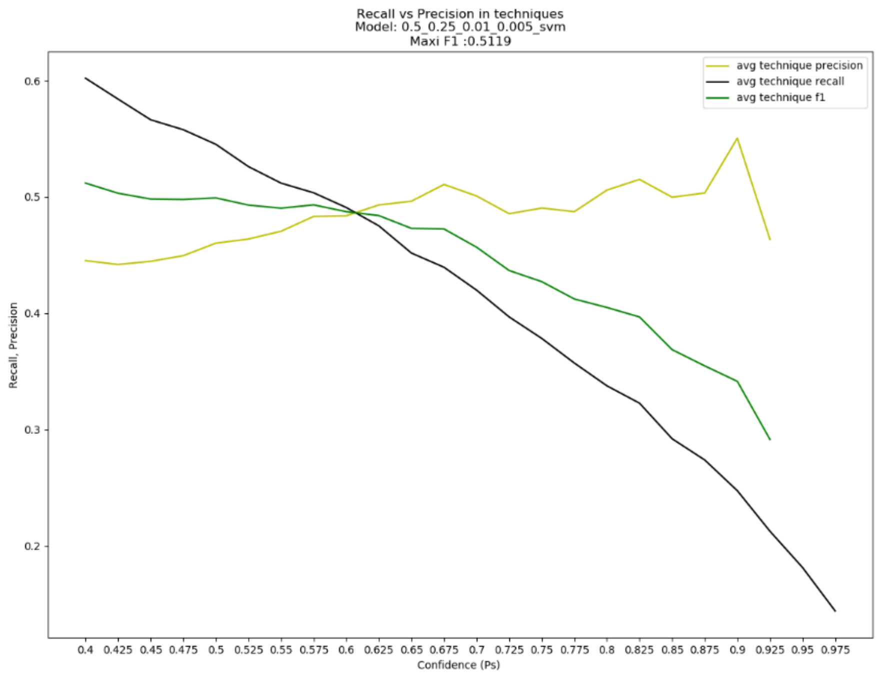


Figure 27: Recall vs Precision for model svm and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step)

The model svm for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005 achieved the highest F1 score (0.5119) at Confidence value 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of around 0.6. At the point of Precision, Recall and F1-Score equality (confidence 0.6), their values are 0.49. The confusion matrix of the model for all test recordings and confidence 0.6 is presented on Table 21 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	20	6	0	0	4
trembling2	0	87	0	2	4
slurring	9	10	68	34	6
2strings	26	56	28	235	4
chord	10	8	0	9	56

Table 21: Confusion Matrix of Techniques model svm for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005

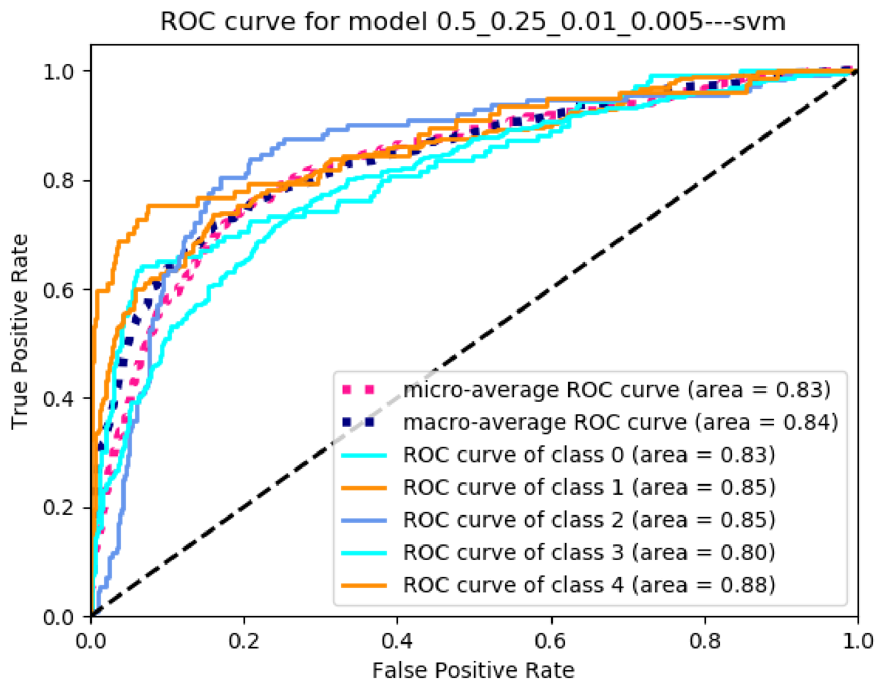


Figure 28: Roc curve for model svm and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step)

The model svm for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005 achieved a micro average ROC curve area of 0.83 and a macro-average ROC curve area of 0.84. The class “Trembling in one string” achieved a ROC curve area of 0.83 and “Trembling in two strings” achieved a ROC curve area of 0.85. The class “Slurring” achieved a ROC curve area of 0.85 and the class “2 strings play” a ROC curve area of 0.80. Finally, the class “Chord” achieved a ROC curve area of 0.88. The AUC scores are presented in Table 22.

Technique	**AUC score**
Trembling in one string	0.83
Trembling in two strings	0.85
Slurring	0.85
2 strings play	0.80
Chord	0.88
Micro-average	0.83
Macro-average	0.84

Table 22: AUC scores for model svm for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005

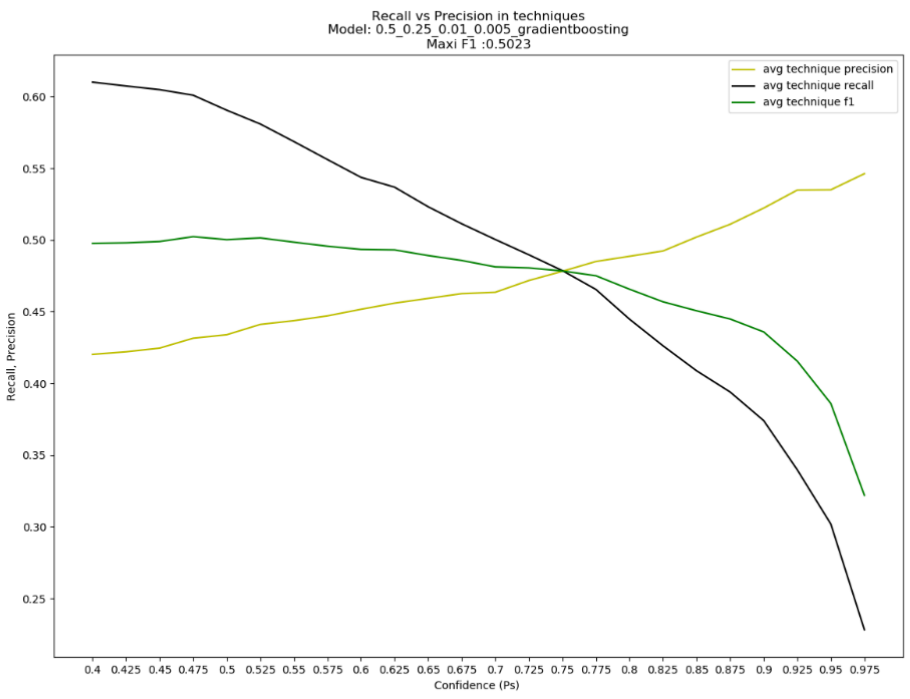


Figure 29: Recall vs Precision for model gradient boosting and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step)

The model gradient boosting for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005 achieved the highest F1 score (0.5023) at Confidence value 0.475. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of around 0.75. At the point of Precision, Recall and F1-Score equality (confidence 0.75), their values are 0.47. The confusion matrix of the model for all test recordings and confidence 0.75 is presented on Table 23 below.

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	39	5	1	0	4
trembling2	0	66	0	2	6
slurring	13	5	58	21	6
2strings	8	46	9	210	5
chord	11	45	0	9	55

Table 23: Confusion Matrix of Techniques model gradient boosting for mid-term window 0.5, mid-term step 0.25, short-term window 0.01 and short-term step 0.005

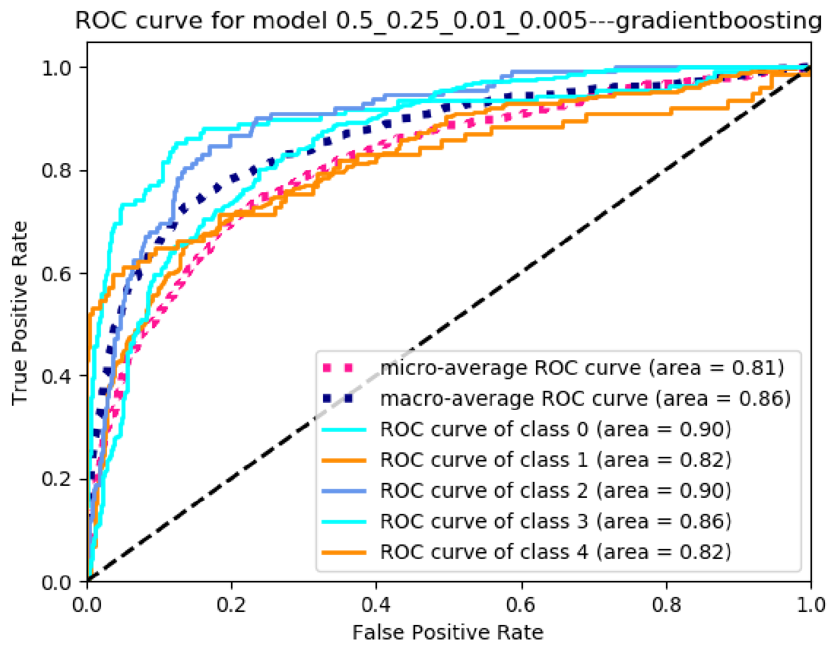


Figure 30: Roc curve for model gradient boosting and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step)

The model gradient boosting and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step) achieved a micro average ROC curve area of 0.81 and a macro-average ROC curve area of 0.86. Specifically, the class “Trembling in one string” achieved a ROC curve area of 0.90 and “Trembling in two strings” achieved a ROC curve area of 0.82. The class “Slurring” achieved a ROC curve area of 0.90 and the class “2 strings play” a ROC curve area of 0.86. The class “Chords” achieved a ROC curve area of 0.82. The AUC scores are presented in Table 24.

Technique	AUC score
Trembling in one string	0.90
Trembling in two strings	0.82
Slurring	0.90
2 strings play	0.86
Chord	0.82
Micro-average	0.81
Macro-average	0.86

Table 24: AUC scores for model gradient boosting and combination 0.5 (mt window), 0.25 (mt step), 0.01 (st window), 0.005 (st step)

II) Metrics for model with best F1-Score

In all cases, precision and recall are equal around 60%-70% confidence. The maximum F1 (~58%) is achieved by model “svm” at combination mt_win = 0.9, mt_step = 0.6, st_win = 0.06, st_step =0.02 at confidence ~40%. Recall and Precision are equal at confidence ~70%. The confusion matrix for this model at confidence 70% is:

True Predicted	trembling1	trembling2	slurring	2strings	chord
trembling1	14	7	0	0	1
trembling2	0	22	0	0	0
slurring	3	2	36	6	1
2strings	4	20	2	99	1
chord	0	0	0	0	25

Table 25: Confusion matrix for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step) only for techniques (confidence 70%)

True Predicted	None	trembling1	trembling2	slurring	2strings	chord
None	101	20	69	7	27	7
trembling1	3	14	7	0	0	1
trembling2	1	0	22	0	0	0
slurring	118	3	2	36	6	1
2strings	88	4	20	2	99	1
chord	23	0	0	0	0	25

Table 26: Confusion matrix for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step) for techniques and the None class

The above results give a total Accuracy of 42% including the None class. The average F1-score, average Precision and average Recall of techniques (None class excluded) are 55.8%. In general, the main misclassifications among techniques are between “Trembling in two strings” and “2 strings play”. This can be explained by the similarity of these two techniques. Both techniques have strikes in exact two strings but differ in the continuity of the strikes. However, in just very small segments, the continuous strike of 2 strings (trembling in two strings) might not be easy to discriminate. As a result, many segments with “trembling in two strings” were misclassified as “2 strings play”. In addition, the two trembling techniques (one string and two strings) have also technically a lot in common. The continuous and fast fashion of playing (in both techniques) might be the reason behind the misclassifications of trembling in two strings as trembling in one string. The confidence value of 0.7 can explain the significant amount of techniques classified as None class. In general, the probabilities of classifications of the model can have values smaller than 0.7. Thus, by raising the confidence of acceptance, a lot of classification might not be accepted and the None class is assigned in the position of the model's classification. Raising the confidence values makes the decisions of the classifier "stronger" in terms of probability but smaller probability classifications could also be true.

As of evaluation metrics for techniques, the class "Trembling in one string" achieved an F1-score of 42.42%, Recall of 34.15% and Precision of 56%. The class "Trembling in two strings" achieved an F1-score of 30.77%, Recall of 18.33% and Precision of 95.65%. The class "Slurring" achieved an F1-score of 34.12%, Recall of 80% and Precision of 21.69%. The class "2 strings play" achieved an F1-score of 57.23%, Recall of 75% and Precision of 46.26%. Finally, the class "Chords" achieved an F1-score of 60.24%, Recall of 71.43% and Precision of 52.08%. A graphical comparison between Recall, Precision and F1-Score among techniques for the model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step) are presented in Figure 31.

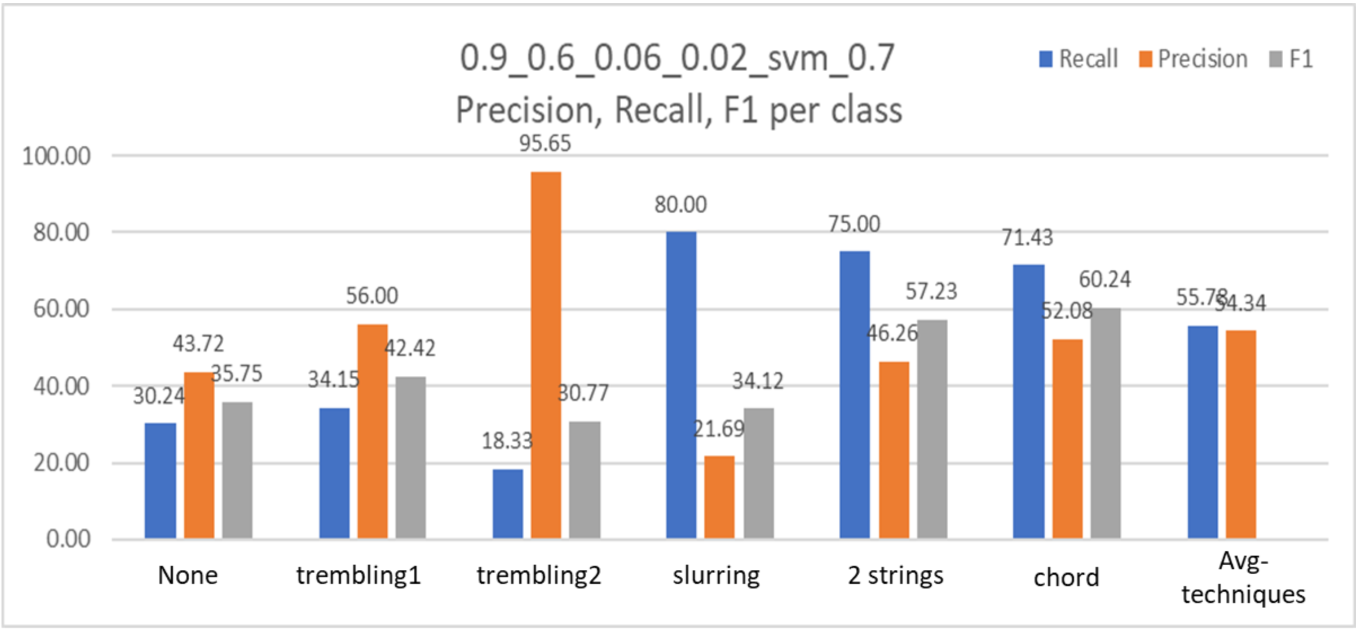


Figure 31: Precision, Recall, F1 per class for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

III) Results per model family

Out of the 5 models (SVM, K-NN, Gradient Boosting, Extra Trees and Random Forests), only SVM and Gradient Boosting achieved F1-scores and accuracies above 50%. The best 9 models were presented in Figures 15-30. However, a comparison between SVM and Gradient Boosting (Figure 32) shows that SVM achieves better F1-score for the same combination of mid-term window, mid-term step, short-term window and short-term step compared to Gradient Boosting. In addition, equality of Precision and Recall for same windows and steps combination is in lower confidence levels compared to Gradient Boosting.

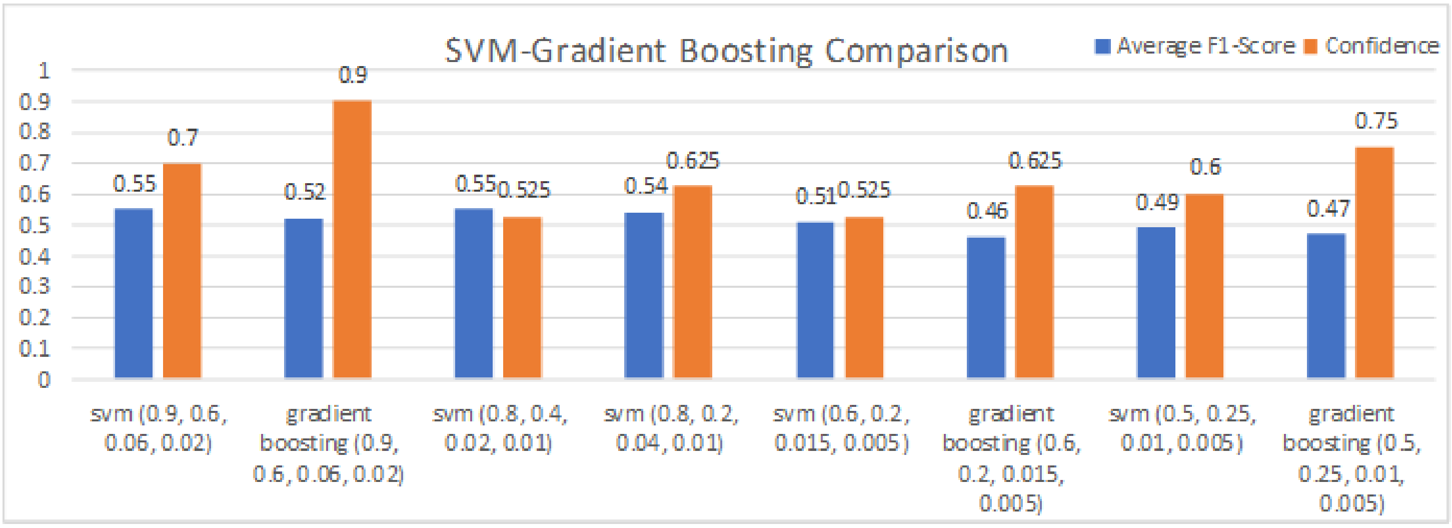


Figure 32: Comparison of F1-scores of models at confidence of Precision-Recall Equality

IV) Examination of predictions on ground truth of test recordings

For a focused evaluation of the models on the separate test recordings, the ground-truth vs predicted comparisons for each recording are presented in Figures 33-42.

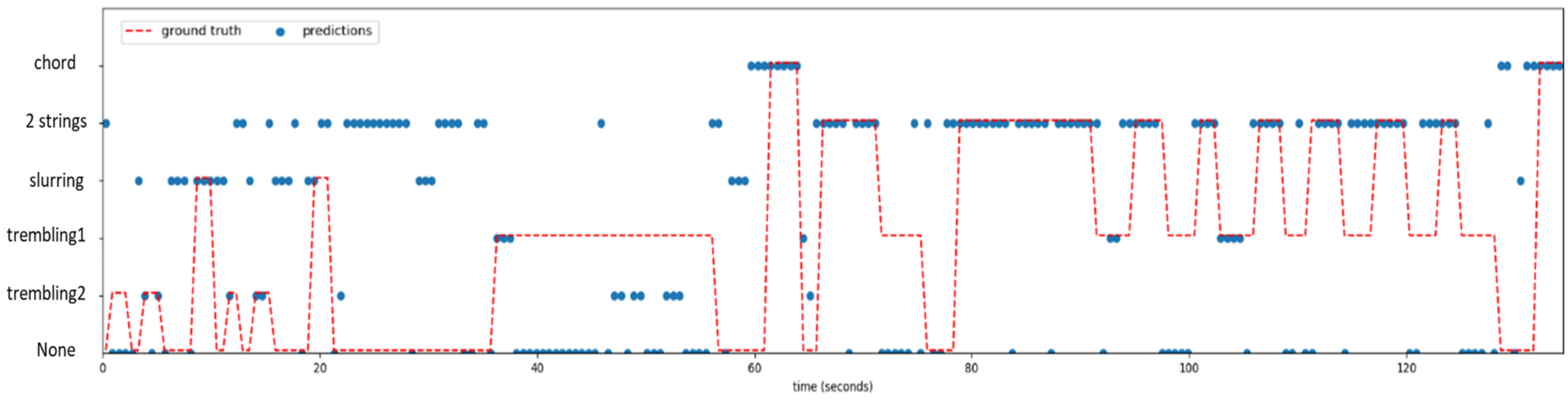


Figure 33: Ground truth vs predicted for test recording “Minore tis avgis” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

In Figure 33, the ground truth of the test recording “Minore tis avgis” is compared with the classifications of the approach. The red line refers to the ground truth of each segment (x-axis) and the blue dots to the classification of the approach. Red and blue values on top of the x-axis signify a None class. The model predicted correctly 91 segments (techniques and None) out of the total 225 segments giving an Accuracy of 40.44% (“Total” bar). The total normalized accuracy is 45.78% (“Normalized Total” bar). 78 out of the 160 total techniques were predicted correctly giving an Accuracy in Techniques of 48.75 (“Identified vs total techniques”) and a Normalized Accuracy in Techniques of 56.25% (“Normalized identified vs total identified” bar). The accuracy in None techniques is 20% (“identified none techniques” bar). This means that many None techniques (ground truth) were misclassified as a technique. Finally, the model predicted correctly 49.37% of the techniques with respect to the total identifications it made (“identified vs total identified" bar). The respective normalized accuracy is 56.96% ("Normalized identified vs total identified" bar). These values are presented in Figures 34 and 35.

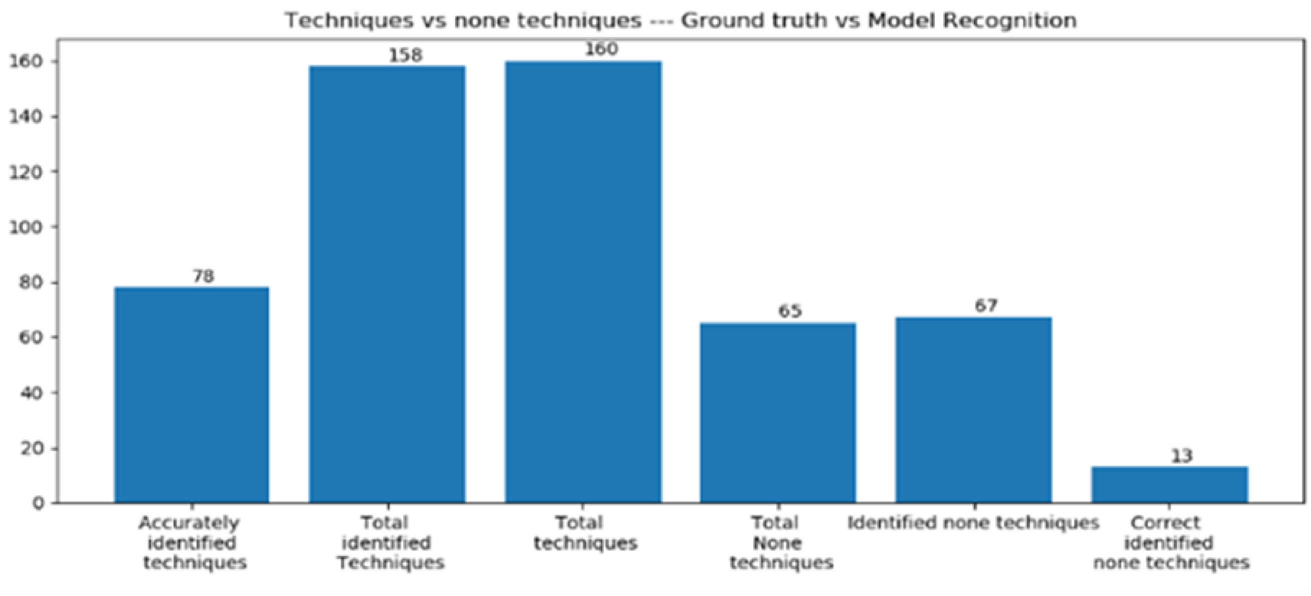


Figure 34: Evaluation metrics for test recording "Minore tis avgis" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

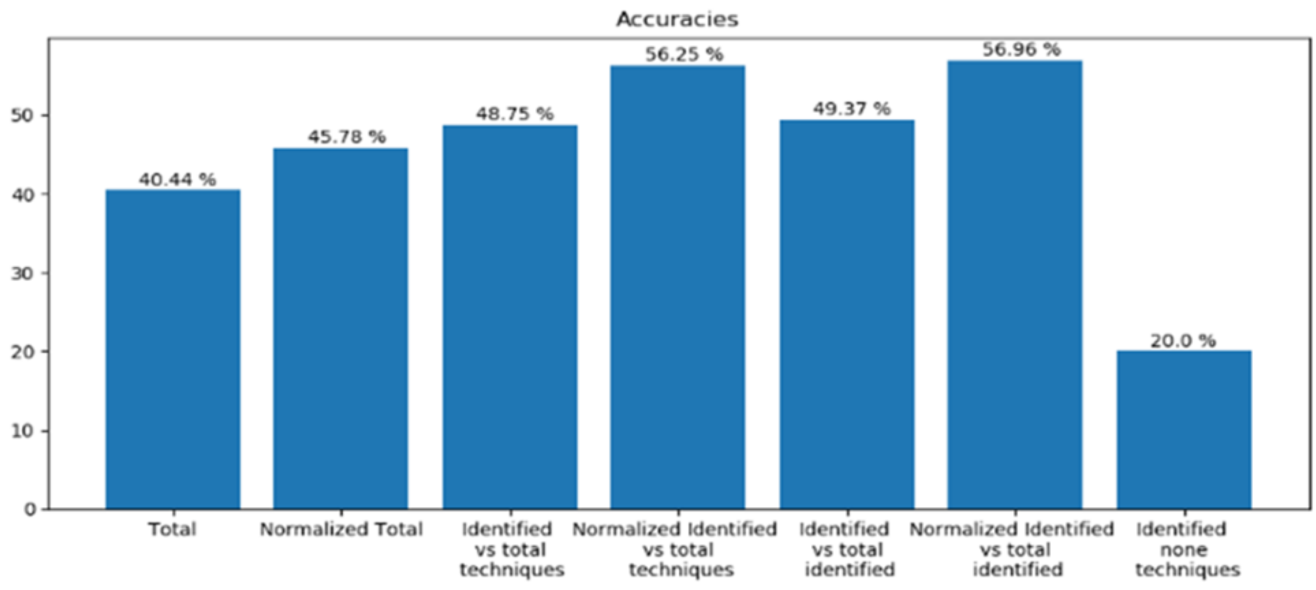


Figure 35: Accuracies for test recording "Minore tis avgis" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

In Figure 36, the ground truth of the test recording “Mi mou ksanafigeis pia” is compared with the classifications of the approach. The red line refers to the ground truth of each segment (x-axis) and the blue dots to the classification of the approach. Red and blue values on top of the x-axis signify a None class.

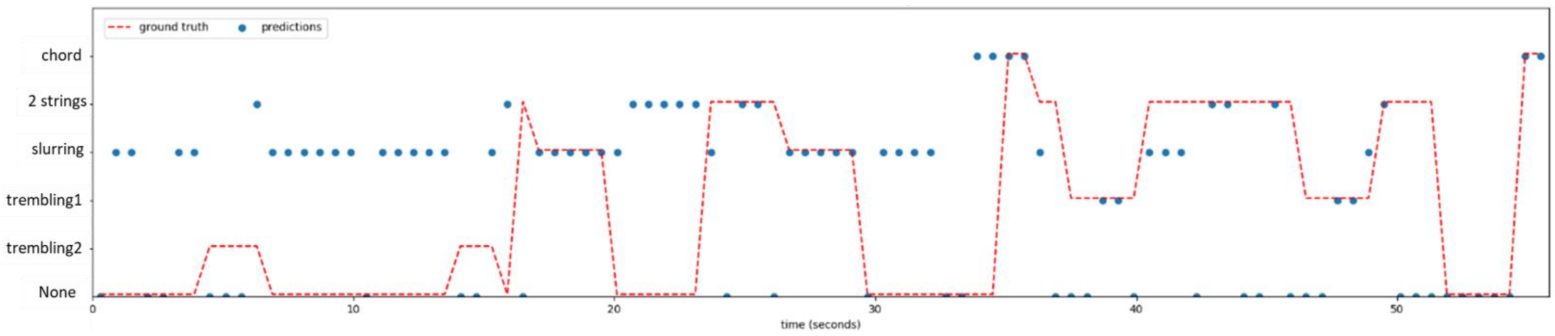


Figure 36: Ground truth vs predicted for test recording “Mi mou ksanafigeis pia” for svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model predicted correctly 36 segments (techniques and None) out of the 93 total segments giving an Accuracy of 38.71% (“Total” bar). The total normalized accuracy is 38.71% (“Normalized Total” bar). 24 out of the 53 total techniques were predicted correctly giving an Accuracy in Techniques of 45.28% (“Identified vs total techniques”) and a Normalized Accuracy in Techniques of 45.28% (“Normalized identified vs total identified” bar). Equality of these accuracies shows that no techniques with error weights lower than 1 were misclassified. The accuracy in None techniques is 30% (“identified none techniques” bar). This means that many None techniques (ground truth) were misclassified as a technique. Finally, the model predicted correctly 40% of the techniques with respect to the total identifications it made (“identified vs total identified” bar). The respective normalized accuracy is 40% (“Normalized identified vs total identified” bar). These values are presented in Figures 37 and 38.

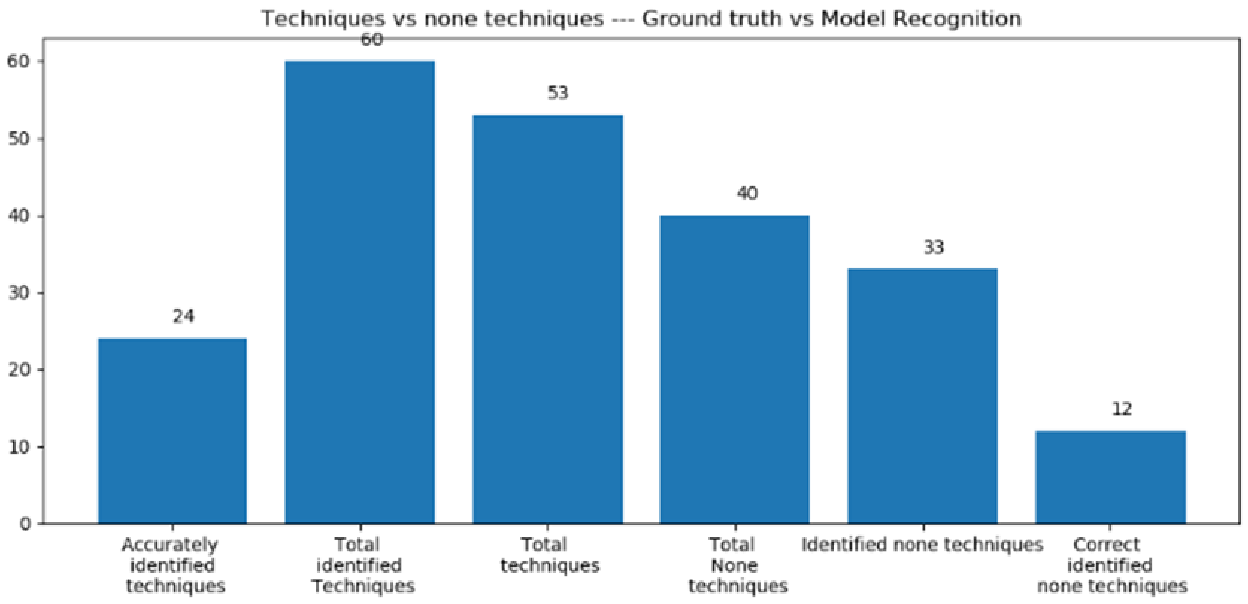


Figure 37: Evaluation metrics for test recording "Mi mou ksanafigeis pia" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

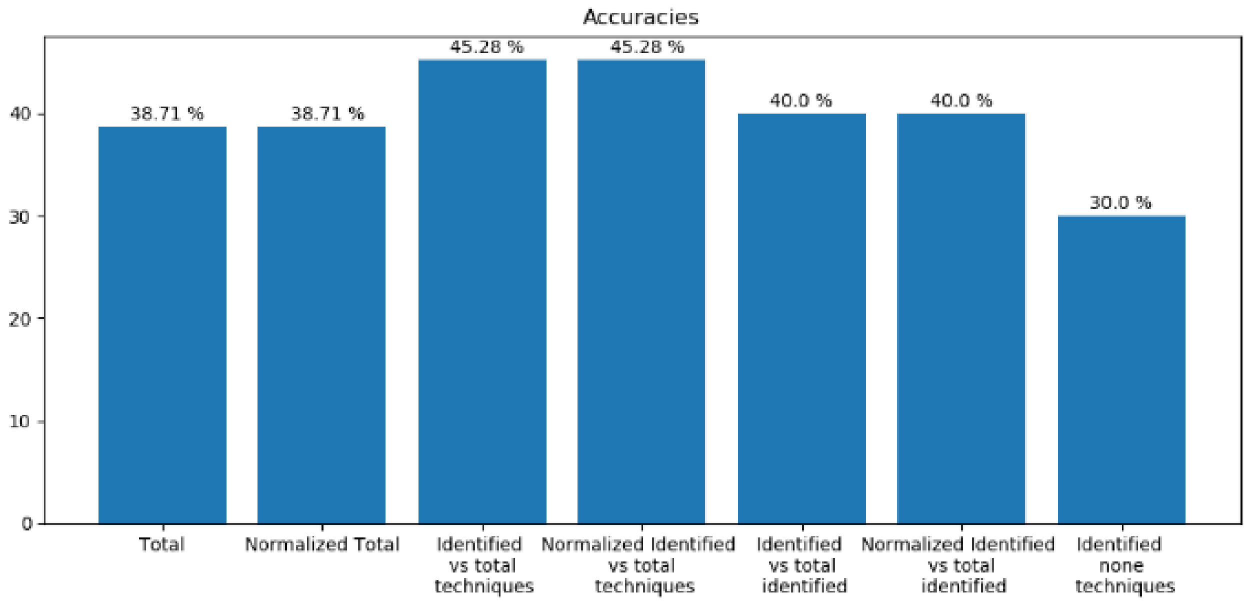


Figure 38: Accuracies for test recording "Mi mou ksanafigeis pia" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

In Figure 39, the ground truth of the test recording “Minore tou teke” is compared with the classifications of the approach. The red line refers to the ground truth of each segment (x-axis) and the blue dots to the classification of the approach. Red and blue values on top of the x-axis signify a None class.

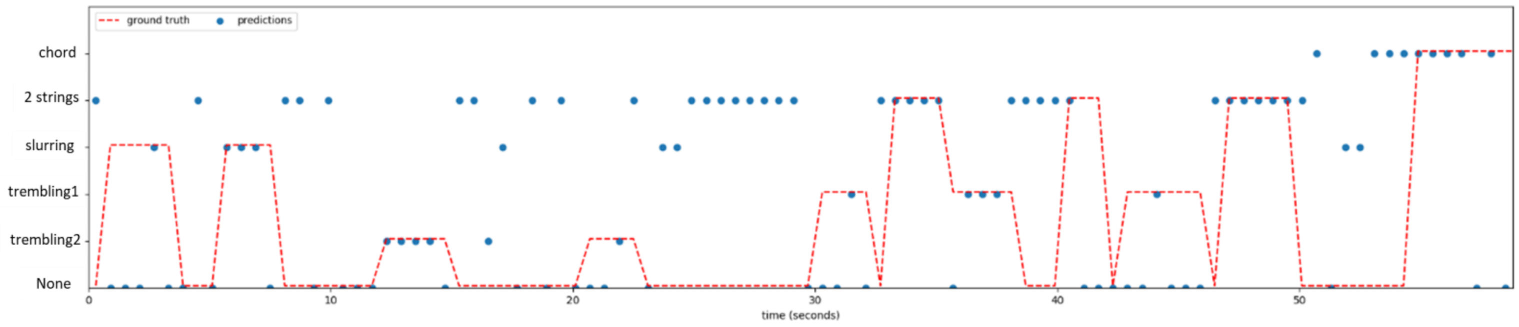


Figure 39: Ground truth vs predicted for test recording “Minore tou teke” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model predicted correctly 43 segments (techniques and None) out of the 99 total segments giving an Accuracy of 43.43% (“Total” bar). The total normalized accuracy is 43.94% (“Normalized Total” bar). 30 out of the 53 total techniques were predicted correctly giving an Accuracy in Techniques of 56.6% (“Identified vs total techniques”) and a Normalized Accuracy in Techniques of 57.55% (“Normalized identified vs total identified” bar). The accuracy in None techniques is 28.26% (“identified none techniques” bar). This means that many None techniques (ground truth) were misclassified as a technique. Finally, the model predicted correctly 46.15% of the techniques with respect to the total identifications it made (“identified vs total identified” bar). The respective normalized accuracy is 46.92% (“Normalized identified vs total identified” bar). These values are presented in Figures 40 and 41.

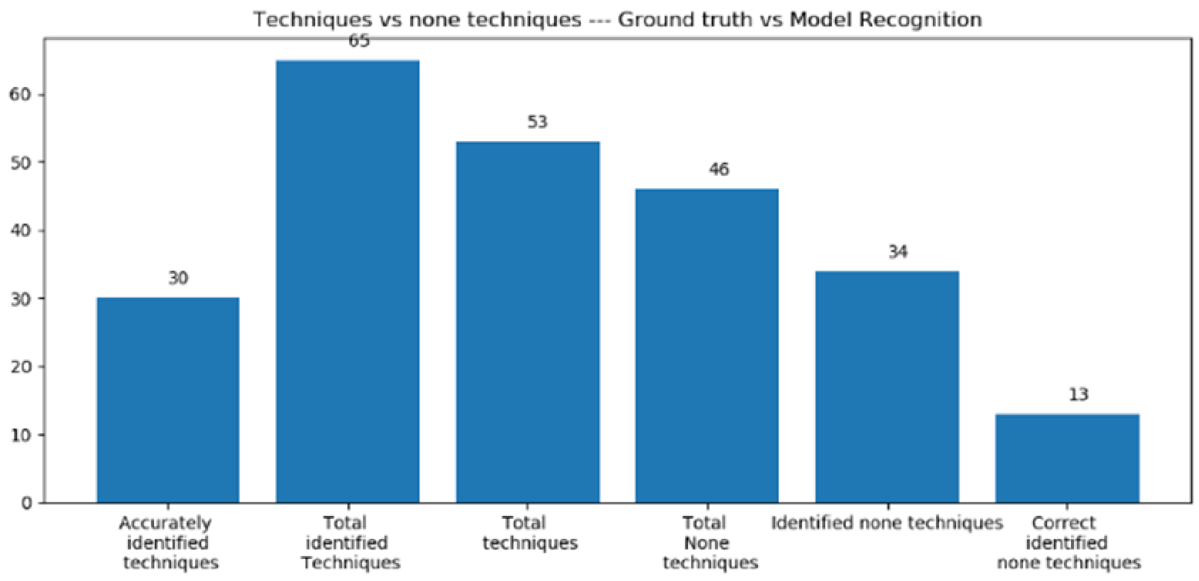


Figure 40: Evaluation metrics for test recording “Minore tou teke” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

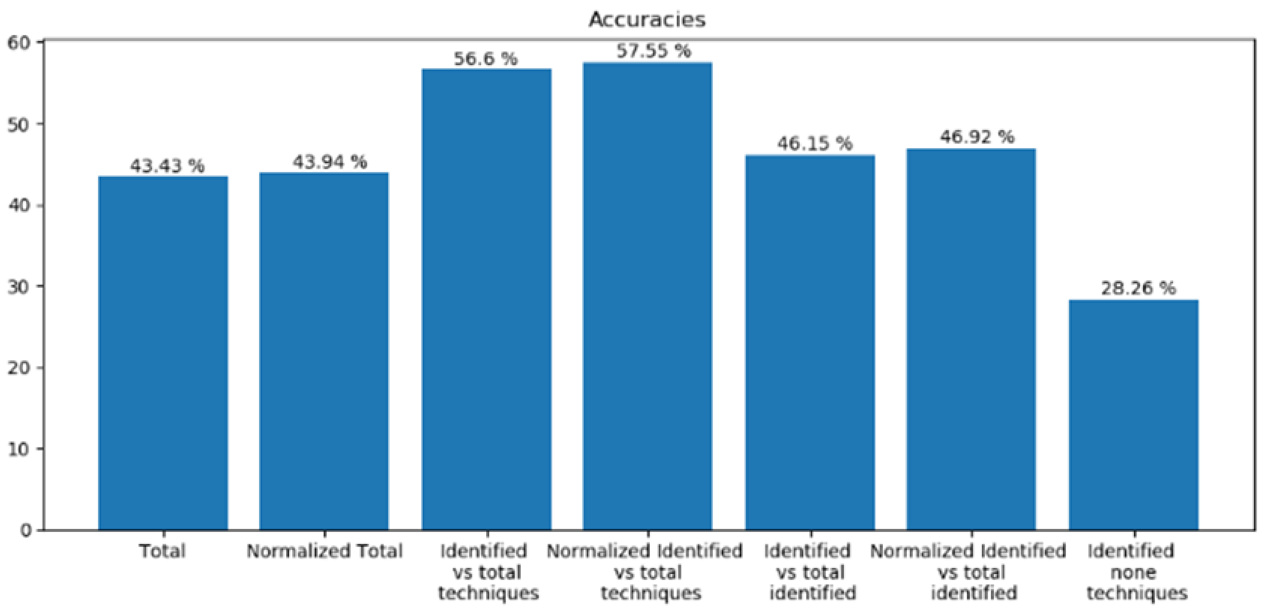


Figure 41: Accuracies for test recording “Minore tou teke” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

In Figure 42, the ground truth of the test recording “Taksimi moraiti” is compared with the classifications of the approach. The red line refers to the ground truth of each segment (x-axis) and the blue dots to the classification of the approach. Red and blue values on top of the x-axis signify a None class.

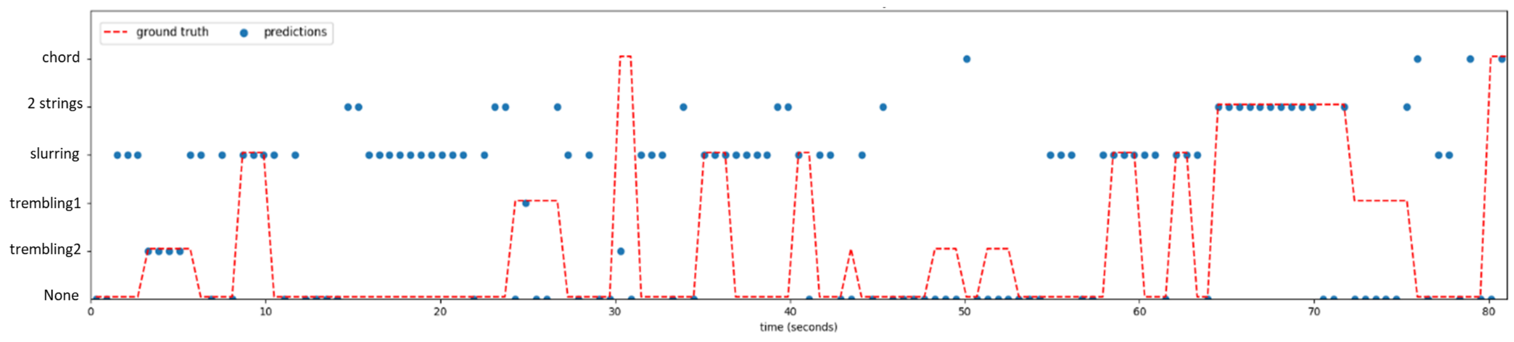


Figure 42: Ground truth vs predicted for test recording “Taksimi moraiti” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model predicted correctly 62 segments (techniques and None) out of the 136 total segments giving an Accuracy of 45.59% (“Total” bar). The total normalized accuracy is 46.32% (“Normalized Total” bar). 30 out of the 54 total techniques were predicted correctly giving an Accuracy in Techniques of 56.56% (“Identified vs total techniques”) and a Normalized Accuracy in Techniques of 57.41% (“Normalized identified vs total identified” bar). The accuracy in None techniques is 39.02% (“identified none techniques” bar). Finally, the model predicted correctly 35.71% of the techniques with respect to the total identifications it made (“identified vs total identified” bar). The respective normalized accuracy is 36.9% (“Normalized identified vs total identified” bar). These values are presented in Figures 43 and 44.

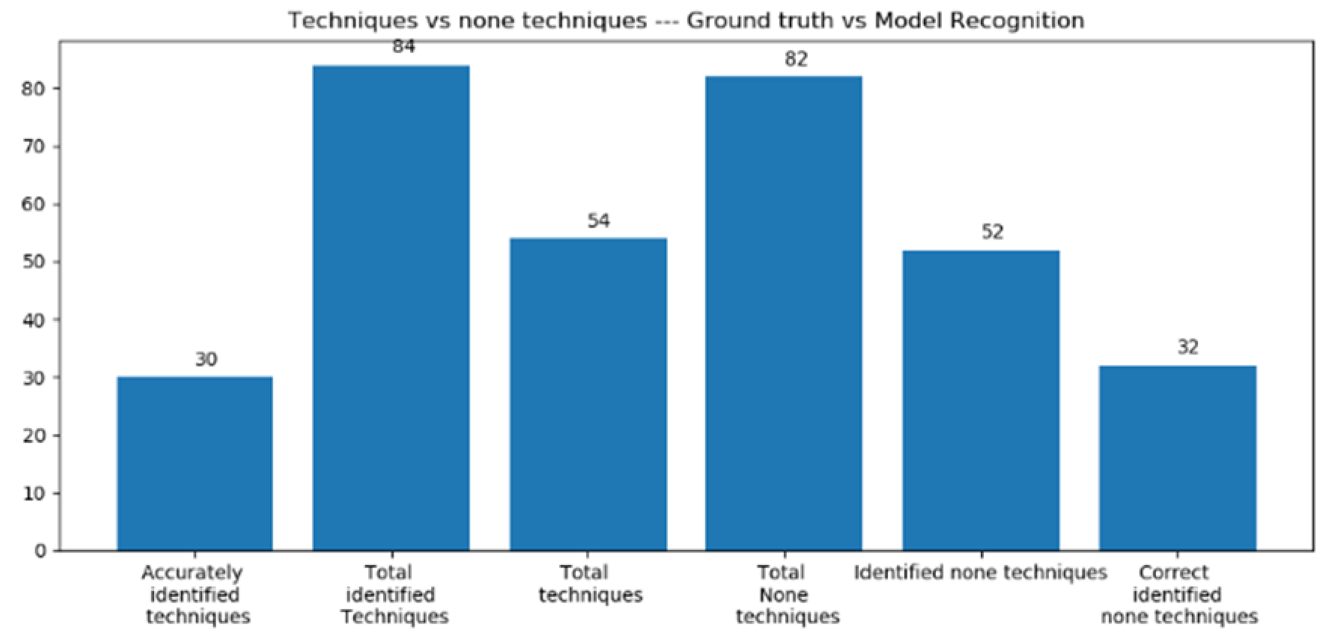


Figure 43: Evaluation metrics for test recording “Taksimi moraiti” for svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

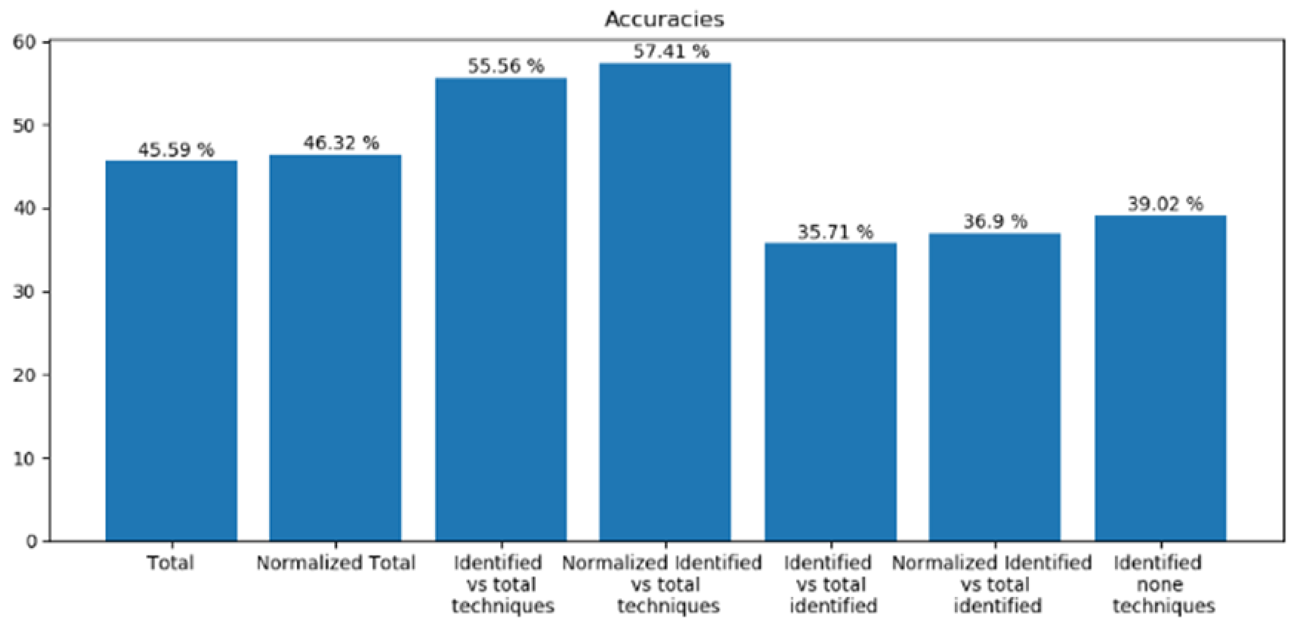


Figure 44: Accuracies for test recording “Taksimi moraiti” for svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

In Figure 45, the ground truth of the test recording “Prin to xarama” is compared with the classifications of the approach. The red line refers to the ground truth of each segment (x-axis) and the blue dots to the classification of the approach. Red and blue values on top of the x-axis signify a None class.

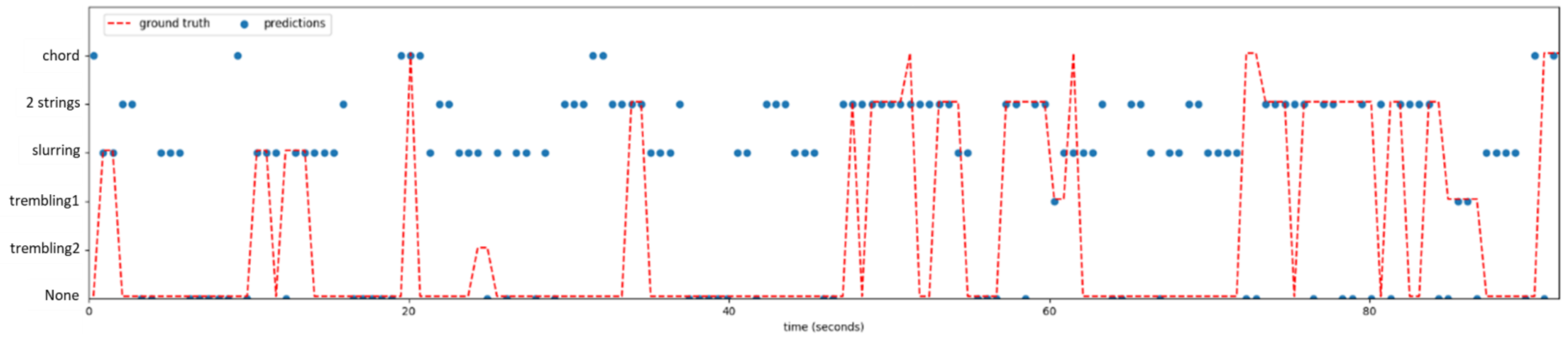


Figure 45: Ground truth vs predicted for test recording “Prin to xarama” for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

The model predicted correctly 65 segments (techniques and None) out of the 154 total segments giving an Accuracy of 42.21% (“Total” bar). The total normalized accuracy is 42.53% (“Normalized Total” bar). 34 out of the 53 total techniques were predicted correctly giving an Accuracy in Techniques of 64.15% (“Identified vs total techniques”) and a Normalized Accuracy in Techniques of 65.09% (“Normalized identified vs total identified” bar). The accuracy in None techniques is 30.69% (“identified none techniques” bar). Finally, the model predicted correctly 31.19% of the techniques with respect to the total identifications it made (“identified vs total identified” bar). The respective normalized accuracy is 31.65% (“Normalized identified vs total identified” bar). These values are presented in Figures 46 and 47.

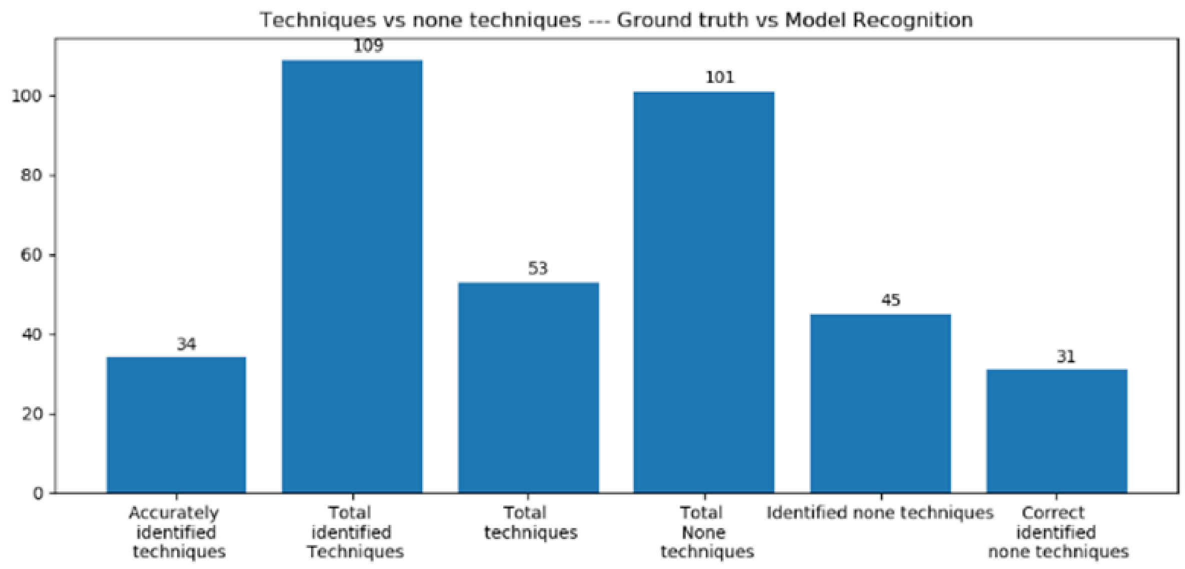


Figure 46: Evaluation metrics for test recording "Prin to xarama" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

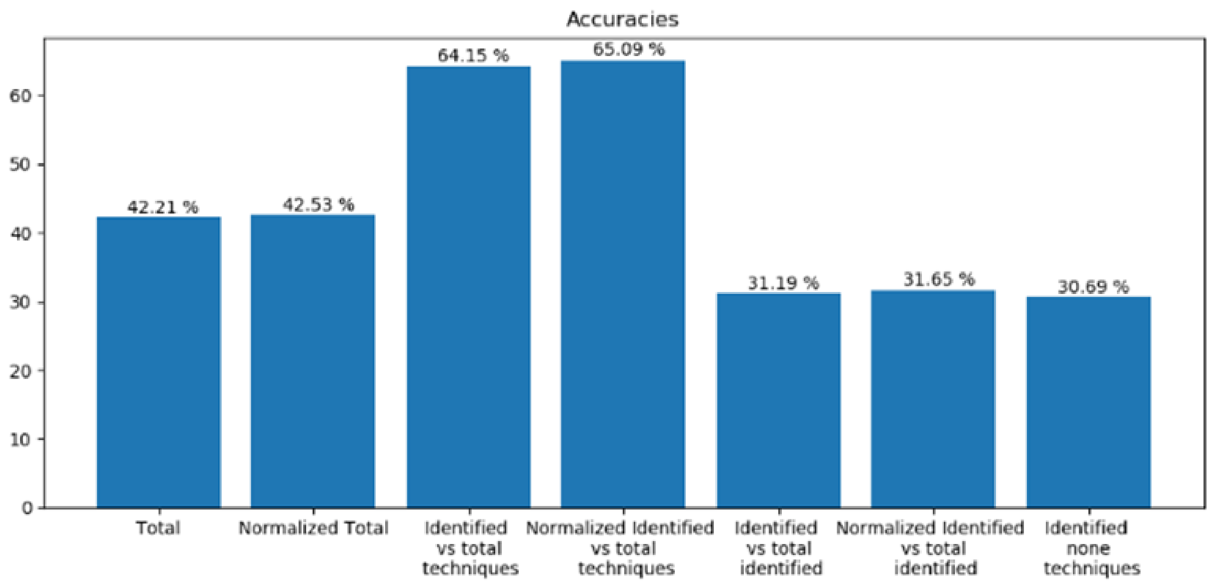


Figure 47: Accuracies for test recording "Prin to xarama" for model svm and combination 0.9 (mt window), 0.6 (mt step), 0.06 (st window), 0.02 (st step)

V) Feature Importances

In order to evaluate the contribution of the audio features in the classification task, the SVM coefficients and the feature importance values were calculated for the best 6 models -available in the open-source program- for SVM and Gradient Boosting models respectively. For the 4 SVM models, the SVM coefficients can help determine the feature importance.

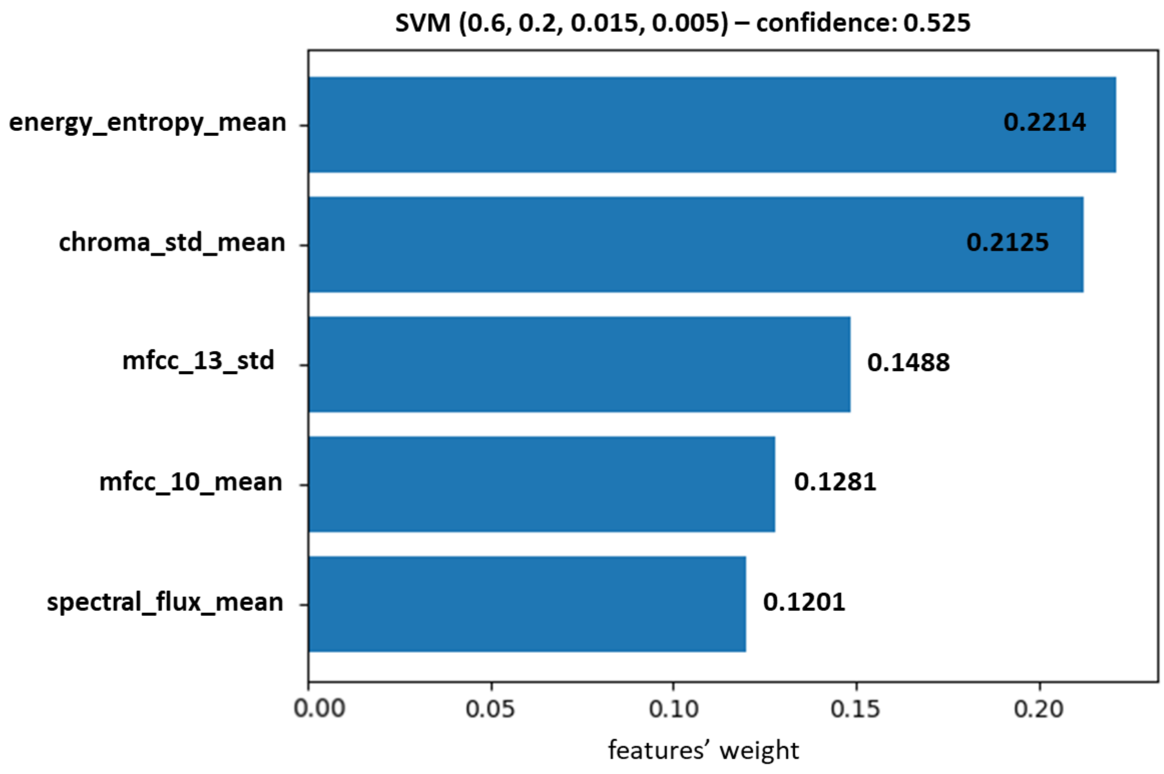


Figure 48: Five highest SVM coefficients for the model SVM with mid-term window 0.5, mid-term step 0.2, short-term window 0.015, short-term step 0.005 and confidence 0.525.

The SVM coefficient of the mean of Energy Entropy has the highest value and therefore has the highest contribution on the separation task. The mean of the standard deviation of the 12 chroma coefficients has a close value to the mean of Energy Entropy mean and therefore also plays an important role in the classification. The SVM coefficients of the model for the features of the standard deviation of the 13th and the mean of the 10th Mel Frequency Cepstral Coefficient have also high-ranked values. The mean of the Spectral Flux complements the top 5 SVM coefficients for the model SVM with mid-term step 0.5, mid-term window 0.2, short-term step 0.015, short-term window 0.005 and confidence 0.525.

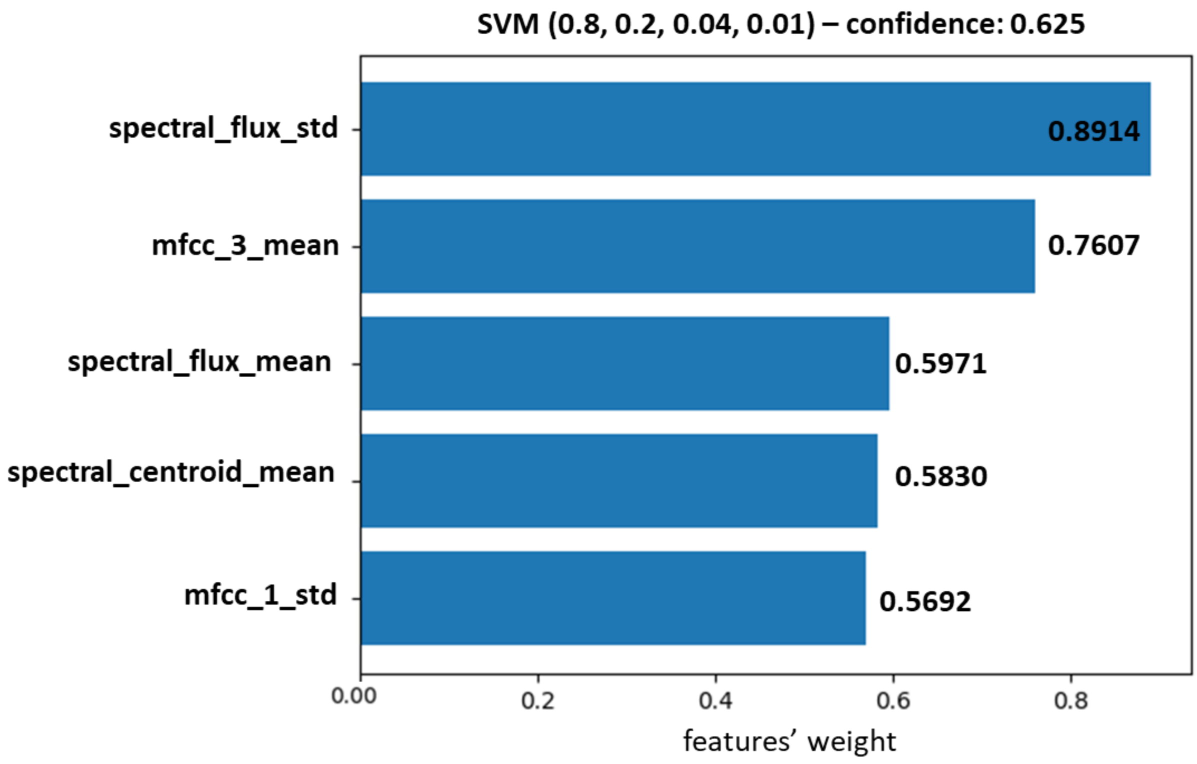


Figure 49: Five highest SVM coefficients for the model SVM with mid-term window 0.8, mid-term step 0.2, short-term window 0.04, short-term step 0.01 and confidence 0.625.

The SVM coefficient of the standard deviation of the Spectral Flux has the highest value and therefore has the highest contribution on the separation task. The mean of the 13th Mel Frequency Cepstral Coefficient has the second highest value. The SVM coefficients of the model for the features of the mean of the Spectral Flux and the mean of the Spectral Centroid have also high-ranked values. The standard deviation of the 1st Mel Frequency Cepstral Coefficient complements the top 5 SVM coefficients for the model SVM with mid-term window 0.8, mid-term step 0.2, short-term window 0.04, short-term step 0.01 and confidence 0.625.

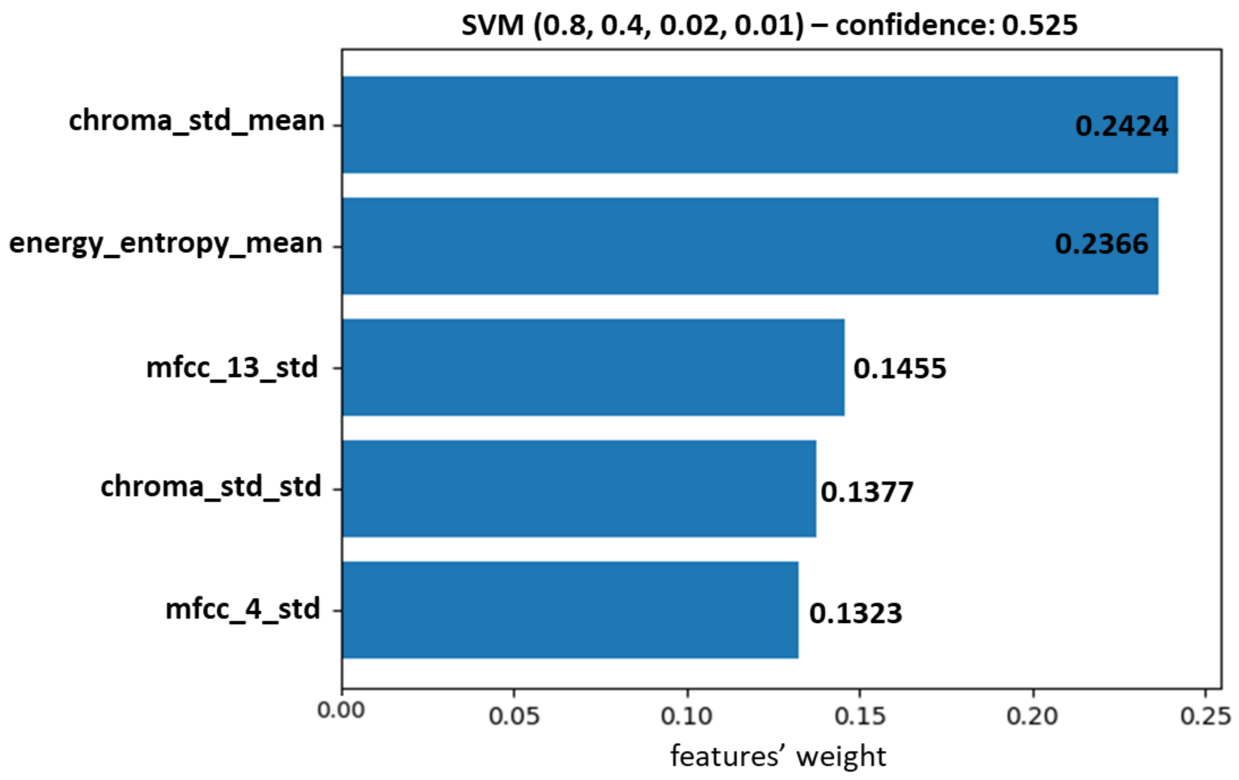


Figure 50: Five highest SVM coefficients for the model SVM with mid-term window 0.8, mid-term step 0.4, short-term window 0.02, short-term step 0.01 and confidence 0.525.

The SVM coefficient of the mean of the standard deviation of the 12 chroma coefficients has the highest value and therefore has the highest contribution on the separation task. The mean of the Energy Entropy has the second highest value. The SVM coefficients of the model for the features of the standard deviation of the 13th Mel Frequency Cepstral Coefficient and the standard deviation of the standard deviation of the 12 Chroma vectors have also high-ranked values. The standard deviation of the 4th Mel Frequency Cepstral Coefficient complements the top 5 SVM coefficients for the model SVM with mid-term window 0.8, mid-term step 0.4, short-term window 0.02, short-term step 0.01 and confidence 0.525.

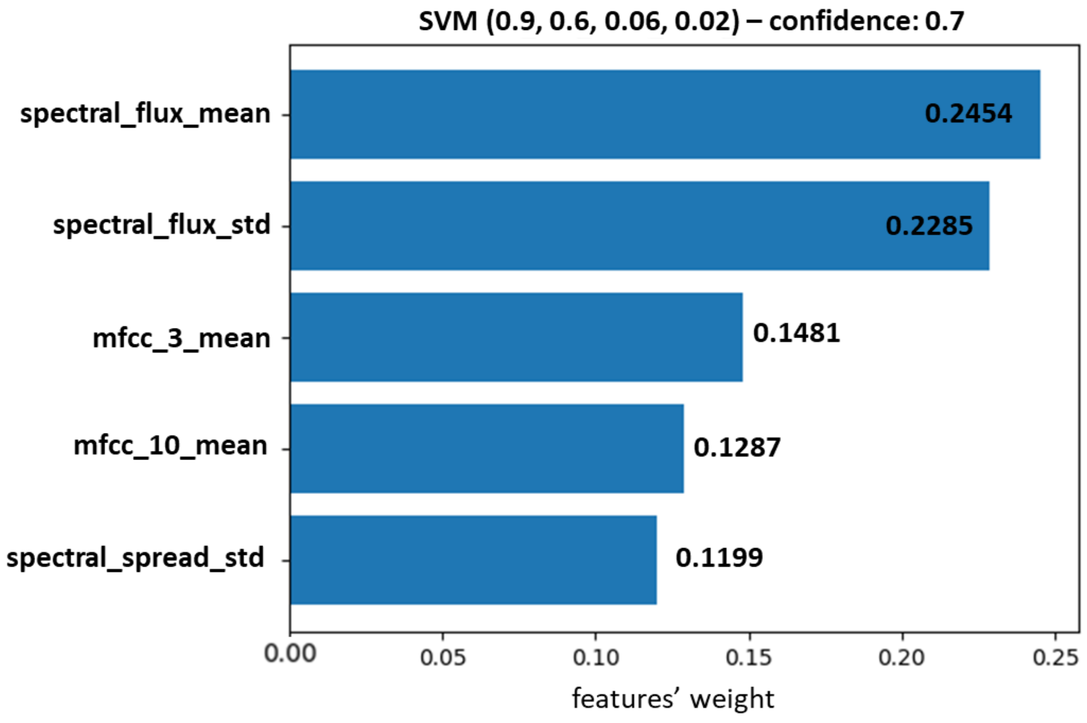


Figure 51: Five highest SVM coefficients for the model SVM with mid-term window 0.9, mid-term step 0.6, short-term window 0.06, short-term step 0.02 and confidence 0.7.

The SVM coefficient of the mean of the Spectral Flux has the highest value and therefore has the highest contribution on the separation task. The standard deviation of the Spectral Flux has the second highest value. The SVM coefficients of the model for the features of the mean of the 3rd and 10th Mel Frequency Cepstral Coefficients have also high-ranked values. The standard deviation of Spectral Spread complements the top 5 SVM coefficients for the model SVM with mid-term window 0.9, mid-term step 0.6, short-term window 0.06, short-term step 0.02 and confidence 0.7.

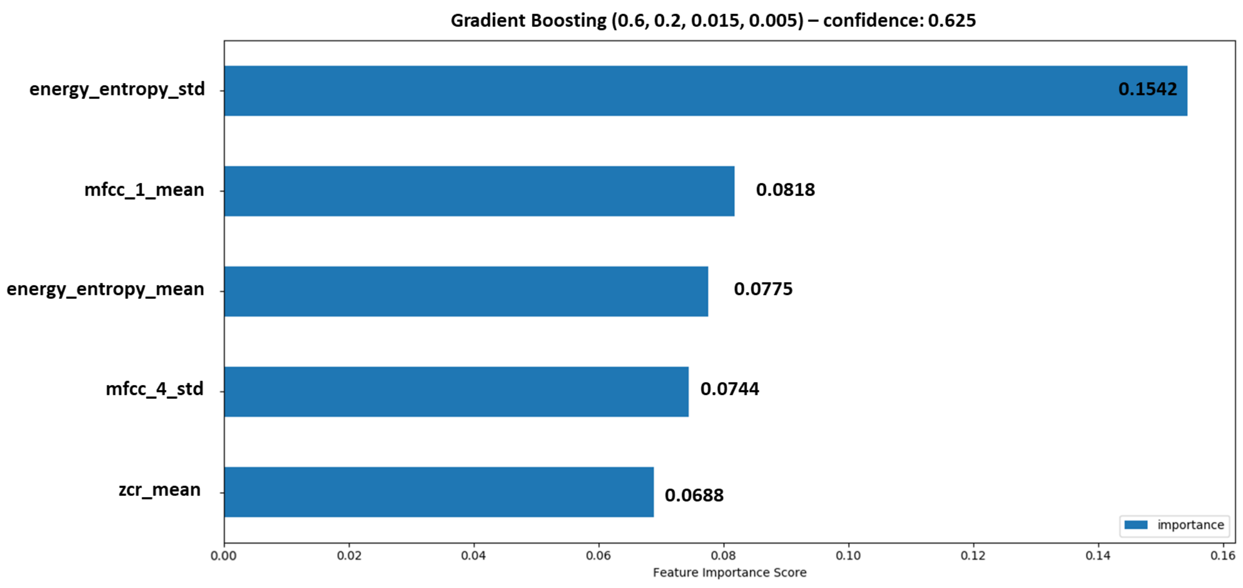


Figure 52: Five most important features for model Gradient Boosting with mid-term window 0.6, mid-term step 0.2, short-term window 0.015, short-term step 0.005 and confidence 0.625.

The most important feature for the model Gradient Boosting with mid-term window 0.6, mid-term step 0.2, short-term window 0.015, short-term step 0.005 and confidence 0.625 is the standard deviation of the Energy Entropy. Second most important feature is the mean of the 1st Mel Frequency Cepstral Coefficient. The mean of the Energy Entropy, the standard deviation of the 4th Mel Frequency Cepstral Coefficient and the mean of Zero Crossing Rate complement the 5 most important features of the model.

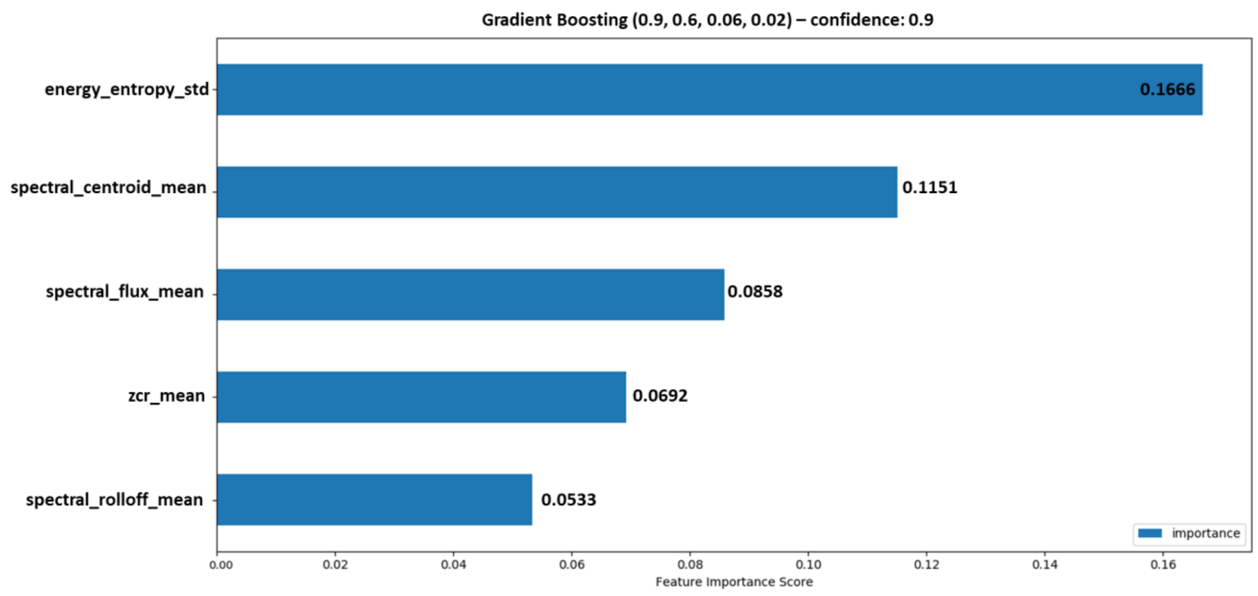


Figure 53: Five most important features for model Gradient Boosting with mid-term window 0.9, mid-term step 0.6, short-term window 0.06, short-term step 0.02 and confidence 0.9.

The most important feature for the model Gradient Boosting with mid-term window 0.9, mid-term step 0.6, short-term window 0.06, short-term step 0.02 and confidence 0.9 is the standard deviation of the Energy Entropy. Second most important feature is the mean of the Spectral Centroid. The mean of the Spectral Flux, the mean of the Zero Crossing Rate and the mean of the Spectral Roll-off complement the 5 most important features of the model.

7. CONCLUSION

Five models (SVM, K-NN, Random Forest, Gradient Boosting, Extra Trees) were trained under 8 different combinations of mid-term window, mid-term step, short-term window and short-term step and under different confidence values in order to classify segments of solo bouzouki recordings as of the playing techniques (5 techniques and a None technique class). Classifications of the model were accepted only if their probability was greater than the investigated confidence. In the case of being lower than the confidence, a 'None' class was assigned to the segment, signifying an absence of technique. Weighted errors were also introduced for the misclassifications of some techniques, which have some similarities. Thus, normalized values of the metrics (Normalized Accuracy) were also calculated. The model that achieved the best evaluation metrics was svm with mid-term window 0.9, mid-term step of 0.6, short-term window of 0.06 and short-term step of 0.02. The model svm for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved the highest F1 (0.5880) score of all models at Confidence value 0.4. As expected, recall drops and precision rises as the confidence limit rises. Precision and recall are equal at confidence value of 0.7. At the point of Precision, Recall and F1-Score equality (confidence 0.7), their values are 0.55. At confidence 0.7, the recall of 'None' class was 30.24%, the precision of 'None' class was 43.72% and respective F1-score was 35.75%. For "trembling in one string", Recall was 34.15%, Precision was 56.00% and F1-score 42.42% for confidence 0.7. Respectively, for "trembling in two strings", Recall was 18.33%, Precision was 95.65% and F1-score 30.77%. For "Slurring", Recall was 80%, Precision 21.69% and F1-score 34.12%. For "2 strings play", Recall was 75%, Precision 46.26% and F1-score of 57.23%. Finally, "Chords" achieved a Recall of 71.43%, Precision of 52.08% and F1-score of 60.24% at confidence 0.7. The average value of techniques in this confidence, Recall and Precision were ~55.5%. The model svm for mid-term window 0.9, mid-term step 0.6, short-term window 0.06 and short-term step 0.02 achieved a micro average ROC curve area of 0.83 and a macro-average ROC curve area of 0.85. Specifically, the class "Trembling in one string" achieved a ROC curve area of 0.74 and "Trembling in two strings" achieved a ROC curve area of 0.83. The class "Slurring" achieved a ROC curve area of 0.85 and the class "2 strings play" a ROC curve area of 0.86. Finally, the class “Chord” achieved a ROC curve area of 0.94. Gradient boosting achieved similar results (>50% F1-score). Random Forest and Extra Trees had much lower scores and K-NN behaved poorly not being able to separate classes. Normalization of accuracy and weighted errors do not add much or raise the metrics, as there were only few misclassifications of more similar techniques.

8. FUTURE WORK

In order to achieve better results and more accurate classifications (identifications) of instrument playing techniques, several methodologies are proposed in addition to the present study. Firstly, parameter tuning could include more parameters. In this case, only C (for svm) and boosting stages (for gradient boosting) were optimized during trained. The rest of parameters (for all investigated models) were used in their default values in all experiments. However, experimenting with more values and setting up an experiment to investigate (statistically) the influence of the parameters in the final results, could improve the accuracy metrics and generalize better. In addition, more models (Logistic Regression, Naïve Bayes, Deep Learning algorithms) can also be investigated. Furthermore, increasing the amount of training Data (more recordings in training set) is also a crucial aspect. Finally, extracting more audio features than the ones discussed in this approach will increase the feature vector in training process and could possibly deliver better results.

9. OPEN SOURCE INSTRUMENT PLAYING TECHNIQUE CLASSIFIER

For the purposes of this Thesis, an open-source standalone python script was developed which classifies each segment of a given solo bouzouki instrument recording as of the playing technique using the best 6 (pre-trained) models of this Thesis. The repository with the code and the pre-trained models of the project can be found in github [91]. The dependencies are pyAudioAnalysis, numpy, pandas and matplotlib [48, 88-90]. Only two arguments are required here. First, the path to the mp3 or wav audio recording must be passed. In addition, the path to the stored pre-trained models is also required. The output of the classifier script are 6 plots (one per model) that show the classification of each segment of each model. The pre-trained models available are presented in Table 27. These were the models that achieved the highest average F1-score in the current Thesis. The command line use is as below:

```
$ python classifier.py <path-to-input-audio> <path-to-pretrained-models>
```

model type	mid term window	mid term step	short term window	short term step	confidence
svm	0.6	0.2	0.015	0.005	52.5 %
svm	0.8	0.2	0.04	0.01	62.5%
svm	0.8	0.4	0.02	0.01	52.5 %
svm	0.9	0.06	0.06	0.02	70%
gradient boosting	0.6	0.2	0.015	0.005	62.5%
gradient boosting	0.9	0.06	0.06	0.02	90%

Table 27: Available pre-trained models in the open-source classifier

There are four functions that are called inside the script. First, the main() function analyzes the arguments passed and calls the function

FileClassification(input_file, model_name, model_type, confidence). FileClassification function has the same functionality as the mtFileClassification function of pyAudioAnalysis. This function needs the audio file path (input_file), the model path (model_name), the model type (model_type) and the confidence of classification acceptance (confidence), which are analyzed by main. Another function, called load_model() is called, which loads the pre-trained classifiers. Then, the input signal is analyzed with use of audioBasicIO of pyAudioAnalysis. Then, the audio features of the input signal are extracted with use of mtFeatureExtraction of the module audioFeatureExtraction of pyAudioAnalysis. For each segment, the classifierWrapper function of audioTrainTest module of pyAudioAnalysis makes a classification as of the technique. Note, that at this point, each segment is assigned with a specific technique and a probability of classification (Ps). The final classification with respect to the confidence level is computed in next step with use of plotSegmentationResults function. Finally, the function plotSegmentationResults(flags_ind, class_names, Ps, mt_win, mt_step, st_win, st_step, model_type, confidence) is called. The first argument (called flags_ind) are the indexes of the classes (0-5) that the previous function computed. In other words, flags_ind has a length of the total segments, where each element corresponds to the classification of FileClassification function to the respective segment. This argument is passed through FileClassification function. The argument "class_names" is a list with the names of the classes. The argument "Ps" is also passed through FileClassification and is a list of classification probabilities of each segment. According to this list, the final classification is made: If the probability of the classification of a segment is greater than the confidence of the model, than it is accepted. Otherwise, a None is assigned to this segment. The four next arguments (mt_win, mt_step, st_win, st_step) are the mid and short-term window and size parameters that were used during training of the respective pre-trained model. The "model_type" signifies the type of the model and is used for visualization purposes. Finally, the "confidence" argument is the confidence of acceptance of the respective pre-trained model and is used along with "Ps" for making the final classifications. This function first parses all classifications (flags_ind) and all classification probabilities (Ps) and checks if the classification probability assigned to each segment in previous steps is greater than the confidence of the current model used. In this case, this classification is accepted. Otherwise, the “None” technique is assigned to the segment. Finally, the results are plotted showing a scatter with x-axis being the time (seconds) and y-axis being the respective technique (or None). Each scattered point signifies a segment and its y value refers to the classification (Technique). This process is repeated for all pre-trained models available.

10. BIBLIOGRAPHY

[1] Downie, J. S. (2003). Music information retrieval. Annual review of information science and technology, 37(1), 295-340.

[2] Casey, M. A., Veltkamp, R., Goto, M., Leman, M., Rhodes, C., & Slaney, M. (2008). Content-based music information retrieval: Current directions and future challenges. Proceedings of the IEEE, 96(4), 668-696.

[3] Giannakopoulos, T., & Pikrakis, A. (2014). Introduction to audio analysis: a MATLAB® approach. Academic Press.

[4] Tzanetakis, G., & Cook, P. (2002). Manipulation, analysis and retrieval systems for audio signals. Princeton, NJ, USA: Princeton University.

[5] Kim, H. G., Moreau, N., & Sikora, T. (2006). MPEG-7 audio and beyond: Audio content indexing and retrieval. John Wiley & Sons.

[6] Tzanetakis, G., & Cook, P. (2000, October). Audio information retrieval (AIR) tools. In Proc. International Symposium on Music Information Retrieval.

[7] Li, T., & Ogihara, M. (2006). Toward intelligent music information retrieval. IEEE Trans. Multimedia, 8(3), 564-574.

[8] Bray, S., & Tzanetakis, G. (2005, September). Distributed Audio Feature Extraction for Music. In ISMIR (pp. 434-437).

[9] Tzanetakis, G., & Cook, P. (2002). Musical genre classification of audio signals. IEEE Transactions on speech and audio processing, 10(5), 293-302.

[10] Li, T. L., Chan, A. B., & Chun, A. H. (2010). Automatic musical pattern feature extraction using convolutional neural network. Genre, 10, 1x1.

[11] Choi, K., Fazekas, G., Sandler, M., & Cho, K. (2017, March). Convolutional recurrent neural networks for music classification. In 2017 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP) (pp. 2392-2396). IEEE.

[12] Costa, Y. M., Oliveira, L. S., & Silla Jr, C. N. (2017). An evaluation of convolutional neural networks for music classification using spectrograms. Applied soft computing, 52, 28-38.

[13] Silla Jr, C. N., Koerich, A. L., & Kaestner, C. A. (2008, September). The Latin Music Database. In ISMIR (pp. 451-456).

[14] Neumayer, R., & Rauber, A. (2007, April). Integration of text and audio features for genre classification in music information retrieval. In European Conference on Information Retrieval (pp. 724-727). Springer, Berlin, Heidelberg.

[15] Cano, P., Batlle, E., Kalker, T., & Haitsma, J. (2005). A review of audio fingerprinting. Journal of VLSI signal processing systems for signal, image and video technology, 41(3), 271-284.

[16] Doets, P. J. O. (2010). Modeling Audio Fingerprints: Structure, Distortion, Capacity.

[17] Giannakopoulos, T., Pikrakis, A., & Theodoridis, S. (2008, October). Music tracking in audio streams from movies. In 2008 IEEE 10th Workshop on Multimedia Signal Processing (pp. 950-955). IEEE.

[18] Giannakopoulos, T., & Petridis, S. (2012, August). Detection and clustering of musical audio parts using Fisher linear semi-discriminant analysis. In 2012 Proceedings of the 20th European Signal Processing Conference (EUSIPCO) (pp. 1289-1293). IEEE.

[19] Seyerlehner, K., Pohle, T., Schedl, M., & Widmer, G. (2007, September). Automatic music detection in television productions. In Proc. of the 10th International Conference on Digital Audio Effects (DAFx'07). SCRIME/LaBRI.

[20] Giannakopoulos, T. (2009). Study and application of acoustic information for the detection of harmful content, and fusion with visual information. Department of Informatics and Telecommunications, vol. PhD. University of Athens, Greece.

[21] Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2008). A speech/music discriminator of radio recordings based on dynamic programming and bayesian networks. IEEE Transactions on Multimedia, 10(5), 846-857.

[22] Pikrakis, A., Giannakopoulos, T., & Theodoridis, S. (2008). An overview of speech/music discrimination techniques in the context of audio recordings. In Multimedia Services in Intelligent Environments (pp. 81-102). Springer, Berlin, Heidelberg.

[23] Alexandre-Cortizo, E., Rosa-Zurera, M., & Lopez-Ferreras, F. (2005, November). Application of fisher linear discriminant analysis to speech/music classification. In EUROCON 2005-The International Conference on" Computer as a Tool" (Vol. 2, pp. 1666-1669). IEEE.

[24] Panagiotakis, C., & Tziritas, G. (2005). A speech/music discriminator based on RMS and zero-crossings. IEEE Transactions on multimedia, 7(1), 155-166.

[25] Papakostas, M., & Giannakopoulos, T. (2018). Speech-music discrimination using deep visual feature extractors. Expert Systems with Applications, 114, 334-344.

[26] Yang, W., Tu, W., Zheng, J., Zhang, X., Yang, Y., & Song, Y. (2018, February). An RNN-based speech-music discrimination used for hybrid audio coder. In International Conference on Multimedia Modeling (pp. 81-92). Springer, Cham.

[27] Celma, O. (2010). Music recommendation. In Music recommendation and discovery (pp. 43-85). Springer, Berlin, Heidelberg.

[28] Chen, H. C., & Chen, A. L. (2001, October). A music recommendation system based on music data grouping and user interests. In Proceedings of the tenth international conference on Information and knowledge management (pp. 231-238). ACM.

[29] Ricci, F., Rokach, L., & Shapira, B. (2011). Introduction to recommender systems handbook. In Recommender systems handbook (pp. 1-35). Springer, Boston, MA.

[30] Brusilovski, P., Kobsa, A., & Nejdl, W. (Eds.). (2007). The adaptive web: methods and strategies of web personalization(Vol. 4321). Springer Science & Business Media.

[31] Van den Oord, A., Dieleman, S., & Schrauwen, B. (2013). Deep content-based music recommendation. In Advances in neural information processing systems (pp. 2643-2651).

[32] Cano, P., Koppenberger, M., & Wack, N. (2005, November). Content-based music audio recommendation. In Proceedings of the 13th annual ACM international conference on Multimedia(pp. 211-212). ACM.

[33] Wang, X., & Wang, Y. (2014, November). Improving content-based and hybrid music recommendation using deep learning. In Proceedings of the 22nd ACM international conference on Multimedia (pp. 627-636). ACM.

[34] Bu, J., Tan, S., Chen, C., Wang, C., Wu, H., Zhang, L., & He, X. (2010, October). Music recommendation by unified hypergraph: combining social media information and music content. In Proceedings of the 18th ACM international conference on Multimedia (pp. 391-400). ACM.

[35] Briot, J. P., Hadjeres, G., & Pachet, F. (2017). Deep learning techniques for music generation-a survey. arXiv preprint arXiv:1709.01620.

[36] Boulanger-Lewandowski, N., Bengio, Y., & Vincent, P. (2012). Modeling temporal dependencies in high-dimensional sequences: Application to polyphonic music generation and transcription. arXiv preprint arXiv:1206.6392.

[37] Yang, L. C., Chou, S. Y., & Yang, Y. H. (2017). MidiNet: A convolutional generative adversarial network for symbolic-domain music generation. arXiv preprint arXiv:1703.10847.

[38] Mao, H. H., Shin, T., & Cottrell, G. (2018, January). DeepJ: Style-specific music generation. In 2018 IEEE 12th International Conference on Semantic Computing (ICSC) (pp. 377-382). IEEE.

[39] Oord, A. V. D., Dieleman, S., Zen, H., Simonyan, K., Vinyals, O., Graves, A., ... & Kavukcuoglu, K. (2016). Wavenet: A generative model for raw audio. arXiv preprint arXiv:1609.03499.

[40] Li, T., & Ogihara, M. (2003). Detecting emotion in music.

[41] Feng, Y., Zhuang, Y., & Pan, Y. (2003, October). Music information retrieval by detecting mood via computational media aesthetics. In Proceedings IEEE/WIC International Conference on Web Intelligence (WI 2003) (pp. 235-241). IEEE.

[42] Yang, Y. H., & Chen, H. H. (2010). Ranking-based emotion recognition for music organization and retrieval. IEEE Transactions on Audio, Speech, and Language Processing, 19(4), 762-774.

[43] Russell, J. A. (1980). A circumplex model of affect. Journal of personality and social psychology, 39(6), 1161.

[44] Yang, Y. H., Lin, Y. C., Su, Y. F., & Chen, H. H. (2008). A regression approach to music emotion recognition. IEEE Transactions on audio, speech, and language processing, 16(2), 448-457.

[45] Han, B., Rho, S., Jun, S. et al. Multimed Tools Appl (2010) 47: 433.

[46] Yang, Y. H., Liu, C. C., & Chen, H. H. (2006, October). Music emotion classification: A fuzzy approach. In Proceedings of the 14th ACM international conference on Multimedia (pp. 81-84). ACM.

[47] Tzanetakis, G., & Cook, P. (2000). Marsyas: A framework for audio analysis. Organised sound, 4(3), 169-175.

[48] Giannakopoulos, T. (2015). pyaudioanalysis: An open-source python library for audio signal analysis. PloS one, 10(12), e0144610.

[49] Mathieu, B., Essid, S., Fillon, T., Prado, J., & Richard, G. (2010, August). YAAFE, an Easy to Use and Efficient Audio Feature Extraction Software. In ISMIR (pp. 441-446).

[50] McFee, B., Raffel, C., Liang, D., Ellis, D. P., McVicar, M., Battenberg, E., & Nieto, O. (2015, July). librosa: Audio and music signal analysis in python. In Proceedings of the 14th python in science conference (Vol. 8).

[51] Bogdanov, D., Wack, N., Gómez Gutiérrez, E., Gulati, S., Boyer, H., Mayor, O., ... & Serra, X. (2013). Essentia: An audio analysis library for music information retrieval. In Britto A, Gouyon F, Dixon S, editors. 14th Conference of the International Society for Music Information Retrieval (ISMIR); 2013 Nov 4-8; Curitiba, Brazil.[place unknown]: ISMIR; 2013. p. 493-8.. International Society for Music Information Retrieval (ISMIR).

[52] Giannakopoulos, T., & Pikrakis, A. (2014). Introduction to audio analysis: a MATLAB® approach. Academic Press.

[53] Lartillot, O., & Toiviainen, P. (2007, September). A Matlab toolbox for musical feature extraction from audio. In International conference on digital audio effects (pp. 237-244).

[54] McKay, C., Fujinaga, I., & Depalle, P. (2005, September). jAudio: A feature extraction library. In Proceedings of the International Conference on Music Information Retrieval (pp. 600-3).

[55] Eyben, F., Wöllmer, M., & Schuller, B. (2010, October). Opensmile: the munich versatile and fast open-source audio feature extractor. In Proceedings of the 18th ACM international conference on Multimedia (pp. 1459-1462). ACM.

[56] Martin, K. D., & Kim, Y. E. (1998). Musical instrument identification: A pattern-recognition approach. The Journal of the Acoustical Society of America, 104(3), 1768-1768.

[57] Eronen, A. (2001, October). Comparison of features for musical instrument recognition. In Proceedings of the 2001 IEEE Workshop on the Applications of Signal Processing to Audio and Acoustics (Cat. No. 01TH8575) (pp. 19-22). IEEE.

[58] Eronen, A., & Klapuri, A. (2000, June). Musical instrument recognition using cepstral coefficients and temporal features. In 2000 IEEE International Conference on Acoustics, Speech, and Signal Processing. Proceedings (Cat. No. 00CH37100)(Vol. 2, pp. II753-II756). IEEE.

[59] Essid, S., Richard, G., & David, B. (2006). Musical instrument recognition by pairwise classification strategies. IEEE Transactions on Audio, Speech, and Language Processing, 14(4), 1401-1412.

[60] Eronen, A. (2003, July). Musical instrument recognition using ICA-based transform of features and discriminatively trained HMMs. In Seventh International Symposium on Signal Processing and Its Applications, 2003. Proceedings. (Vol. 2, pp. 133-136). IEEE.

[61] Kaminsky, I., & Materka, A. (1995, November). Automatic source identification of monophonic musical instrument sounds. In Proceedings of ICNN'95-International Conference on Neural Networks (Vol. 1, pp. 189-194). IEEE.

[62] Eronen, A. (2001). Automatic musical instrument recognition. Mémoire de DEA, Tempere University of Technology, 178.

[63] Lostanlen, V., Andén, J., & Lagrange, M. (2018). Extended playing techniques: The next milestone in musical instrument recognition. arXiv preprint arXiv:1808.09730.

[64] Metcalf, C. D., Irvine, T. A., Sims, J. L., Wang, Y. L., Su, A. W., & Norris, D. O. (2014). Complex hand dexterity: a review of biomechanical methods for measuring musical performance. Frontiers in psychology, 5, 414.

[65] http://forumnet.ircam.fr/product/orchids-en/

[66] Bernays, M., & Traube, C. (2013). Expressive production of piano timbre: touch and playing techniques for timbre control in piano performance. In Proceedings of the 10th Sound and Music Computing Conference (SMC2013) (pp. 341-346). Stockholm, Sweden: KTH Royal Institute of Technology.

[67] Wang, C., Benetos, E., Meng, X., & Chew, E. (2019, May). HMM-based Glissando Detection for Recordings of Chinese Bamboo Flute. Sound and Music Computing Conference.

[68] Chen, Y. P., Su, L., & Yang, Y. H. (2015, October). Electric Guitar Playing Technique Detection in Real-World Recording Based on F0 Sequence Pattern Recognition. In ISMIR (pp. 708-714).
[69] Young, D. (2008, June). Classification of Common Violin Bowing Techniques Using Gesture Data from a Playable Measurement System. In NIME (pp. 44-48).
[70] Loureiro, A. M., & Hugo, B. (2004). de Paula, Yehia HC Timbre classification of a single musical instrument. CEFALA: Brazil.
[71] Reboursière, L., Lähdeoja, O., Drugman, T., Dupont, S., Picard-Limpens, C., & Riche, N. (2012, May). Left and right-hand guitar playing techniques detection. In NIME.
[72] Su, L., Yu, L. F., & Yang, Y. H. (2014, October). Sparse Cepstral, Phase Codes for Guitar Playing Technique Classification. In ISMIR (pp. 9-14).
[73] Abeßer, J., Lukashevich, H., & Schuller, G. (2010, March). Feature-based extraction of plucking and expression styles of the electric bass guitar. In 2010 IEEE International Conference on Acoustics, Speech and Signal Processing (pp. 2290-2293). IEEE.
[74] https://en.wikipedia.org/wiki/Tremolo
[75] https://en.wikipedia.org/wiki/Alternate_picking
[76] https://en.wikipedia.org/wiki/Hammer-on
[77] http://libsearch.teiep.gr/Record/1%2F655
[78] http://libsearch.teiep.gr/Record/1%2F762
[79] http://libsearch.teiep.gr/Record/1%2F13990
[80] http://libsearch.teiep.gr/Record/1%2F34498
[81] http://libsearch.teiep.gr/Record/1%2F31663
[82] https://github.com/tyiannak/multimodalAnalysis
[83] Giannakopoulos, T., Smailis, C., Perantonis, S. J., & Spyropoulos, C. D. (2014, August). Realtime depression estimation using mid-term audio features. In AI-AM/NetMed@ ECAI (pp. 41-45).
[84] Xu, M., Duan, L. Y., Cai, J., Chia, L. T., Xu, C., & Tian, Q. (2004, November). HMM-based audio keyword generation. In Pacific-Rim Conference on Multimedia (pp. 566-574). Springer, Berlin, Heidelberg.
[85] https://github.com/kostispar/ThesisData/tree/master/Data

[86] https://github.com/kostispar/ThesisData/upload/master/testData

[87] https://github.com/kostispar/ThesisData/blob/master/training_testing.py

[88] https://numpy.org/

[89] https://pandas.pydata.org/

[90] https://matplotlib.org/

[91] https://github.com/kostispar/IPTSegmentClassifier