MINISTRY OF EDUCATION

TECHNICAL UNIVERSITY
OF CLUJ-NAPOCA, ROMANIA

Electronics, Telecommunications and Informational Technology

PhD THESIS

– SUMMARY –

Enhancing robots emotion recognition algorithms using speech based signals

PhD Student:
Toma TELEMBICI

PhD Supervisor:
Prof. Ing. Corneliu RUSU, PhD

Examination committee:
Chair: Prof. Eng. **Sorin Hintea**, PhD – Technical University of Cluj-Napoca;
PhD Supervisor: Prof. Eng. **Corneliu Rusu**, PhD – Technical University of Cluj-Napoca;
Members:
-Prof. Eng. **Corneliu Burileanu**, PhD – Politehnica Bucharest National University of Science and Technology
-Prof. Eng. **Corina Naforniță**, PhD – The Polytechnic University of Timisoara
-Assoc. Prof. Eng. **Lăcrimioara Grama**, PhD – Technical University of Cluj-Napoca;

– Cluj-Napoca –
2024

TABLE OF CONTENTS

Introduction

Because humans are emotional beings, everything that enters in contact with us seems lifeless and cold if emotion is not present. As robots have a major impact on our everyday life there is a visible necessity in making them more human-like. As the field of voice recognition and speaker recognition reaches its peak, the next field that its naturally to be under research is the field of emotion recognition. This field is relatively new, and efforts are being made in order to raise the quality of the results and the available resources in this field.

As we know there are numerous countries that push the development of the field of emotion recognition. There are at the moment powerful emotion recognition algorithms based on image recognition that are used in the security field mainly in the USA and in China. The field of emotion recognition based on voice patterns is still underdeveloped but there are a lot of countries developing databases and searching for practical algorithms.

The title of the PhD thesis is Enhancing robots' emotion recognition algorithms using speech-based signals. As we already mentioned the emotion plays an important aspect in how humans perceive different encounters with robots or things. There are categories of people who have limited interaction with other humans from safety, healthy or other reasons. For them an interaction with emotional robots can have a big impact on their mental health and have a swing in their current emotional state. The objective of this research is to create an emotional database in Romanian language in order to include people from a demographic area into the potential beneficiary of the emotional robots. Another objective is to find an algorithm that has good results on the already existent databases and to test the new database to see if it can be included with the others in the research part.

In order to achieve all our objectives, first of all we tried to get up to date to the all-new research that have been done in the field. Then we tried to see what emotions are, how they are produced, what are the emotion families and what are the emotion borders. In order to create our own emotion database, we needed to determine what emotions are usually present in our everyday life and what sentences can have different meanings when they are said under different emotions. In order to create an algorithm that is performant in the emotional speech field we needed to research what feature extractors and what classifiers are used in this domain. We have picked one classifier and then tested different feature extractors to see the result obtained by those combinations using all the test databases and the new database that we have created. After we have eliminated the worst feature extractors, we have added more classifiers to see what combinations can give a powerful algorithm. In the end we have created a model that was tested on the freshly created database.

In the current stage of knowledge part of the thesis, the starting chapter is the "Robots" chapter, where the different commercial or service robots are briefed in order to see what we have to work with and what is the development directorate. The next chapter in the thesis is entitled "Emotional Databases" where we define what an emotional database is, what are the basic emotions, what are the contrasting basic emotions and what groups of emotions are there. In this chapter we also say what is the importance of creating a new database. In the next chapter we cover what databases are used as test in this work and how are they composed. Next, we talk about the different types of extracted characteristics and what classifiers are used.

In the personal contribution part of the thesis, first we talk about how we created the new database, what is the content of the database and some analysis done based on its content. The next chapter shows the implementation and the results obtained using only one selected classifier. The thesis continues with the implementation of the rest of the classifiers

and their results. After that we have a look on other results obtained in the same field and compare with the results obtained by our research. At the end we have our final conclusions based on the work done so far followed by the references list of figures and list of tables.

Creation of the database

The emotional database has been thought and designed in such a way that each sentence can have different meanings depending on the emotion conveyed by that sentence. The sentences are proposed to be understood and used by a service bot used in an indoor environment. The proposed sentences are as follows:

"The test results are in."
"I'm out of medication."
"The test is positive."
"The interpretation was wrong."
"It's grown a lot."
"It's gone down a lot."
"I closed the door on him."
"I turned off the light."
"I'm very well."
"The keys are in the door."

In order to better understand each sentence, we will look at the situations in which the sentences might be used and the different emotions with which they might be interpreted.

The first sentence, "The test result is in," said in a happy tone, may indicate an escape from illness, an expected pregnancy test, or a positive test result. This answer may also come as a surprise to the speaker. The first sentence spoken in a neutral tone may indicate that the answer received is not a surprise, but rather comes as a validation to the speaker. When this sentence is uttered with a sad emotion, it may mean the speaker has received bad news, such as the discovery or aggravation of an illness, or a negative result of an exam. By knowing these emotions and deciphering the hidden message, the service bot can have a broader understanding of the events that have occurred. The service bot can act with a much more expected and obvious response to humans.

The second sentence," I have no more medication", received by the robot with a cheerful emotion, may support that the speaker has finished his prescribed medication which would also imply the end of his illness. In a neutral tone, the speaker may realise that the medication has run out, he still needs it, but it does not hinder his health if he goes for more later. On a sad emotion, the sentence reflects the speaker's urgent need for medication, which may even endanger his life.

The third sentence, "The test is positive", can only be deciphered from the context or the emotion conveyed. On a positive emotion, the sentence conveys the speaker's happiness and that a positive event has happened in the speaker's life, for example a pregnancy test. The sentence on a neutral tone merely implies that the speaker expected this outcome. We can't concretely say that it is a positive or negative event in the speaker's life, from the context we can better figure out what it refers to. A negative emotion imprinted on this sentence can only result that the speaker has received bad news and that the event the speaker is referring to is a negative one, e.g. a SarsCov-2 test.

The fourth sentence, "The interpretation was wrong", leads us to think of a misdiagnosis made by one doctor, then having it refuted by another. The emotion of this sentence can totally change its meaning. If we have a happy emotion, the real diagnosis is one that helps the patient. It may be that he is completely healthy or just that the disease he has is not as serious as the disease he was originally diagnosed with. The emotion of sadness that monopolizes this sentence is the exact

opposite of the sentence with happy emotion. We can tell that the diagnosis given later is a severe one that may be life-threatening for the speaker or may need immediate help.

The fifth sentence, "It has grown a lot", without context or emotion may be almost impossible for humans to decipher, resulting in robots trying to decipher this sentence being the equivalent of a random function result. In the absence of context we can rely on emotion. For an emotion of happiness, e.g. the speaker may rejoice at the growth in height of a child. For a neutral emotion, we cannot judge whether it is a happy or sad event in the speaker's life, we can only assume that the event is one that the speaker expected to happen. A negative emotion follows that in the speaker's life a negative, even expected event has occurred. For example, an increase in cholesterol at a test or an increase in a hematoma since the last check-up.

The sixth sentence, "It has dropped a lot.", is extremely similar to its predecessor, this sentence having been introduced both to give the speaker more choices of expression and to introduce difficulty in recognising the correct sentence.

The seventh sentence, "I shut the door on him.", introduces more colloquial language that the robot is likely to hear. This sentence may also activate the locomotor functions of the service robot, depending on the emotion conveyed. If the speaker conveys a positive emotion with this sentence, he/she wants to express that the door is closed and should stay that way. If the service robot detects a neutral emotion, the message it should receive is that the door is closed, but at any time it may receive a message to look for the keys or just keep the door closed. If the emotion conveyed by this message is a negative one, the robot should understand that the door is closed and should look for the keys if the door is locked and open it.

The eighth sentence is similar to its predecessor, having similar meanings but different purposes. Here the service robot, can help turn the light bulb on or off in a particular room. This sentence is very useful for people with mobility disabilities.

The ninth sentence, "I'm very well.", may expose one of the problems of the century, namely depression. If the service bot can intervene and distinguish depression at an early stage, it can help that person immeasurably. That person can take action in removing the depression until it is in an advanced form. If the emotion conveyed by this sentence is positive, it means that the paraverbal message is the same as the verbal message. If the sentence received by the robot has a neutral message, this does not mean that the speaker wants to deceive the robot with the message conveyed, the speaker may just be tired. However the robot should remain vigilant in following messages and emotions conveyed by the speaker. If the emotion conveyed is a negative one, the robot should understand that the speaker is not feeling well and should offer emotional comfort or call for help if there is a physical accident.

The last sentence, "The keys are in the door", can have several paraverbal meanings, even if the verbal message is quite clear. If the emotion that accompanies the sentence is positive, it may have as a message the joy of finding the keys after a long search, or just the joy that those keys are where they should be. If the emotion is neutral, the message may be that the robot will bring those keys to the speaker, or just a message that the location of those keys is known. If the emotion is a negative one, the robot should remove those keys from the door and bring them to the speaker.

These sentences can make robots come closer to helping humans and understand both the discrete language and the message conveyed paraverbally. With these 10 sentences we can give an example of the use of an emotional database and its importance. There is a possibility that a service bot that does not understand the paraverbal language may not be able to decode the transmitted message or even decode it wrongly and act in a wrong way.

The database has a total of 4713,391 seconds or 78.5 minutes. It consists of 10 previously explained sentences, each sentence having 3 types of emotions explained. All 7 speakers repeated each sentence on each emotion 10 times. Each sentence on average is 2.244 seconds long. The

database has a total of 2100 sentences, of which 700 are on happy emotion, 700 have a neutral emotion, and 700 have a sad emotion.

The database is structured as follows:

- Each folder represents the class, and each of them is numbered as follows: "PersonName_pk_emotion", where k represents the number of the sentence and the emotion can be happy, neutral or sad.
- Each class has 10 instances named as follows: "NamePerson_pk_emotion_n.wav", where n goes from 01 to 10 and represents the instance number.
- P1 stands for "Test result came back.", P2 stands for "No more medicine.", P3 stands for "Test is positive.", P4 stands for "Interpretation was wrong.", P5 stands for "Increased a lot.", P6 stands for "Decreased a lot.", P7 stands for "Door is closed.", P8 stands for "Light bulb is off.", P9 stands for "I am very well." And finally, P10 stands for "The keys are in the door.".
- Persons 1, 3, 4 and 7 are male and persons 2, 5 and 6 are female.

Fig. 4.1 shows the distribution of instances for 1 out of 210 classes.

Fig. 4.1 – Distribution of records

Class	No.	Person1 [s]	Person2 [s]	Person3 [s]	Person4 [s]	Person5 [s]	Person6 [s]	Person7 [s]	Total [s]
P1 - *f*	1	2.560	2.731	3.157	2.901	2.731	2.816	2.816	**19.712**
	2	2.731	2.987	3.243	3.157	2.731	3.072	2.560	**20.481**
	3	2.645	3.072	2.475	2.987	2.475	3.072	2.816	**19.542**
	4	2.475	2.987	2.731	3.157	2.560	2.901	2.901	**19.712**
	5	2.645	2.560	2.987	2.816	2.389	2.816	2.901	**19.114**
	6	2.645	3.413	2.816	2.901	2.475	2.901	2.731	**19.882**

	7	2.645	2.816	3.072	2.731	2.389	3.243	2.645	**19.541**
	8	2.645	3.840	3.072	2.731	2.389	2.560	2.645	**19.882**
	9	2.645	2.901	2.645	2.645	2.475	2.731	2.645	**18.687**
	10	2.816	3.072	2.901	2.987	2.389	2.731	2.475	**19.371**
P1 - *n*	1	2.645	3.072	2.731	3.072	2.645	2.816	2.219	**19.200**
	2	2.560	3.328	3.157	3.072	2.219	2.731	2.560	**19.627**
	3	2.389	3.328	2.560	2.731	2.475	2.816	2.048	**18.347**
	4	2.475	3.413	2.901	2.987	2.389	2.475	2.304	**18.944**
	5	2.475	3.755	2.731	2.816	2.219	2.560	2.219	**18.775**
	6	2.475	3.413	2.987	2.901	2.389	2.816	2.219	**19.200**
	7	2.645	3.328	2.645	2.987	2.133	2.645	2.219	**18.602**
	8	2.475	3.072	2.901	2.731	2.219	2.987	2.304	**18.689**
	9	2.475	3.328	2.560	2.816	2.389	2.560	2.219	**18.347**
	10	2.731	3.072	2.731	2.731	2.304	2.560	2.219	**18.348**
P1 - *t*	1	3.072	3.413	2.987	2.645	2.560	2.816	2.645	**20.138**
	2	2.987	3.413	3.072	2.304	2.560	2.560	2.645	**19.541**
	3	3.072	3.243	2.987	2.389	2.475	2.645	2.560	**19.371**
	4	3.157	3.243	2.645	2.475	2.475	2.901	2.475	**19.371**
	5	3.157	3.328	2.560	2.475	2.560	2.560	2.389	**19.029**
	6	3.072	3.243	2.816	2.304	2.475	2.560	2.304	**18.774**
	7	3.072	3.072	2.645	2.389	2.304	2.645	2.475	**18.602**
	8	3.157	3.499	2.731	2.389	2.389	2.731	2.389	**19.285**
	9	3.072	3.157	2.731	2.304	2.645	2.560	2.645	**19.114**
	10	2.816	3.243	2.560	2.219	2.304	2.901	2.731	**18.774**
P2 - *f*	1	1.877	2.560	2.645	2.304	2.475	2.731	2.304	**16.896**
	2	2.219	2.731	2.475	2.475	2.133	2.560	2.389	**16.982**
	3	2.219	2.475	2.389	2.475	2.048	2.560	2.475	**16.641**
	4	2.304	2.560	2.560	2.560	1.963	2.389	2.219	**16.555**
	5	2.560	2.304	2.304	2.304	1.963	2.219	2.048	**15.702**
	6	2.048	3.072	2.560	2.389	2.133	2.304	2.560	**17.066**
	7	1.877	2.645	2.560	2.389	1.963	2.304	2.219	**15.957**
	8	2.133	2.645	2.219	2.645	2.133	2.304	2.389	**16.468**
	9	2.133	2.731	2.475	2.219	1.963	2.560	2.475	**16.556**
	10	2.304	2.901	2.219	2.816	2.048	2.645	2.219	**17.152**
P2 - *n*	1	2.133	2.645	2.731	2.304	2.219	2.731	1.707	**16.470**
	2	2.219	3.157	2.304	2.219	2.048	2.475	1.963	**16.385**
	3	2.048	2.731	2.560	2.304	1.963	2.219	1.707	**15.532**
	4	1.877	2.987	2.475	2.389	1.963	1.877	1.877	**15.445**
	5	2.219	2.987	2.475	2.475	2.048	2.219	1.877	**16.300**
	6	2.048	2.901	2.560	2.133	1.792	2.133	1.963	**15.53**
	7	2.560	2.901	2.475	2.389	1.877	2.219	1.963	**16.384**
	8	2.389	3.669	2.560	2.389	1.877	2.304	1.877	**17.065**
	9	1.877	2.304	2.475	2.219	2.048	2.219	1.707	**14.849**
	10	2.048	2.475	2.560	2.219	2.048	2.389	2.219	**15.958**
P2 - *t*	1	2.219	3.499	2.475	1.877	2.219	1.963	2.133	**16.385**
	2	2.304	2.731	2.560	2.219	2.304	2.389	2.219	**16.726**
	3	2.389	2.816	2.219	1.963	2.389	2.645	2.475	**16.896**
	4	2.133	3.328	2.304	2.048	2.219	2.389	2.048	**16.469**
	5	2.475	3.072	2.304	1.963	2.304	2.475	2.048	**16.641**

	6	2.219	3.243	2.304	1.963	2.219	2.475	1.877	**16.300**
	7	2.304	2.901	2.219	2.133	2.133	2.475	2.389	**16.554**
	8	2.475	2.901	2.304	2.133	1.963	2.304	2.219	**16.299**
	9	2.389	2.816	2.219	2.133	2.048	2.389	2.133	**16.127**
	10	2.389	3.072	2.048	1.963	2.133	2.389	2.133	**16.127**
P3 - *f*	1	2.133	2.389	2.219	2.133	2.219	2.389	2.219	**15.701**
	2	2.475	2.901	2.389	2.304	2.304	2.304	2.133	**16.810**
	3	2.304	2.731	2.304	1.963	2.048	2.219	2.304	**15.873**
	4	2.304	2.731	2.304	2.304	2.048	2.389	2.219	**16.299**
	5	2.304	2.304	2.304	2.304	2.048	2.133	2.219	**15.616**
	6	1.963	2.816	2.901	2.048	2.133	2.389	2.304	**16.554**
	7	2.133	2.475	2.645	2.304	2.048	2.133	2.133	**15.871**
	8	2.048	3.243	2.475	2.048	2.048	2.048	2.048	**15.958**
	9	2.133	2.901	2.304	2.133	2.048	2.304	2.219	**16.042**
	10	2.304	2.901	2.389	1.963	2.048	2.133	2.133	**15.871**
P3 - *n*	1	1.792	2.560	2.389	3.072	2.133	2.475	2.048	**16.469**
	2	1.963	2.645	2.645	2.389	2.133	2.219	2.133	**16.127**
	3	1.877	2.389	2.645	2.048	1.877	2.133	2.475	**15.444**
	4	1.963	2.645	2.475	2.475	1.877	2.219	1.877	**15.531**
	5	1.963	2.901	2.645	2.048	1.877	2.219	1.707	**15.360**
	6	1.792	2.304	2.389	2.219	1.877	2.389	1.877	**14.847**
	7	1.963	2.645	2.475	2.304	1.707	2.133	1.707	**14.934**
	8	1.877	2.560	2.133	2.304	1.877	2.389	1.621	**14.761**
	9	1.963	2.560	2.219	2.133	1.877	2.304	1.707	**14.763**
	10	2.048	2.389	2.304	2.048	1.877	2.219	1.792	**14.677**
P3 - *t*	1	2.133	2.987	2.219	2.133	2.219	2.219	2.133	**16.043**
	2	2.304	3.072	2.389	2.133	2.048	2.560	2.389	**16.895**
	3	2.048	3.072	2.304	1.877	1.963	2.304	2.389	**15.957**
	4	2.304	3.072	2.219	1.877	2.048	2.389	2.389	**16.298**
	5	2.219	3.157	2.133	1.792	1.963	2.475	2.133	**15.872**
	6	2.219	2.987	2.304	1.792	2.133	2.475	1.963	**15.873**
	7	2.304	2.901	2.133	1.877	2.133	1.963	2.133	**15.444**
	8	2.133	3.157	2.389	1.963	2.048	2.219	1.877	**15.786**
	9	2.048	2.987	2.219	2.389	2.048	2.731	1.792	**16.214**
	10	2.048	2.816	2.304	2.048	1.963	2.304	1.963	**15.446**
P4 - *f*	1	2.475	2.731	2.731	2.219	2.389	2.304	2.816	**17.665**
	2	2.304	2.987	2.816	2.133	2.133	2.731	2.304	**17.408**
	3	2.475	3.243	2.816	2.133	2.133	2.304	2.304	**17.408**
	4	2.475	2.816	3.243	2.731	2.304	2.816	2.560	**18.945**
	5	2.475	3.499	3.157	2.048	2.219	2.816	2.219	**18.433**
	6	2.304	3.243	2.901	2.133	2.304	2.645	2.219	**17.749**
	7	2.304	2.816	2.731	1.963	2.048	2.475	2.048	**16.385**
	8	2.219	3.243	2.645	2.048	2.219	2.645	2.048	**17.067**
	9	2.475	2.816	2.901	2.133	2.304	2.645	2.133	**17.407**
	10	2.219	3.328	2.816	2.389	2.219	2.731	2.304	**18.006**
P4 - *n*	1	2.048	3.072	2.901	2.219	2.219	2.645	2.304	**17.408**
	2	2.389	2.731	2.816	2.645	2.304	2.731	2.048	**17.664**
	3	2.219	2.901	2.901	2.389	1.963	2.816	2.389	**17.578**
	4	2.389	2.901	2.645	2.304	2.133	2.901	2.304	**17.577**

	5	2.560	2.901	2.816	2.389	2.133	2.389	2.133	**17.321**
	6	2.219	2.731	2.560	2.304	1.963	2.475	2.133	**16.385**
	7	2.219	2.987	2.645	2.219	1.877	2.731	1.877	**16.555**
	8	2.219	2.987	2.731	2.133	1.963	2.645	2.048	**16.726**
	9	2.133	3.243	2.816	2.389	2.304	2.816	1.877	**17.578**
	10	2.219	2.987	2.560	2.304	1.963	2.901	2.133	**17.067**
P4 - *t*	1	2.560	3.328	2.645	2.304	2.304	2.731	2.475	**18.347**
	2	2.475	3.328	2.731	2.304	2.560	2.731	2.560	**18.689**
	3	2.816	3.499	2.560	2.133	2.133	2.901	2.475	**18.517**
	4	2.731	3.243	2.731	1.963	2.048	2.987	2.475	**18.178**
	5	2.475	3.328	2.731	1.877	2.219	2.816	2.389	**17.835**
	6	2.475	3.328	2.560	2.048	2.219	2.645	2.304	**17.579**
	7	2.560	3.584	2.731	2.048	2.219	2.645	2.304	**18.091**
	8	2.645	3.072	2.645	1.877	2.219	2.901	2.389	**17.748**
	9	2.731	3.413	2.304	2.048	2.133	2.901	2.475	**18.005**
	10	2.560	3.413	2.645	1.877	2.304	2.816	2.219	**17.834**
P5 - *f*	1	2.048	2.645	2.048	2.219	1.877	2.219	2.219	**15.275**
	2	2.304	2.731	2.219	1.963	2.133	1.963	2.219	**15.532**
	3	2.133	2.560	2.219	2.048	2.219	2.133	2.219	**15.531**
	4	2.219	3.157	2.219	1.963	1.963	1.963	2.133	**15.617**
	5	2.133	2.816	2.475	1.963	2.133	2.048	2.304	**15.872**
	6	2.304	2.901	2.304	1.877	2.219	2.133	2.133	**15.871**
	7	2.219	3.157	2.304	2.048	2.133	2.133	2.048	**16.042**
	8	2.133	3.499	2.475	1.792	2.219	2.133	2.048	**16.299**
	9	2.304	3.328	2.475	1.877	2.048	2.048	1.792	**15.872**
	10	2.133	3.413	2.389	2.048	2.133	2.048	1.877	**16.041**
P5 - *n*	1	1.877	2.816	2.560	2.048	1.877	2.133	1.621	**14.932**
	2	1.792	2.560	2.560	2.048	1.877	2.304	1.621	**14.762**
	3	1.963	2.645	2.304	2.048	1.877	2.048	1.451	**14.336**
	4	1.877	2.560	2.389	1.877	1.707	2.133	1.365	**13.908**
	5	1.877	2.560	2.133	1.963	1.792	2.048	1.792	**14.165**
	6	2.048	2.560	2.219	1.707	1.963	2.304	1.536	**14.337**
	7	2.048	2.560	2.219	1.792	1.877	2.219	2.304	**15.019**
	8	1.707	2.731	2.560	1.792	1.877	2.219	1.621	**14.507**
	9	1.707	2.731	2.133	1.792	1.707	2.219	1.621	**13.910**
	10	1.963	2.901	2.219	1.877	1.792	2.389	1.707	**14.848**
P5 - *t*	1	2.133	3.072	2.645	2.475	2.048	2.389	2.560	**17.322**
	2	2.389	3.328	2.816	1.963	2.048	2.133	2.304	**16.981**
	3	2.475	3.072	2.389	2.048	2.048	2.304	2.560	**16.896**
	4	2.219	2.901	2.304	2.048	1.877	2.219	2.219	**15.787**
	5	2.304	2.816	2.219	1.877	1.963	2.389	2.048	**15.616**
	6	2.475	2.901	2.560	1.792	2.048	2.389	2.048	**16.213**
	7	2.389	3.157	2.389	1.792	2.048	2.475	2.048	**16.298**
	8	2.389	3.328	2.645	1.792	2.048	2.475	2.133	**16.810**
	9	2.389	3.072	2.560	1.963	1.963	2.304	2.048	**16.299**
	10	2.560	3.243	3.157	1.792	2.048	2.560	2.133	**17.493**
P6 - *f*	1	2.133	3.499	2.304	2.219	2.219	2.475	1.877	**16.726**
	2	2.389	3.499	2.219	2.133	2.219	2.048	1.877	**16.384**
	3	2.389	3.157	2.389	1.877	2.389	2.133	1.963	**16.297**

	4	1.792	2.901	2.304	1.963	2.133	2.048	2.048	**15.189**
	5	2.133	3.243	2.133	1.792	2.048	2.048	2.048	**15.445**
	6	2.219	3.157	2.389	1.963	2.133	1.963	2.219	**16.043**
	7	2.133	3.243	2.133	2.133	2.219	1.963	1.877	**15.701**
	8	2.304	2.901	2.304	2.133	2.048	2.048	2.048	**15.786**
	9	2.133	3.328	2.219	2.048	2.219	2.133	2.133	**16.213**
	10	2.219	3.072	2.133	1.963	2.133	1.707	1.877	**15.104**
P6 - *n*	1	1.963	2.560	1.963	1.707	1.792	2.219	1.792	**13.996**
	2	1.877	2.901	2.645	2.133	1.792	2.048	2.133	**15.529**
	3	1.963	2.389	2.389	1.963	1.792	2.304	1.707	**14.507**
	4	1.792	2.645	2.475	1.707	1.877	2.475	1.877	**14.848**
	5	1.536	2.901	2.645	1.707	1.877	2.304	1.621	**14.591**
	6	1.621	2.645	2.731	1.707	1.963	2.133	1.792	**14.592**
	7	1.792	2.560	2.731	1.707	1.792	2.304	1.707	**14.593**
	8	1.792	2.645	2.304	1.707	1.877	2.389	1.792	**14.506**
	9	2.048	2.645	2.133	1.707	1.877	2.219	1.707	**14.336**
	10	1.792	2.816	2.304	2.048	1.792	2.475	1.621	**14.848**
P6 - *t*	1	2.560	3.243	2.560	1.963	2.048	2.304	2.475	**17.153**
	2	2.475	2.816	2.475	1.963	2.133	2.048	2.048	**15.958**
	3	2.219	3.243	2.645	1.963	1.877	2.389	2.048	**16.384**
	4	2.219	3.157	2.475	1.877	1.877	2.048	2.219	**15.872**
	5	2.389	3.157	2.475	1.792	1.877	2.304	2.219	**16.213**
	6	2.219	3.157	2.731	1.963	1.963	2.304	2.048	**16.385**
	7	2.389	3.157	2.560	2.048	1.877	2.389	2.048	**16.468**
	8	2.389	2.987	2.645	2.389	1.877	1.963	2.133	**16.383**
	9	2.731	3.243	2.731	1.707	1.792	2.560	2.304	**17.068**
	10	2.219	3.072	2.816	1.877	1.792	2.475	2.304	**16.555**
P7 - *f*	1	1.963	2.475	1.877	1.877	1.963	1.792	2.731	**14.678**
	2	2.048	2.645	1.792	1.792	2.048	2.133	1.536	**13.994**
	3	2.219	2.560	1.792	1.877	2.133	1.707	1.707	**13.995**
	4	2.304	2.645	2.048	1.707	1.963	1.621	1.365	**13.653**
	5	2.048	2.816	1.877	1.621	1.707	1.963	1.792	**13.824**
	6	1.963	2.901	1.792	1.707	1.963	1.877	1.792	**13.995**
	7	2.219	2.475	1.707	1.707	1.877	1.792	1.792	**13.569**
	8	2.048	3.328	1.707	2.219	1.621	1.963	2.219	**15.105**
	9	2.219	2.645	1.877	1.792	1.707	2.048	1.877	**14.165**
	10	2.219	2.816	1.877	1.707	1.792	2.304	1.792	**14.507**
P7 - *n*	1	1.621	2.304	2.048	1.621	1.792	1.792	1.451	**12.629**
	2	1.877	2.645	2.048	1.536	1.877	1.877	1.621	**13.481**
	3	1.792	2.645	2.133	1.877	1.707	1.792	1.451	**13.397**
	4	1.963	2.475	1.877	1.792	1.792	2.133	1.536	**13.568**
	5	1.707	2.816	1.877	1.877	1.707	1.963	1.792	**13.739**
	6	1.877	2.304	2.048	1.792	1.792	2.304	1.877	**13.994**
	7	1.877	2.645	2.048	1.621	1.792	2.048	1.707	**13.738**
	8	1.877	2.389	2.048	1.451	1.621	2.133	1.536	**13.055**
	9	2.048	2.389	2.048	1.707	1.707	2.219	1.365	**13.483**
	10	1.707	2.645	1.877	1.792	1.963	2.219	1.451	**13.654**
P7 - *t*	1	1.792	2.560	2.219	1.707	1.536	2.304	1.877	**13.995**
	2	1.877	2.987	2.389	1.792	1.707	2.219	1.877	**14.848**

	3	1.792	2.901	2.219	1.451	1.707	2.133	2.133	**14.336**
	4	2.133	3.243	2.048	1.707	1.707	1.963	1.963	**14.764**
	5	1.877	2.731	2.389	1.451	1.792	1.963	2.048	**14.251**
	6	2.219	2.560	2.048	1.536	1.707	2.048	1.792	**13.910**
	7	2.389	2.731	2.304	1.365	1.621	1.963	1.963	**14.336**
	8	2.133	2.987	2.475	1.707	1.877	2.048	1.707	**14.934**
	9	2.048	2.645	2.219	1.877	1.877	1.963	1.707	**14.336**
	10	1.963	2.645	2.560	1.707	1.707	2.048	1.707	**14.337**
P8 - *f*	1	2.133	2.389	2.048	1.792	1.707	1.963	2.048	**14.080**
	2	1.963	2.389	2.048	1.877	1.792	1.963	1.963	**13.995**
	3	1.707	2.645	1.963	1.621	1.536	1.877	1.707	**13.056**
	4	2.048	2.560	1.963	1.877	1.792	1.451	1.792	**13.483**
	5	1.707	2.219	2.048	2.048	1.707	1.621	1.877	**13.227**
	6	1.707	2.475	1.792	2.048	1.707	1.707	2.048	**13.484**
	7	1.877	2.389	1.792	1.621	1.792	1.707	1.877	**13.055**
	8	1.707	2.389	1.963	1.621	1.621	1.963	1.877	**13.141**
	9	1.963	2.475	2.219	1.451	1.707	1.963	1.792	**13.57**
	10	1.877	2.816	2.219	1.536	1.792	1.963	1.621	**13.824**
P8 - *n*	1	1.792	2.475	2.048	1.621	1.877	1.963	1.707	**13.483**
	2	1.707	2.389	2.219	1.536	1.707	1.707	1.451	**12.716**
	3	1.621	2.560	2.133	1.621	1.877	1.963	1.451	**13.226**
	4	1.792	2.560	2.048	1.621	1.707	2.219	1.365	**13.312**
	5	1.877	2.475	2.048	1.621	1.536	2.048	1.536	**13.141**
	6	1.877	2.475	2.219	1.707	1.536	2.048	1.792	**13.654**
	7	1.536	2.645	1.792	1.707	1.707	1.963	1.536	**12.886**
	8	1.707	2.560	2.048	1.536	1.792	2.048	1.621	**13.312**
	9	1.621	2.645	2.133	1.621	1.536	2.475	1.621	**13.652**
	10	1.792	2.645	1.792	1.707	1.707	2.133	1.451	**13.227**
P8 - *t*	1	1.963	2.816	2.389	2.133	1.877	2.133	1.621	**14.932**
	2	2.048	2.901	2.389	1.707	1.792	2.133	1.536	**14.506**
	3	2.133	2.987	2.475	1.792	1.792	2.048	1.877	**15.104**
	4	2.219	2.987	2.219	1.621	1.707	1.963	1.877	**14.593**
	5	1.963	2.987	2.560	1.792	1.877	2.133	1.792	**15.104**
	6	1.877	2.816	2.475	1.707	1.792	2.219	1.792	**14.678**
	7	1.963	2.987	2.475	1.877	1.963	2.304	1.451	**15.02**
	8	2.048	2.901	2.560	1.536	1.963	2.219	1.792	**15.019**
	9	2.048	2.987	2.645	1.707	1.963	2.048	1.707	**15.105**
	10	2.048	2.987	2.389	1.621	1.707	1.877	1.792	**14.421**
P9 - *f*	1	1.451	2.560	1.792	1.621	1.621	1.621	2.048	**12.714**
	2	2.048	2.645	1.963	1.792	1.707	1.621	1.963	**13.739**
	3	1.792	2.304	1.792	1.707	1.792	1.792	1.707	**12.886**
	4	1.451	2.133	1.792	2.048	1.792	1.877	1.621	**12.714**
	5	2.048	2.219	1.621	1.877	1.707	1.963	1.877	**13.312**
	6	2.048	2.389	1.877	1.792	1.621	2.048	1.963	**13.738**
	7	2.048	2.389	1.792	1.792	1.792	2.133	1.792	**13.738**
	8	1.536	2.389	1.877	1.792	1.707	1.536	2.048	**12.885**
	9	1.877	2.389	1.963	1.707	1.451	1.877	1.963	**13.227**
	10	1.707	2.560	1.877	1.536	1.707	1.707	2.048	**13.142**
P9 - *n*	1	1.707	2.645	1.877	1.536	1.707	2.048	1.621	**13.141**

	2	1.707	2.731	1.963	1.621	1.707	2.048	1.451	**13.228**
	3	1.621	2.475	1.877	1.536	1.877	1.536	1.536	**12.458**
	4	1.621	2.560	1.792	2.048	1.707	1.621	1.536	**12.885**
	5	1.536	2.304	1.792	1.621	1.707	1.963	1.707	**12.630**
	6	1.877	2.389	1.877	2.133	1.707	1.707	2.133	**13.823**
	7	1.792	2.475	1.792	1.621	1.877	1.963	1.792	**13.312**
	8	1.451	2.560	1.792	1.536	2.133	1.877	1.707	**13.056**
	9	1.792	2.731	1.877	1.365	1.451	1.963	1.621	**12.800**
	10	1.707	2.731	1.792	1.792	1.536	1.877	1.707	**13.142**
P9 - *t*	1	1.963	2.901	2.816	1.963	1.792	1.536	1.792	**14.763**
	2	2.389	3.328	2.901	1.792	1.792	1.707	1.707	**15.616**
	3	2.133	2.645	2.816	2.133	1.707	1.621	1.621	**14.676**
	4	1.963	2.560	2.645	1.707	1.877	1.707	2.048	**14.507**
	5	1.963	2.560	2.475	1.536	1.707	2.048	1.792	**14.081**
	6	2.048	2.901	2.901	1.621	1.621	1.963	1.792	**14.847**
	7	1.877	2.731	2.645	1.451	1.621	1.877	2.048	**14.25**
	8	1.877	2.987	2.475	1.536	1.621	1.536	1.792	**13.824**
	9	1.792	2.987	2.560	1.621	1.621	2.304	1.963	**14.848**
	10	2.133	3.072	2.901	1.707	1.707	2.048	1.792	**15.360**
P10 - *f*	1	2.048	2.816	2.048	2.304	2.133	2.048	1.963	**15.360**
	2	2.133	2.560	2.133	1.963	2.133	2.133	2.048	**15.103**
	3	2.304	2.389	1.963	2.389	2.048	2.133	1.877	**15.103**
	4	2.219	2.731	1.963	2.475	1.963	2.048	1.877	**15.276**
	5	2.048	2.645	1.792	2.475	1.877	1.792	2.048	**14.677**
	6	2.475	2.816	1.877	2.219	1.792	2.048	1.877	**15.104**
	7	2.304	2.731	1.877	2.560	1.963	2.389	1.877	**15.701**
	8	2.475	2.560	1.877	2.645	2.048	1.707	1.963	**15.275**
	9	2.389	2.901	2.048	2.731	1.963	2.133	1.963	**16.128**
	10	2.219	2.987	1.877	2.304	1.963	2.133	1.963	**15.446**
P10 - *n*	1	1.963	2.475	1.963	2.133	1.963	2.475	2.133	**15.105**
	2	2.048	2.816	2.048	2.048	1.963	2.133	2.048	**15.104**
	3	1.877	2.816	1.963	1.963	1.963	2.048	2.048	**14.678**
	4	2.048	2.645	1.963	1.707	1.877	1.963	2.219	**14.422**
	5	1.963	2.901	2.048	2.389	1.877	2.048	2.133	**15.359**
	6	2.133	2.816	1.877	1.621	1.792	2.219	1.963	**14.421**
	7	2.133	2.816	1.963	1.707	1.707	2.048	2.048	**14.422**
	8	1.963	3.157	1.963	1.621	1.877	2.133	1.877	**14.591**
	9	1.877	2.731	1.877	1.536	1.877	2.048	1.877	**13.823**
	10	2.133	3.157	1.963	1.963	1.877	2.048	1.792	**14.933**
P10 - *t*	1	2.219	3.755	2.389	1.792	2.048	2.219	1.963	**16.385**
	2	2.304	3.328	2.389	2.219	1.707	2.304	1.877	**16.128**
	3	2.304	3.243	2.560	1.963	1.877	1.792	2.133	**15.872**
	4	2.133	3.413	2.475	2.133	1.877	2.304	1.877	**16.212**
	5	2.304	3.157	2.560	2.048	1.877	2.219	2.133	**16.298**
	6	2.304	3.157	2.475	1.963	1.963	1.963	1.963	**15.788**
	7	2.048	3.072	2.475	2.304	1.792	2.133	1.963	**15.787**
	8	2.133	2.901	2.560	2.048	1.963	2.048	2.133	**15.786**
	9	2.389	3.584	2.475	1.963	1.792	2.219	1.877	**16.299**
	10	2.304	2.987	2.645	2.133	1.877	2.304	1.963	**16.213**

Total duration [s]	**4713.391**

Table 4.1 – Duration of each instance

As we can see in the table, the sentences have been chosen in such a way that they are all of similar length. We can see that on average the person with the number 2 had the longest sentences in terms of time and the person with the number 7 had the shortest sentences in terms of time. This can also be seen in Fig. 4.2 below.

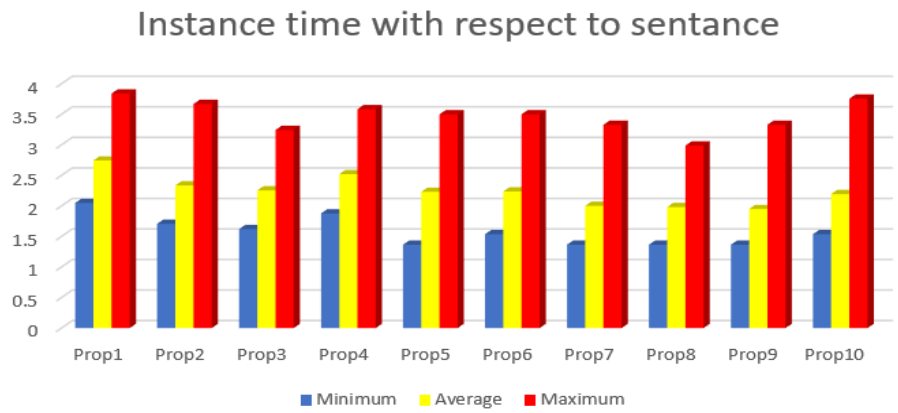

Fig. 4.2 – Minimum, average and maximum values of time instances as a function of sentence

The minimum is represented by blue, the average is represented by yellow and the maximum is represented by red. It can be seen that sentence number 1(A came the result of the test) has the highest minimum value of 2.048 seconds, and is also the only one with a minimum value of more than 2 seconds. The smallest minimum value is given by sentences 5(It got very high), 7(I closed the door), 8(I turned off the light) and 9(I am very well) with a value of 1.365 seconds. The highest average value also belongs to sentence number 1, with a value of 2.743 seconds. The only 2 sentences with an average length of only 2 seconds are sentence 8 and sentence 9. It is likely that sentences 8 and 9 are quite common for speakers and therefore have an average length of less than 2 seconds. Sentence number 1 also has the highest maximum duration, which is 3.84 seconds. The smallest and only maximum that is below 3 seconds is sentence number 8. This may also be due to its colloquial expression.

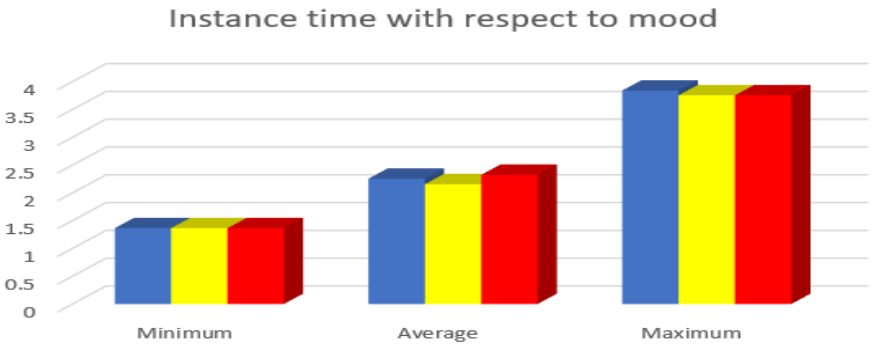

Fig. 4.3 – Minimum, mean and maximum values of time instances as a function of the emotion conveyed.

In Fig. 4.3 above we can see the sentence lengths by sentence. Happy emotion is represented with blue, neutral emotion is represented with yellow, and sad emotion is represented with red. Even though we would expect sad sentences to be much longer than happy or neutral sentences, we can see that they are about the same length. If we consider the minimum, all 3 emotions are equal with a sentence length of 1.365 seconds. When we consider the average length, the sad emotion is

the longest with a length of 2.324 seconds followed by the happy emotion with 2.254 seconds and finally the neutral emotion with 2.155 seconds. The longest maximum value belongs to the happy emotion with a length of 3.84 seconds, followed by the two remaining emotions neutral and sad with 3.755 seconds. After this analysis, we can say that we can't categorize a sentence by emotion according to its length

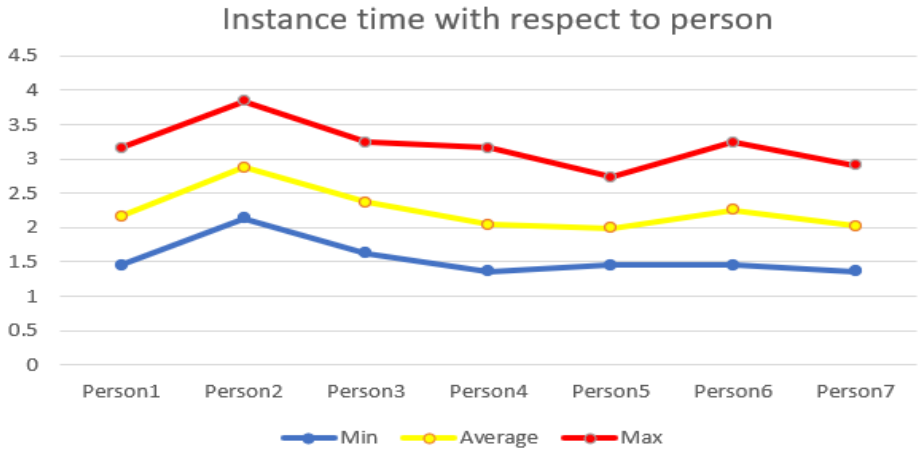

Fig. 4.4 – Minimum, average and maximum values of time instances by person/voice

From Fig. 4.4 above we can analyse by person the minimum, average and maximum. The minimum is represented with blue, the average is represented with yellow and the maximum is represented with red. We can see that both the minimum, average and maximum values have the highest values for person number 2 with the values corresponding to the minimum of the average and the maximum: 2.133 seconds, 2.876 seconds and 3.84 seconds. The lowest value for the minimum sentence duration is person 4 and person 7 with 1.365 seconds. The lowest value, and the only one that falls below 2 seconds for the average value, belongs to person number 5. The minimum of the maximum duration also belongs to person number 5 with a value of 2.731 seconds.

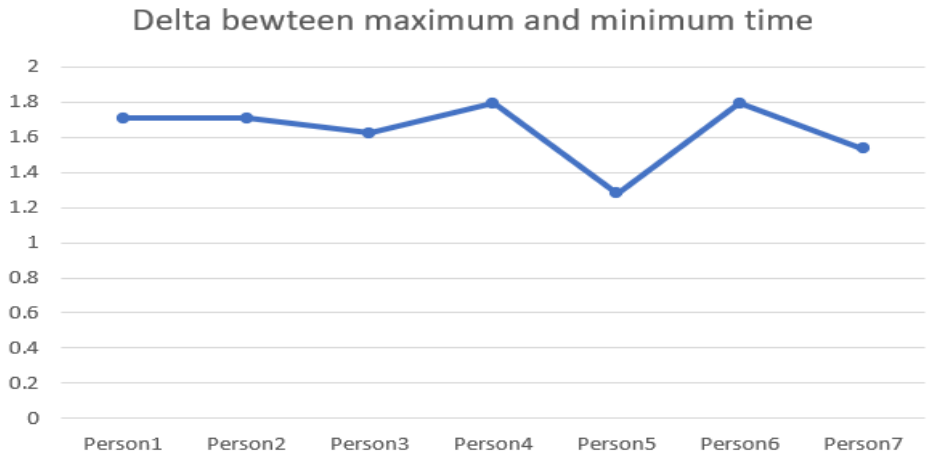

Fig. 4.5 – Difference between maximum and minimum value of time instances by person/voter

In Fig. 4.5 above explains the difference between the maximum and minimum sentence lengths for all individuals. There is quite a big difference because sentence number 1 is longer than sentence number 9 which on average is the shortest. We can see that for persons 1, 2, 3, 4 and 6 the time difference is in the range of 1.6 seconds and 1.8 seconds. Person number 7 has a delta of 1.536 seconds, but the person with the smallest delta is person number 5, who has a deviation of only 1.28 seconds.

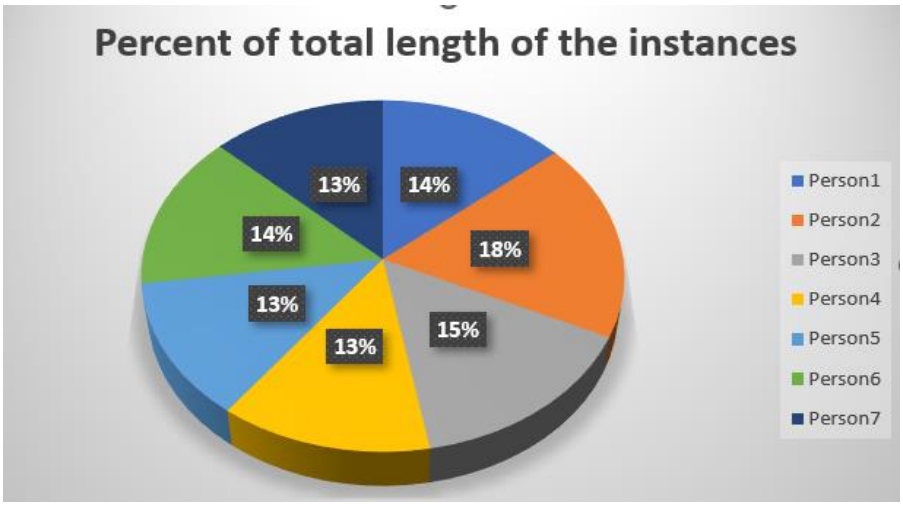

Fig. Fig. 4.6 – Total sentence time by person in percent

In Fig. 4.6 shown above, we have the total seconds per person represented in percentages. We can see that the percentages are roughly equal for all people except person number 2, who has 3 percent more than the next person.

Implementation

Python is a multi-paradigm dynamic programming language created in 1989 by Dutch programmer Guido van Rossum. Van Rossum is still today a leader in the software development community working to perfect Python and its core implementation, CPython, written in C. Python is a multi-purpose language used for example by companies like Google or Yahoo! for programming web applications, but there are also a number of scientific or entertainment applications programmed partly or entirely in Python. The growing popularity and power of the Python programming language has led to its adoption as the primary development language by specialist programmers and even to the teaching of the language in some university environments. For the same reasons, many Unix-based systems, including Linux, BSD and Mac OS X include the CPython interpreter out of the box.

Python emphasizes cleanliness and simplicity of code, and its syntax allows developers to express some programmatic ideas in a clearer and more concise manner than in other programming languages such as C. In terms of programming paradigm, Python can serve as a language for object-oriented software, but also allows for imperative, functional or procedural programming. The typing system is dynamic and memory management is automatic via a garbage collector service. Another advantage of the language is the existence of a large standard library of methods. The reference implementation of Python is written in C and is therefore called CPython. This implementation is free software and is managed by the Python Software Foundation [1].

In order to extract features we need to determine which audio files we want to extract. For example, for the CREMA database the following code was used:

```
directory = "D:/Doctorat_Toma/CREMA/"
```

```
    directory2= "D:/Doctorat_Toma/CREMA/MFCC/"
    # put your own directory here
    # directory to put our results in, you can change the name if
you like
    for it in os.scandir(directory):
        if it.is_dir():
            directoryName = it.path
            #resultsDirectory = directoryName + "/MPEG"

            # make a new folder in this directory to save our
results in
            #if not os.path.exists(resultsDirectory):
                #os.makedirs(resultsDirectory)

            # MFCCs for every .wav file in our specified directory
.csv SAVE
            for filename in os.listdir(directoryName):
                if filename.endswith('.wav'):  # only get MFCCs
from .wavs
                    # read in our file
                    (rate, sig) = wav.read(directoryName + "/" +
filename)
                        directoryName_final = directoryName + "/" +
filename
```

where directory defines where the database is defined and directory2 defines the file where the features will be extracted. To identify all the ".waw" files, we see that we use the .endswith function to be able to search by the required file type.

For MFCC we use the Python package called "python_speech_features" from which we choose the MFCC function. This function has the following parameters:

sig - which is the signal used for feature extraction

Fs - desired frequency in Hz

Wl - or window length, is the length of the analysis window measured in seconds

Ws - or window step, is the step between successive windows measured in seconds.

L - is the number of MFCC coefficients

M - is the number of filterbanks,

N_fft = is the size of the fast Fourier transform.

LF - is the minimum frequency

HF - is the maximum frequency

alpha - is the pre-emphasis coefficient applied to the pre-emphasis filter

Lf - number of cepstral parameters

So the MFCC function will look like in the following example:

```
python_speech_features.base.mfcc(sig, Fs, winlen=Wl, winstep=Ws,
numcep=L, nfilt=M, nfft=N_fft, lowfreq=LF, highfreq=HF,
preemph=alpha, ceplifter=Lf, appendEnergy=True,
winfunc=numpy.hamming)
```

To extract the coefficients of the CREMA database, we need to see how it is composed and we need to group the emotions into different classes. For example, in each sentence name in the CREMA database from position 39 to position 42 we have a grouping of 3 letters which can be 'ANG', 'DIS', 'FEA', 'HAP', 'NEU' or 'SAD' which correspond to the following emotions: nervousness, disgust, fear, happiness, neutral and sadness. The following code was used to group the emotions into the appropriate classes:

```
if directoryName_final[39:42] == 'ANG':
                        classNo = 1
                    if directoryName_final[39:42] == 'DIS':
                        classNo = 2
                    if directoryName_final[39:42] == 'FEA':
                        classNo = 3
                    if directoryName_final[39:42] == 'HAP':
                        classNo = 4
                    if directoryName_final[39:42] == 'NEU':
                        classNo = 5
                    if directoryName_final[39:42] == 'SAD':
                        classNo = 6
```

For other databases, similar approaches have been used, for example for the SAVEE database, from position 33 to position 35 we have a grouping of 1 or 2 letters which can be 'a', 'd', 'f', 'h', 'n', 'sa' or 'su' corresponding to the following emotions: nervousness, disgust, fear, happiness, neutral, sadness and surprise.

```
if directoryName_final[33:34] == 'a':
                         classNo = 1
                    if directoryName_final[33:34] == 'd':
                         classNo = 2
                    if directoryName_final[33:34] == 'f':
                         classNo = 3
                    if directoryName_final[33:34] == 'h':
                         classNo = 4
                    if directoryName_final[33:34] == 'n':
                         classNo = 5
                    if directoryName_final[33:35] == 'sa':
                         classNo = 6
                    if directoryName_final[33:35] == 'su':
                         classNo = 7
```

For the Emo-DB database, from position 33 to position 34 we have 1 letter which can be 'F', 'W', 'L', 'E', 'A', 'T' or 'N' which are from German and correspond to the following emotions: happiness, anger, boredom, disgust, anxiety/fear, sadness and neutral.

```
if directoryName_final[33:34] == 'F':
                        classNo = 1
                    if directoryName_final[33:34] == 'W':
                        classNo = 2
                    if directoryName_final[33:34] == 'L':
                        classNo = 3
                    if directoryName_final[33:34] == 'E':
                        classNo = 4
                    if directoryName_final[33:34] == 'A':
                        classNo = 5
                    if directoryName_final[33:34] == 'T':
                        classNo = 6
                    if directoryName_final[33:34] == 'N':
                        classNo = 7
```

For the Ravdess database, the class number is in the sound title and is found from position 57 to 59. This is translated into code as follows:

```
classNo = int(directoryName_final[57:59])
```

As we know, the Romanian database has only 3 emotions: happiness, neutral and sadness. To create the classes for these 3 emotions, we use all the file names of an emotion is found from position 39 to 40 where 'f' represents the emotion of happiness, 'n' represents the emotion of neutral, 't' represents the emotion of sadness. The code for these will be:

```
if directoryName_final[39:40] == 'f':
                        classNo = 1
                    if directoryName_final[39:40] == 'n':
                        classNo = 2
                    if directoryName_final[39:40] == 't':
                        classNo = 3
```

Now that we've seen how all the databases are organized, let's shift our attention to the remaining feature extractors. To use LPC we imported the "spafe" packet. The coefficients needed for LPC are similar to those used by MFCC. The coefficients are: sig, Fs, Wl, Ws, L where now is the LPC model order and alpha. The code in python will be:

```
lpc_coef = lpc(sig, Fs, order=L, win_len=Wl, win_hop=Ws,
pre_emph_coeff=alpha, win_type=numpy.hamming)
```

To use GFCC we imported the "spafe" package. The coefficients needed for GFCC are similar to those used by MFCC. The coefficients are: sig, Fs, Wl, Ws, L where now is the order of the GFCC model, M, N_fft, LF, HF, Lf, alpha and D which is the type of discrete cosine transform is used. The code in python will be:

```
gfcc_coef = gfcc(sig, Fs, win_len=Wl, win_hop=Ws, num_ceps=L,
nfilts=M, nfft=N_fft, low_freq=LF, high_freq=HF,
pre_emph_coeff=alpha, lifter=Lf, use_energy=True,
win_type=numpy.hamming, dct_type=D)
```

To use MSRCC we imported the "spafe" package. The coefficients needed for MSRCC are similar to those used by GFCC. The coefficients are: sig, Fs, Wl, Ws, L where now is the order of the MSRCC model M, N_fft, LF, HF, Lf, alpha and D which is the type of discrete cosine transform is used. The code in python will be:

```
msrcc_coef = msrcc(sig, Fs, win_len=Wl, win_hop=Ws, num_ceps=L,
nfilts=M, nfft=N_fft, low_freq=LF, high_freq=HF,
pre_emph_coeff=alpha, lifter=Lf, use_energy=True,
win_type=numpy.hamming, dct_type=D)
```

To use the NGCC we imported the "spafe" package. The coefficients required for NGCC are similar to those used by GFCC and MSRCC. The coefficients are: sig, Fs, Wl, Ws, L where now is the NGCC model order, M, N_fft, LF, HF, Lf, alpha and D which is the type of discrete cosine transform is used. The code in python will be:

```
ngcc_coef = ngcc(sig, Fs, win_len=Wl, win_hop=Ws, num_ceps=L,
nfilts=M, nfft=N_fft, low_freq=LF, high_freq=HF,
pre_emph_coeff=alpha, lifter=Lf, use_energy=True,
win_type=numpy.hamming, dct_type=D)
```

If we look at the mel feature, we used the "librosa" package where we used the melspectogram function which has the following coefficients: sig, Fs, N_fft, hop_length which is the number of samples between 2 frames, WL, Wt being the type of window used, pad_mode being the type of sound attenuation on the edges of the frame, power being the exponent of the magnitude of the mel spectogram and n_mels being the number of samples that are extracted to create the mel spectogram feature. The code in python will be:

```
mel_coef = melspectrogram(y=sig, sr=Fs, S=None, n_fft=2048,
hop_length=512, win_length=None, window='hann', center=True,
pad_mode='constant', power=2.0, n_mels=128)
```

To arrive at the chroma feature, we used the "librosa" package where we used the chroma_stft function which has the following coefficients: sig, Fs, N_fft, hop_length, WL, Wt, pad_mode, and n_chroma being the number of points that are extracted to create the chroma spectrogram feature. The code in python will be:

```
chroma_coef = chroma_stft(y=sig, sr=Fs, S=None, norm=numpy.inf,
n_fft=2048, hop_length=512, win_length=None, window='hann',
center=True, pad_mode='constant', tuning=None, n_chroma=12)
```

To calculate the fundamental frequency characteristic F0, we used the "pandas" package where we used the yin function which has the following coefficients: wav_data being the file name, sr being the sampling frequency, fmin and fmax, being the minimum and maximum frequency. The code in python will be:

```
f0 = librosa.yin(wav_data, sr=48000, fmin=50, fmax=2100)
```

For F0, for each database a differentiation was made for the gender of the speaker to better see the differences between male and female speakers and the differences between the two categories. For example for the Emo-DB database the code will be:

```
if filename[0:2] == '03':
                        gender = 2
                    if filename[0:2] == '08':
```

```
    gender = 1
if filename[0:2] == '09':
    gender = 1
if filename[0:2] == '10':
    gender = 2
if filename[0:2] == '11':
    gender = 2
if filename[0:2] == '12':
    gender = 2
if filename[0:2] == '13':
    gender = 1
```

So far we've seen all the feature extractors transposed into Python code. To have a result you also need a classification. For that we need to see how the *k*-NN classifier is implemented. It took the package "sklearn" from where we used the function KNeighborsClassifier with the following coefficients: n_neighbors representing the number of neighbors, weights, representing the weight of each neighbor. Leaf_size represents the size of the leaves. This can affect the speed of construction and query as well as the memory required to maintain the tree. The optimal value depends on the nature of the problem.P represents the power parameter for the Minkowski metric. When p = 1, this is equivalent to using the manhattan distance (l1), and the Euclidean distance (l2) for p = 2. For arbitrary p, the Minkowski distance is used. The metric refers to the type of distance calculated. Minkowski distance is a metric in a normed vector space, which can be considered a generalisation of both Euclidean distance and Manhattan distance. It is named after the German mathematician Hermann Minkowski. The code for *k*-NN would look like this:

```
KNeighborsClassifier(n_neighbors=5, weights='uniform',
algorithm='auto', leaf_size=30, p=2, metric='minkowski',
metric_params=None, n_jobs=None)
```

To integrate the cross validation as well, we used the cross_val_score function which we set to do 10 times the cross validation, divided into 10 samples. The code for this will be:

```
result_rkf = cross_val_score(estimator=pipe, X=X, y=y,
scoring='accuracy', cv=RepeatedKFold(n_splits=10, n_repeats=10))
```

SVC(C=100, kernel='rbf', degree=3, gamma=0.1, coef0=0.0, shrinking=True, probability=False,
tol=0.1, cache_size=200, class_weight=None, verbose=False, max_iter=- 1, decision_function_shape='ovr', break_ties=False, random_state=None)

The implementation is based on libsvm. The matching time scales at least quadratically with the number of samples and may be impractical beyond tens of thousands of samples. Multiclass support is handled according to a one-to-one scheme.

Vector support machines are effective in large spaces. It uses a subset of training points in the decision function (called support vectors), so it is also efficient in memory. Different kernel functions can be specified for the decision function. Common kernels are provided, but it is also possible to specify custom kernels.

When training an SVM with the RBF kernel, two parameters must be considered: C and gamma. The C parameter, common to all SVM kernels, changes the misclassification of training examples with the simplicity of the decision surface. A low C makes the decision surface smooth, while a high C targets the correct classification of all training examples. gamma defines how much

influence a single training example has. The higher the gamma, the closer the other examples need to be to be affected.

The second parameter of the function specifies the type of kernel that will be used in the algorithm 'linear', 'poly', 'rbf', 'sigmoid' or 'precomputed'. The degree of the polynomial kernel function is taken into account only by the 'poly' type. Coef0 is an independent term in the kernel function, being significant only in 'poly' and 'sigmoid' type. Probability is set to 'False' because it is not desired to activate probability estimates. The stopping tolerance criterion is considered to be 0.1. The size of the kernel cache has a strong impact on the run times for larger problems. 'class_weight' sets the C parameter of class i to class_weight[i]*C for the SVC. If not given a value, all classes should have weight one. The 'balanced' mode uses the values of y to automatically adjust the weights inversely proportional to the class frequencies in the input data as n_samples / (n_classes * np.bincount(y)). If verbose output is enabled, it may not work correctly in a multi-class context. The decision function depends on a particular subset of the training data, called support vectors [2].

LogisticRegression(C=50, multi_class = 'multinomial', penalty= 'l1', solver = 'saga', tol=0.1)

Some penalties may not work with some resolvers. The chosen penalty norm is L1. Lasso regression (or L1 regularization) is a regularization technique that penalizes large-valued, correlated coefficients. It introduces a regularization term (also called a penalty term) into the sum-squared error loss function of the model. This penalty term is the absolute value of the sum of the coefficients.

The algorithm used in the optimization problem is "lbfgs". In order to choose the optimal solution, multiclass prediction and the possibility of intercepting multinomial losses are taken into account, thus choosing 'saga'.

These are some of the essential parameters we may encounter when using logistic regression in scikit-learn. The actual parameters may vary slightly depending on the specific version of scikit-learn we are using. To get the most accurate and up-to-date information, it is always a good idea to consult the official documentation for the version we are working with [3].

```
O RandomForestClassifier(n_estimators=100,criterion='gini',
                                        max_depth=None,
min_samples_split=2,
                                        min_samples_leaf=1,
min_weight_fraction_leaf=0.00,
                                        max_features='auto',
max_leaf_nodes=None, min_impurity_decrease=0.0,
                                        bootstrap=True,
oob_score=False,
                                        n_jobs=None,
random_state=None, verbose=0, warm_start=False,
                                        class_weight=None,
ccp_alpha=0.0, max_samples=None)
```

Random forest is a metaestimator that fits a number of decision tree classifiers on different subsamples of the dataset and uses averaging to improve predictive accuracy and control overfitting. The forest trees use the best splitting strategy, i.e. equivalent to passing splitter="best" to the basic DecisionTreeRegressor. The subsample size is controlled with the max_samples parameter if bootstrap=True (default), otherwise the entire dataset is used to construct each tree.

The number of trees in the forest is taken to be 100. The function for measuring the quality of a data split, the 'criterion' has the criterion 'gini', for the Gini impurity for the Shannon information gain. This parameter is tree specific.

The minimum number of samples required to split an internal node is chosen to be 2, and the minimum number of samples required to be at a leaf node is 1. A split point at any depth will only be considered if it leaves at least min_samples_leaf training samples in each of the left and right branches. This can have the effect of smoothing the model, especially in regression.

The 6th parameter represents the minimum weighted fraction of the total sum of weights (from all input samples) required to be at a leaf node. Samples have equal weight when the sample_weight parameter is not provided. The number of features to consider when looking for the best split is considered auto. The best nodes are defined as a relative reduction of impurities. It is considered 'None', then an unlimited number of leaf nodes are available. A node will be split if this splitting induces an impurity decrease greater than or equal to the value of the next parameter. Bootstrap samples are used when constructing trees. The number of jobs to run in parallel is chosen 1. If this parameter 'random_state' is set, it ensures reproducibility by setting random seeds for random number generation.

These parameters can be adjusted to tune the performance of the Random Forest model based on the specific characteristics of the dataset and the problem at hand.

Results

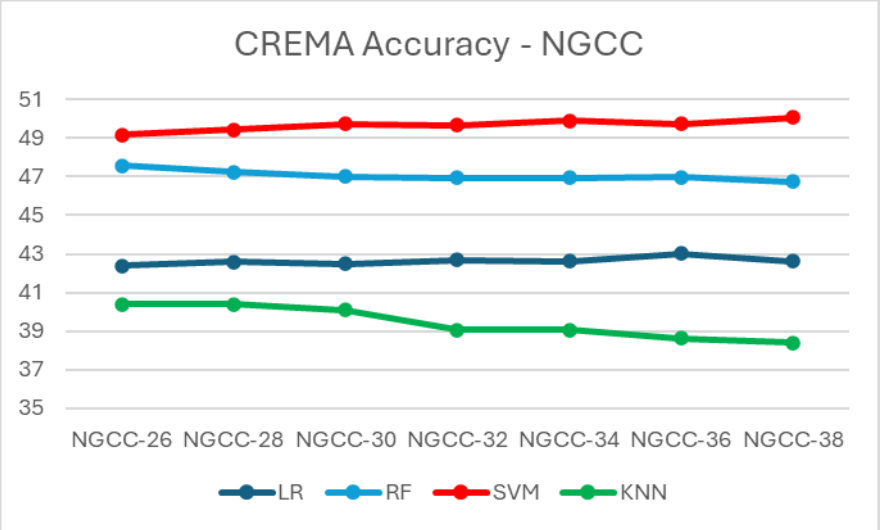

Fig. 6.4 – Correct classification rate for the CREMA database using the NGCC features extractor.

From fig. 6.4 we can extract that *k*-NN has the lowest correct classification rate with a maximum of 40.4% for 26 and 28 features. The minimum for *k*-NN is 38.39% for 38 features. *K*-NN is the minimum for each number of features. LR has an approximately upward slope starting from a minimum of 42.38% for 26 features and then increasing to a maximum of 43.02% for 36 features. LR has the second worst result for each number of characteristics. RF shows a jump of about 4 percent over LR and ranks 2nd in this statistic for the best correct classification rate. The results for LR represent a downward slope but with values not too far apart from one feature to another, with a maximum for 26 features with a value of 47.59% and a minimum value of 46.73% for 38 features. The SVM has the best accuracy for each number of features, showing an upward slope, and is also the only one to achieve a correct classification rate of over 50%. The maximum for SVM is 50.08% for 38 features and a minimum of 49.18 for 26 features.

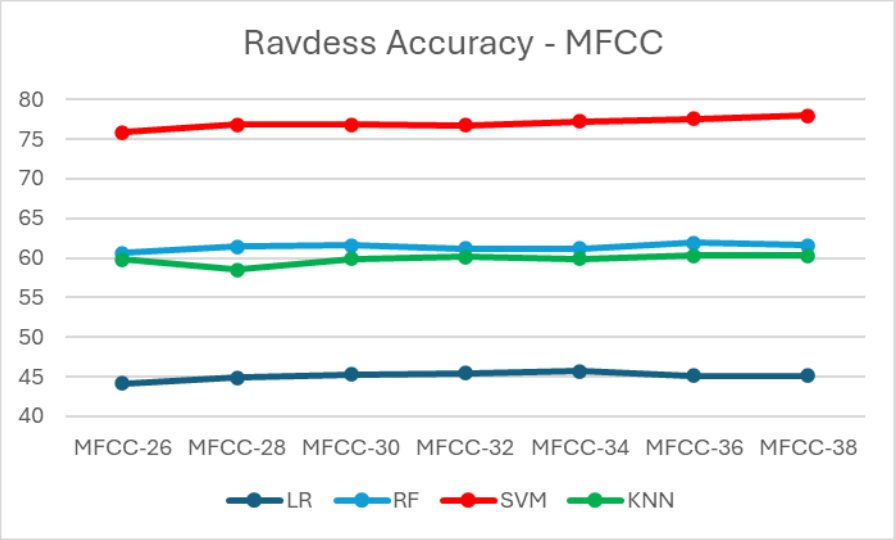

Fig. 6.13 – Correct classification rate for the Ravdess database using the MFCC features extractor.

Moving on to another database, namely Ravdess, we see a fairly strong change in the correct classification rate. As we can see the worst classifier in terms of accuracy becomes LR, which remains roughly constant around 45%, with a minimum value of 44.11% for a number of 26 features and a maximum value of 45.47% for a number of 32 features. *K*-NN managed to outperform LR but also to come close to RF, with a minimum value of 58.53% for 28 features and a maximum value of 60.28% for 36 features. The RF classifier ranks 2nd in terms of accuracy, with the results remaining approximately equal for all feature numbers. The RF had a minimum of 60.6% for a number of 26 features, but also a maximum of 61.91% for 36 features. The best classifier in terms of correct classification rate and the only classifier that managed to pass an accuracy of 70% is the SVM. The results of the SVM classifier describe an upward slope starting from 26 features with an accuracy of 75.9% and reaching 39 features with an accuracy of 77.94%.

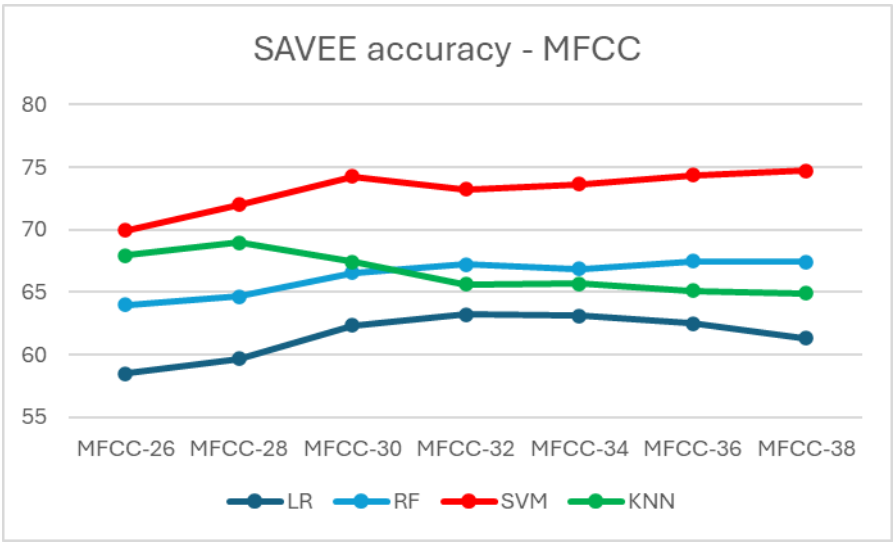

Fig. 6.23 – Correct classification rate for the SAVEE database using the MFCC features extractor.

Fig. 6.23 shows the accuracy of the SAVEE database using the MFCC feature extractor for the LR, RF, SVM and *k*-NN classifiers. For the LR classifier, an improvement over the CREMA and Ravdess databases can be seen, where it had an accurate classification rate below 50%. LR has a minimum accuracy value of 58.5% for 26 features and a maximum value of 63.2% for a total of

32 features. RF describes an upward slope, starting from a minimum value of 64% for 26 features and reaching a maximum value of 67.48% for 36 features. On the other side is the *k*-NN classifier which describes a downward slope, starting from a maximum value of 68.99% for 28 features, going down to a minimum value of 64.89%. The best classifier in terms of accuracy remains the SVM, with maximum values for each number of features. The SVM has a minimum value of 69.96% correct classification rate for 26 features and a maximum value of 74.71% for 38 features.

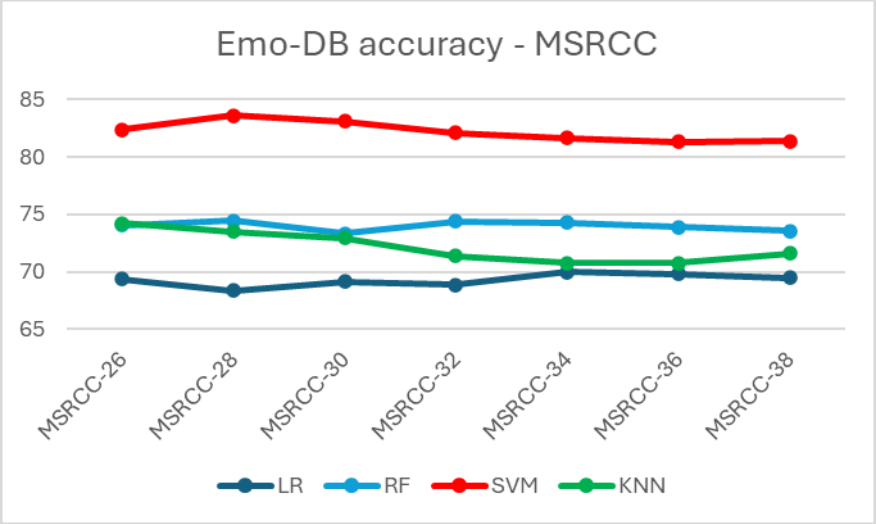

Fig. 6.39 – Correct classification rate for the Emo-DB database using the MSRCC feature extractor.

In fig. 6.39 we are shown the accuracy for the Emo-DB database using the MSRCC feature extractor and all classifiers used so far. We can see an increase for the *k*-NN classifier and maintenance for the other three classifiers compared to previous experiments for the same database. The classifier with the worst results in terms of accuracy is LR, with a minimum value of 68.37% for 28 features and a maximum of 69.96% for 34 features. The next classifier in terms of the rate of accurate classification is *k*-NN, which although it has had a fairly large increase over previous results only ranks 3rd. This classifier has a minimum accuracy of 70.76% for 36 features and a maximum value of 74.2% for 26 features. In second place in terms of accuracy is the RF classifier with a minimum value of 73.34% for 30 features and a maximum value of 74.42% for 32 features. The best classifier in terms of accuracy is the SVM, the only classifier with a correct classification rate of over 80%. The lowest accuracy value for SVM is 81.29% for 36 features, but the highest accuracy value is given for 28 features with a value of 83.6%.

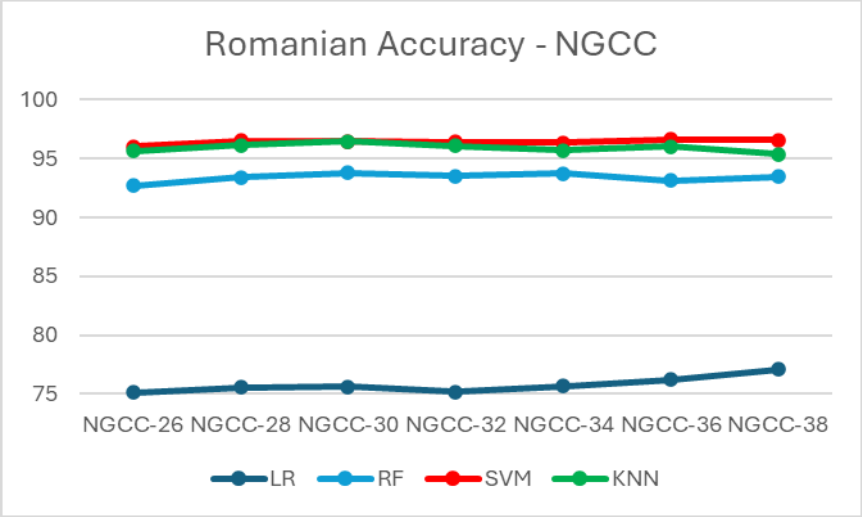

Fig. 6.46 – Correct classification rate for the Romanian database using the NGCC features extractor.

Fig. 6.46 shows the correct classification rate for the Romanian database, using the NGCC features extractor for all classifiers so far. Compared to the previous experiment, we can see that all the correct classification rates have been increased, the order of the classifiers changes only between the first 2 classifiers compared to the previous test. The worst correct classification rate belongs to the LR classifier for all classifier numbers. It can also be seen that the results for LR show an upward slope. The minimum value for the correct classification rate of the LR classifier is 75.13% for 26 features, and the maximum value for LR is 77.11% for 38 features. The next most accurate classifier is also the RF classifier with a minimum classification rate of 92.7% for 26 classifiers and a maximum value of 93.77% for 30 classifiers. The next classifier is *k*-NN, which has a minimum classification rate of 93.34% for 38 classifiers and a maximum classification rate of 96.46%. The best classifier in terms of accuracy is SVM, with a minimum classification rate of 96.04% for 26 features, and a maximum value of 96.63% for 36 features.

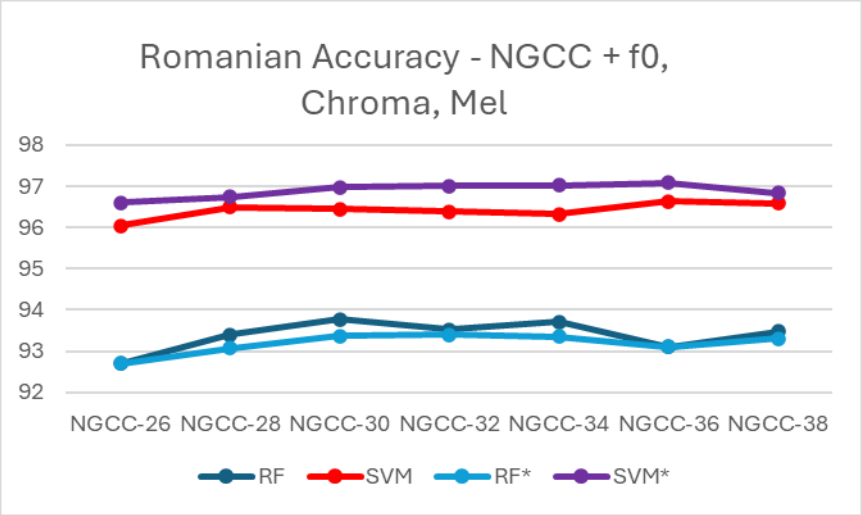

Fig. 6.52 – Correct classification rate for the Romanian database using the NGCC feature extractor as well as F0, Chroma and Mel features.

Fig. 6.52 shows the case where we change the MFCC feature extractor to NGCC. With this change we can observe both a decrease in the difference between SVM* and SVM and a change in

the combination of RF and RF* where RF has the best correct classification rate in 6 out of 7 cases, in the other case both have the same accuracy value. If we look at the case where SVM* has higher values than SVM, we can see that SVM* has a minimum value of 96.61% for 26 features, but also a maximum value that exceeds the 97% threshold and reaches a value of 97.08% for 36 features.

For the Romanian database, the most optimal combination of extractor and classifier in terms of correct classification rate is between NGCC and SVM for 36 features, if we also consider the features F0, Chroma and Mel. If we also consider time, the most optimal combination would be NGCC with *k*-NN for 30 features.

From all the results so far, we decided to create models for the NGCC feature extractors with 30 features, NGCC with 36 features, MFCC with 36 features and the *k*-NN and SVM classifiers. In total 6 models will be created. For these models 80% of the data was used for training and 20% for testing. For testing, sounds 3 and 8 were chosen from each class and the rest for training.

```
For NGCC-30 + kNN
Train accuracy is: 82.440 %
Test accuracy is: 73.810 %

For NGCC-36 + kNN
Train accuracy is: 82.500 %
Test accuracy is: 74.762 %

For MFCC-36 + kNN
Train accuracy is: 91.190 %
Test accuracy is: 90.952 %

For NGCC-30 + SVM
Train accuracy is: 98.690 %
Test accuracy is: 91.667 %

For NGCC-36 + SVM
Train accuracy is: 98.929 %
Test accuracy is: 91.667 %

For MFCC-36 + SVM
Train accuracy is: 100.000 %
Test accuracy is: 95.000 %
```

Fig. 6.53 – Results for each model

Fig. 6.53 shows the current classification rates for each model. We can see that for training, the correct classification rate is sun 100% for all cases except when we use the MFCC with 36 features and the SVM classifier. The lowest correct classification rate for training belongs to the NGCC model with 30 features and *k*-NN with an accuracy of 82.44%. For this model is also the lowest correct classification rate for testing with an accuracy of 73.81%. The highest correct classification rate belongs to the model using MFCC as feature extractor with 36 features and SVM as classifier.

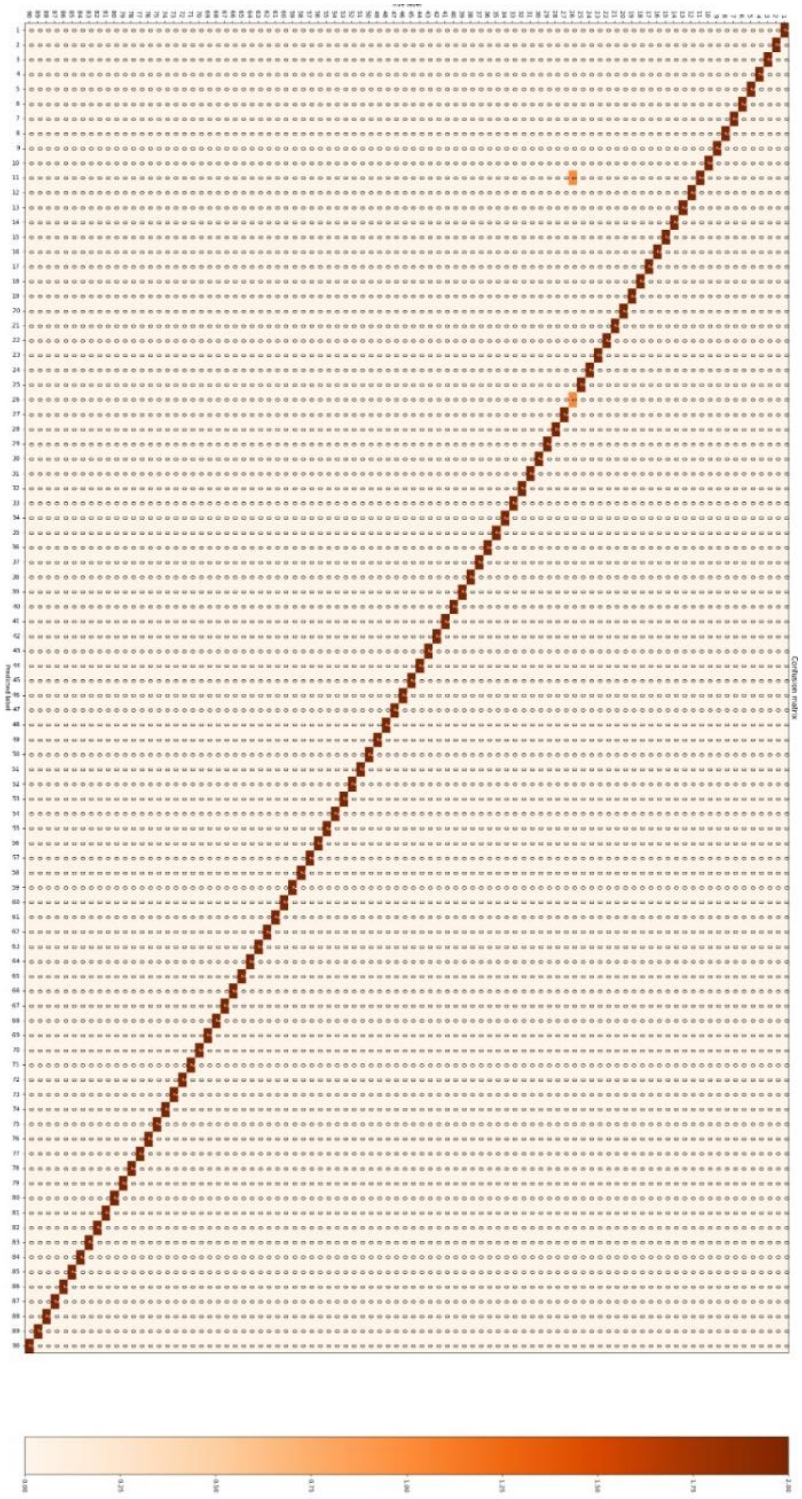

Fig. 6.54 – Confusion matrix for MFCC-36 and SVM, for the first 3 speakers

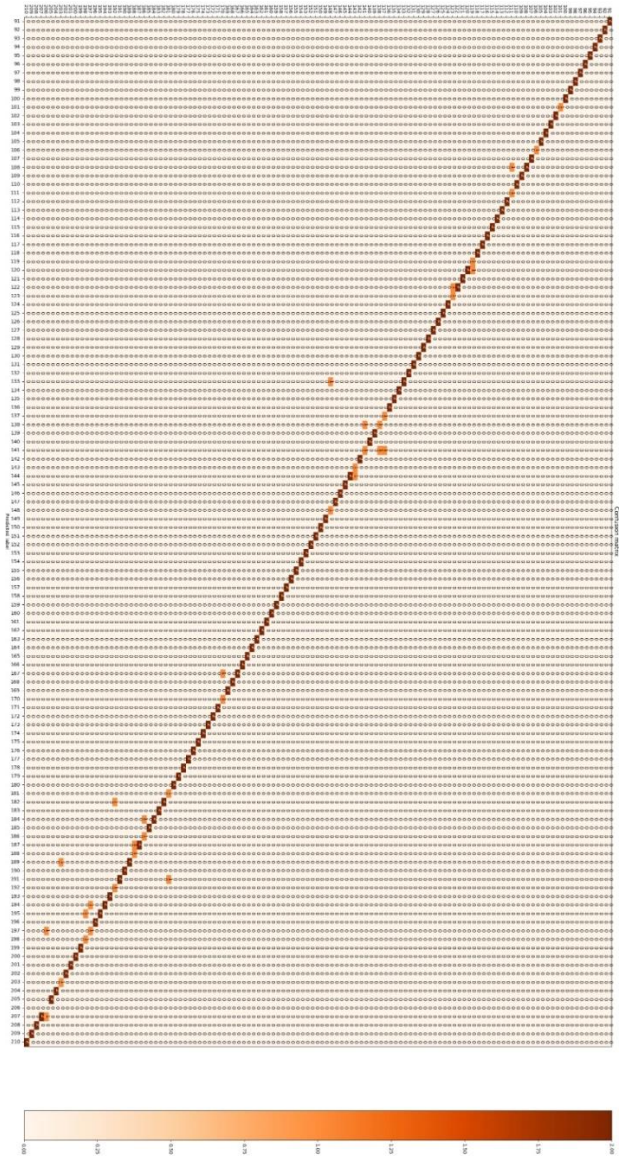

Fig. 6.55 – Confusion matrix for MFCC-36 and SVM, for the last 4 speakers

Because the MFCC model with 36 features and SVM performed best, we created the confusion matrix for the test results in Fig. 6.54 and 6.55. We chose to split the confusion matrix into 2 to aid readability. It can be seen that two speakers had 100% correct classification, 2 other speakers, one male and one female, had only one misidentified sentence. In terms of classification, the last 2 speakers had 6 misidentified sentences. One male speaker was misclassified as a female speaker for 2 sentence instances. From what we observed, a person's gender does not influence the rate of correct classification. The female speaker was misclassified for the sentence "I'm very well.", said in a cheerful tone, which was mistaken for the sentence "The interpretation was wrong.", said in a neutral tone. The male speaker who had only one misclassified sentence was misclassified for the sentence "The door is closed." Spoken in a neutral tone, which was classified as "It has dropped a lot" spoken in the same tone. The most common misclassification of emotion was from neutral to sad. The least common misclassification of emotion was from happy to sad.

General conclusion

As a result of our work, we have managed to create a new database that can be used for research in the emotion recognitions field. The scope of the database is to come as a help for robots to understand the human primary emotions. With this database we can start the development of emotional intelligent robots that can understand the emotional subtext in the Romanian language. The database comes as a help because it is first of its kind in recorded in the Romanian language. The emotional database has 3 different emotions: happy, neutral and sad. There are 10 sentences that can have different meanings depending on the emotion that is used alongside with. The sentences are "The test results are in." , "I'm out of medication." , "The test is positive." , "The interpretation was wrong." , "It's grown a lot." , "It's gone down a lot." , "I closed the door on him." , "I turned off the light." , "I'm very well." , "The keys are in the door.". The people recording the sentences were 4 males and 3 females. The age of the subjects was between 24 and 28 at the moment of registration. Each of the presented sentences have a different meaning for each emotion that is incorporated with the sentence. With those sentences understood by a robot, including all their meanings, the bots can come as a physical and mental help for humans.

Having done this database, another barrier between humans and robots is taken down, as robots can now start to understand our emotions and can act accordingly. Without the emotional feature, robots can be perceived as cold and distant. Sometimes the robots make the wrong decisions even if they understood the sentence transmitted. This usually happens when the message is transmitted through the paraverbal language that includes emotions and body language.

Here comes a limitation of the database. The people were very close in age and for a more reliable database there should be more people involved that could expand the age gap. In this database there should be registered also children and elder citizens. As Romania is a diverse country from the diversity and ethnographic point of view, people from all over the country should be included in the database in order to vary the accents and the dialects.

Another limitation of the database comes in the number of registered emotions. We can see that only 3 emotions are used. From the studies of emotions we saw that there are numerous emotions that a human being can experience. Having more emotions would help robots and machines to understand more of our behaviour.

This thesis brings contributions to the field of emotion recognition. First of all, the newly created database in Romanian language opens new opportunities of research. Having a new database in a language that was not included in the emotion recognition field is a great achievement. Studies can be done to see how similar a country can be with others in terms of emotions. It can also be studied the easiness of transition from one emotion to another and to see how people from a certain country can shift their emotions.

Secondly, a great contribution that comes with this thesis is the study done in comparison between emotional databases in other languages and the newly created database in Romanian language. It is possible to see that for different languages, different feature extractors give the best results. It is interesting to see how much does the F_0 also fluctuate when the emotions do change.

Next, in addition to the analysis performed, we have also added the characteristics for F_0, Mel and Chroma to see if they will improve the correct classification rate. It is fascinating to observe that in some cases the correct classification rate went down after adding the mentioned characteristics in the analysis, but for our new database the correct classification rate has improved. It was also interesting to see that by only using the previous mentioned characteristics, for the Romanian database the correct classification rate was between 35% for Chroma and 50% for F_0.

Finally, if we talk about the highest correct classification rate for the Romanian database, we have obtained a maximum of 97.08% using NGCC as feature extractor and SVM as classifier. In order to obtain this result, we have also added the following features to help in the classification: F_0, Mel and Chroma.

Shortly, the main contributions of this thesis can be synthesized as the following points:

- Gathered volunteers and managed to change their emotions in order to create an emotional database.
- Created an emotional database in Romanian that can be used for research in the robotics field.
- The emotional database created has 3 emotions: happy, neutral and sad, which are the most common emotions in our everyday life.
- Tested multiple databases in different languages to discover that for different languages, the best algorithm differs.
- The algorithms created are composed of different feature extractors and classifiers, that in their turn can also be adjusted with the help of different specific parameters.
- Created an optimized algorithm, based on the tests done on the preexisting databases, for the database in Romanian language that has an accuracy comparable with the accuracy for speech recognition.
- Created a stronger bonding between humans and robots as robots can now surpass random guessing when the audio message transmitted is via emotions.
- Robots can now come as a help when talking about mental health monitoring and social interactions.

References

[1] "Python programming language" En.wikipedia.org, 2023, [Online], Available: https://en.wikipedia.org/wiki/Python_(programming_language) [Accessed: 05- Nov- 2023].

[2] F. Gao, C. -H. Chen, J. -G. Hsieh and J. -H. Jeng, "Support Vector Classifier Trained by Gradient Descent," 2021 International Conference on Sensing, Measurement & Data Analytics in the era of Artificial Intelligence (ICSMD), Nanjing, China, 2021, pp. 1-5, doi: 10.1109/ICSMD53520.2021.9670839.

[3] Defazio, Aaron, Francis Bach, and Simon Lacoste-Julien. "SAGA: A fast incremental gradient method with support for non-strongly convex composite objectives." Advances in neural information processing systems 27 (2014).