University of Tartu

Faculty of Science and Technology

Institute of Technology

Kirill Rodionov

# Embedded system for real-time emotional arousal classification

Bachelors thesis (12 ECTP)

Computer Engineering Curriculum

Supervisor:

Prof. Gholamreza Anbarjafari

Tartu 2020

# Resümee/Abstract

**Embedded system for real-time emotional arousal classification**

Me, inimesed, oskame kergelt tajuda teiste emotsioone, ning ootame mingi emotsionaalset tagasisidet suhtlemise korral. Masinad, kuid, ei oma emotsioonidega seotud oskust, mistõttu inimese ja masina vastastikmju tundub hingetu ja võõrana. Seepärast, tõhusa emotsiooni tunnustamise arendus on üks ülioluline samm inimesesarnase tehisintellekti suuna. Tava inimene ka saab leida kasu emotsiooni tunnustamises. See saab aidata inimesi, kellel on erinevate põhjuste tõttu nõrk kontroll oma emotsioonide üle või nad ei saa teiste emotsioone tundma.

Käesolev töö keskendub kompaktse riistvara baseeritud lahenduse peale emotsiooni liigitamiseks sõltuvalt temast erutusest. Selleks, emotsiooni puudutav teooria oli kogutud, mille pärast arvukad masinõppimise ja tunnuste ekstraheerimise meetodid olid vaadeldatud ja ära proovitud. Need meetodid on tugivektor-masinad, otsustusmetsad, näoorientiiri tunnuste ekstraheerimine ja suunatud gradientide histogramm.

Kehva tulemuste tõttu projekt jäi seisma: väikese mastaabi riistvara kujunes vimetuks laiaulatusliku masinõppimise sooritamise jaoks. Seda saab jätkada, kui lisada projekti võimeka riistvara, et ta treeniks tajumiste muudelit ja edastaks kompaktsele riistvarale juba eeltreenitud muudelit rakendamiseks.

**CERCS:**

**Märksõnad:** masinõppimine, emotsioon, pilditöötlus, tugivektor-masin, otsustusmets, scikit-learn, tehisintellekt

**Embedded system for real-time emotional arousal classification**

We, humans, can distinguish the emotions of others with ease and we always expect any sort of emotional response during a conversation. Machines, however, do not possess emotion related skills, which makes human-machine interactions feel alien and soulless. Therefore, development of an efficient emotion recognition system is one of the crucial steps towards human-like artificial intelligence. A common person can also find use in emotion recognition. It would be a great help to the people, who by various reason either have weak control over own emotions or devoid of any ability to perceive emotions of others.

This thesis focuses on creating a solution based on compact hardware to classify emotions in relation to its level of arousal. For this, theory concerning the emotions and their classifications were gathered, after which numerous methods of machine learning and feature description were reviewed and tried out. The methods list support vector machines, random forests, facial landmark feature extraction and histogram of oriented gradients.

The project has came to a halt halfway through due to poor results: small scale hardware appeared unsuitable for extensive machine learning operations. It can be resumed with the possibility of introducing another set of hardware purely for recognition models training and leaving the compact one deal with pre-made model.

**CERCS:** T125 Automation, robotics, control engineering; T111 Imaging, image processing

**Keywords:** machine learning, emotion, image processing, support vector machine, random forest, scikit-learn, artificial intelligence

# Contents

# List of Figures

# List of Tables

# List of abbreviations, constants etc.

**ADHD** - Attention-deficit/hyperactivity disorder

**AI** - Artificial Intelligence

**ANS** - Autonomic Neural System

**BFS** - Behavioral Facilitation System

**BIS** - Behavioral Inhibition System

**CNS** - Central Nervous System

**CPU** - Central Processing Unit

**CV** - Cross Validation

**DNN** - Deep Neural Network

**EC** - Emotion Classification

**GPU** - Graphics Processing Unit

**GSCV** - GridSearchCV

**HMI** - Human-Machine Interface

**HOG** - Histogram of Oriented Gradients

**Lip-sync** - Lip Synchronization

**ML** - Machine Learning

**NA** - Negative Activation

**ovo** - one-vs-one

**ovr** - one-vs-rest

**PA** - Positive Activation

**PANA** - Positive Activation - Negative Activation

**Poly** - Polynomial

**RBF** - Radial Basis Function

**RF** - Random Forest

**RFC** - Random Forest Classifier

**RSCV** - RandomizedSearchCV

**SVM** - Support Vector Classifier

**SVM** - Support Vector Machine

# 1 Introduction

Nowadays people come across with digital image processing pretty much every day, be it either applying a filter on their photos on the social network or adding a funny caption to a picture found on the Internet. But these are but smaller, simpler capabilities of image processing. In time, much more sophisticated technologies should reach the general public popularity. Technologies such as emotion classification, a real-time one to be exact. Why would it? What could it offer to a future consumer?

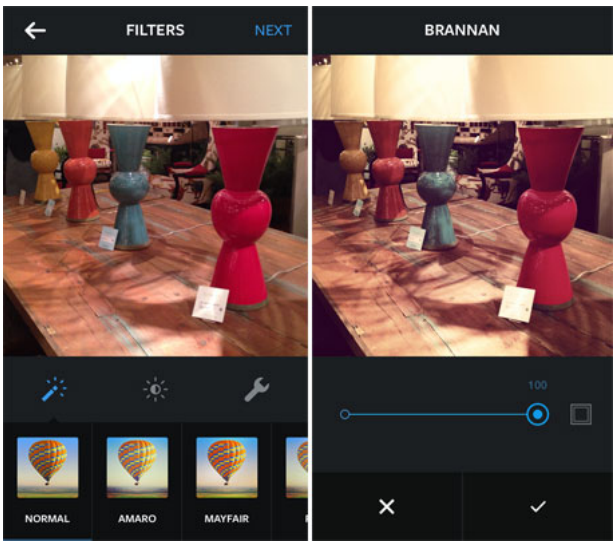


Figure 1.1: A most common application of image processing - applying a filter using nowadays popular mobile app Instagram [1]

We as humans subconsciously use our facial expressions and gestures to convey own emotions, feelings and disposition towards certain topics or things brought up during communications. Large, plain, emotionless pieces of information, most of the time, are automatically disregarded by our brains for lack of interest. Even in modern text-based conversations one may spot, how people often mimic their actual smiles with pictures or ideograms, such as emoji. If people spend time manually looking for an image co-responding to their inner feelings, then

that means it is information, which is regarded to be highly important for others to perceive. Ignoring such valuable information would be foolish, wouldn't it. If so, what actual applications could possibly benefit from this data?

To begin with, Emotion Classification (EC) would greatly benefit Human-Machine Interfaces (HMIs). It is believed that disabled people would receive a better and somewhat faster control over assistance equipment like wheelchairs. For example, during a stroll outside a quick detection of fear might stop the movement of said wheelchair preventing a collision or other sorts of accidents [2]. Moreover, human emotion recognition is a step towards developing human-like Artificial Intelligence (AI) [3]. AI without a proper emotional response is easily detectable for a living person. In services like healthcare, artificial nursing assistants with the ability to understand emotional feelings would diagnose hospital patients more efficiently. Patients themselves would not feel alienated and would be more open to answering treatment necessary questions.

In the fields of recreation, like film and computer game industries, EC would also find itself in a successful demand. These are extremely profitable business areas, where computer graphics are a top requirement. Movie goers and gamers demand ever more beautiful picture with each consecutive year. Creation of believable imagery in turn demands lots of visual references, thus needing a lot of time, money and multiple graphics designers and actors. For this tasks implementation of assistance algorithms has already been debuted in high budget film and computer game productions which saves both money and time [4, 5].

As of more Real-time application examples, quite recently a new The Simpsons episode have aired on television featuring the main character, Homer Simpson, answering live phone calls from the viewers. Dan Castellaneta, the actor behind the character, would improvise in a dialogue with a viewer, while a program named Adobe Character Animator would track actor's voice generating co-responding Lip Synchronization (Lip-sync) animation [6]. In Japan a similar experience has boomed in popularity, however on a different media - the internet. A new phenomena was born: "Virtual Youtuber". It follows a basic premise of Youtube Streaming, although one sees a 2.5D Computer Generated Model of a girl, voiced by an anonymous actor, playing some computer game instead of an actual person [7].

Not only does a big production company find use of this technology, but a typical everyday life gamer can have multiple ways of utilising it as well. An interactive virtual avatar immensely expands the possibilities of cyberspace communications with the rise in popularity of such games like VRChat. In VRChat players have a vast range of their avatar customization and avatar manipulation to the lengths of Full-Body Tracking. However, one thing lies missing: despite any dynamics in tone or facial expressions, the avatar's face remains static, which leaves out a crucial point of human interaction as described above.

Stepping away from recreation topics, there are people suffering from disorders, which can alter the ability of emotion experiencing (e.g. Bipolar Disorder, when a person's emotions unconsciously fall into quickly fluctuating extremes [8]), or which deny emotion perception of others, such as the case of Social-emotional agnosia [9]. Having a small device in a pocket capable of assisting in emotion understanding of both self and others would greatly impact their lives.

## 1.1 Problem overview

So, if the case of interactive avatar feature is so valuable, why not implement all possible motion trackers and ECs into computer game or virtual communication software right away? The difficulties may arrive with hardware limitations, due to implemented algorithms' tendencies to consume huge chunks of computational power. This is a vital obstacle especially for computer game streamers, who most of the time have no luxury redirecting paramount operative force from computer graphics rendering. In such case one should look for options to either replace components of one's computer with newer, more efficient and more expensive counterparts or expand already existing rig utilising application-specific add-on.

The latter solution would take up the image processing in its entirety saving the rest of the computer from a computational overload just like a GPU assists a CPU. It's main advantage is alleviation of all set-up planning from the user providing a fully-ready out-of-the-box system.

As for the emotion perception assistant tool, a big, heavy and clunky apparatus would be impractical to carry around and use. Such tool should be of reasonable size and weight. Ideally, it would be around the physical qualities of a smartphone, a wallet or a power bank.

Developing of such solution shall be the main focus of this thesis paper.

## 1.2 Goals

The solution must represent itself as a complete embedded system of hardware and software dedicated to performing all the necessary computations for a real-time emotional classification.

# 2 Problem review

## 2.1 General look at emotions

To begin analyzing and classifying emotions, one must first understand them. This is not an easy task: psychologists till this day still struggle giving a definitive description, because again it is something everyone seem to have a basic in-built realisation of. This vague grasp on the topic can be observed when looking up the definition of the word ”emotion” in various dictionaries:

- a strong feeling deriving from one’s circumstances, mood, or relationships with others [10].
- a strong feeling such as love, fear or anger; the part of a persons character that consists of feelings [11].
- an affective state of consciousness in which joy, sorrow, fear, hate, or the like, is experienced, as distinguished from cognitive and volitional states of consciousness [12].
- any strong agitation of the feelings actuated by experiencing love, hate, fear, etc., and usually accompanied by certain physiological changes, as increased heartbeat or respiration, and often overt manifestation, as crying or shaking [12].
- An emotion is a feeling such as happiness, love, fear, anger, or hatred, which can be caused by the situation that you are in or the people you are with [13].
- Emotion is the part of a person’s character that consists of their feelings, as opposed to their thoughts [13].
- a conscious mental reaction (such as anger or fear) subjectively experienced as strong feeling usually directed toward a specific object and typically accompanied by physiological and behavioral changes in the body [14]

In addition, Wikipedia provides this description: Emotion is a mental state associated with the nervous system brought on by chemical changes variously associated with thoughts, feelings, behavioural responses, and a degree of pleasure or displeasure [15].

To summarize, emotion is a state of a person's feeling, which dictates a person's behaviour, gestures, voice, posture and facial expressions. Coincidentally, we humans observe these features to determine the emotional state of others. People are able to perform these predictions seemingly automatically; understanding of individuals in a society is beneficial to our communal survival. But how did we received such a complex yet useful ability. One branch of theories states that emotion recognition bears a cultural origin; that people are taught since their early years to distinguish emotions within the boundaries of their upbringing. Contrary to that, another belief suggests an evolutionary origin, meaning that emotion recognition is innate and universal between all the individuals, no mater where they are from. So, which of these theories are correct? As shown in the article Universal Facial Expressions of Emotion by Paul Ekman, both of them bear a bit of truth. People have both the innate ability of emotional communication along with learned culture specific traits [16, 17]. Computers, however, lack any sort of prior skill in this field, and thus they must be taught from the ground up.

To begin our road towards the solution, we have to decide upon the emotional model, within constraints of which our future machine will try to operate. Various researchers attempted different approaches to formulating a definitive human emotion model. The majority of models fall into two categories:

1. Dimensional models; these models suppose, that every emotion could be placed on continuous axes tied to some descriptor of given emotion [18].

2. Discrete models; these models view each and every emotion to be independent occurrences, or have a list of core emotions which can constellate into different complex emotions [18].

Let's look at these approaches more closely.

## 2.2 Dimensional models

This group of models, as apposed to dividing emotions into specific independent classes, provide a clustered view on a continuous space, where human emotions represent more-or-less vaguely bordered subsections of said space. One of the earliest models by Wilhelm Max Wundt placed a person's feelings on the 3 axis of pleasurable and unpleasurable, arousing and subduing, straining and relaxing [19].

Later studies, however, reported the third dimension as either small or seemingly non-present, which leads to nowadays popular dimensional models usually incorporating only 2 dimensions [20, 21]. Such are the cases of the circumplex and the vector models, that value valence (pleasure  displeasure) and arousal. Despite the identical axis definition, the models differ in their arrangement of emotions inside of the two-dimensional plane. The circumplex model strives to allocate emotions along a circular pattern (hence the name) with a center at the intercrossing point of the axis, a point of neutral valence and medium arousal [22, 23]. The vector model, as the name implies, has vectors in its base structure: two vectors spring from the common point of zero arousal and neutral valence, although heading into two opposing directions: one vector extends into the region of negative valence, while another - into positive one. This fundamental dissimilarity between the respective frameworks of these models, spawn a disagreement concerning the existence of an emotion of neutral valence and high arousal descriptors. The circumplex model hints at the possibility of such emotions (alarmed and interested being valid candidates), whereas the vector model outright renounces such a phenomena [23].

In addition to these models, there is also a proposal of utilising the "consensual" Positive Activation - Negative Activation (PANA) model, which is claimed to be an alternative rotational view of the circumplex model by placing in the same emotional plane its own axis 45° away from valence and arousal. These axis originally named Positive and Negative Affect in the work of David Watson and Auke Tellegen are described as following: "The first factor, Positive Affect, represents the extent to which a person avows a zest for life. The second factor, Negative Affect, is the extent to which a person reports feeling upset or unpleasantly aroused". Negative Activation (NA) is described by words such as distressed, fearful and scornful on its High end, while with relaxed, placid and calm on the Low end. Positive Activation (PA), in turn, is depicted with words as active, enthusiastic and excited, along with drowsy, sleepy, sluggish in

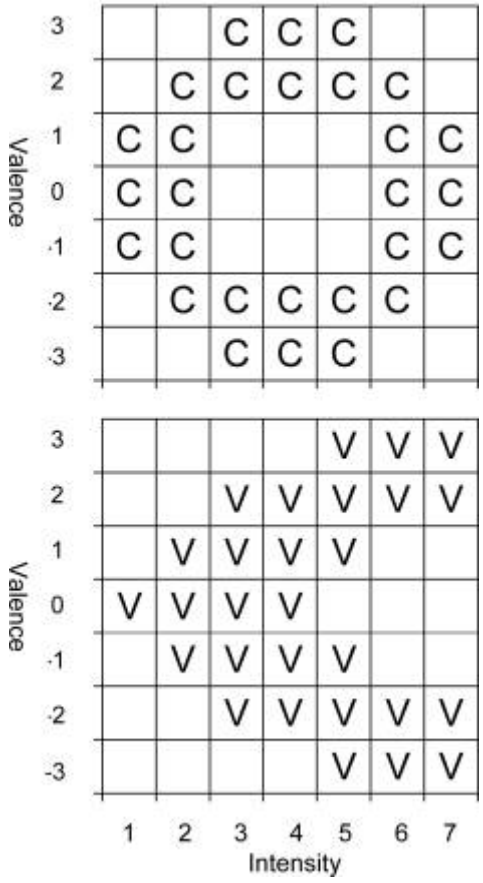


Figure 2.1: Representations of the emotion placement patterns on the emotion plane, according to the circumplex model (top panel) and the vector model (bottom panel) [23]

the High and in the Low states respectively , [21]. This model has trouble finding words denoting description for states of simultaneous High PA and High NA, which are represented by the high arousal and neutral valence in the circumplex model. Coupled with the statements, that PA and NA are ”truly unipolar constructs that essentially are defined by their high poles”, ”the activated, high ends of the dimensions fully capture their essential qualities” and ”the low poles of these dimensions ultimately reflect the absence of a certain kind of activation rather than the presence of a certain affective state (such as sluggishness or relaxation)” results in this model acting akin to the vector model, in spite of being based around the circumplex model [23, 24].

In some particular cases of studies, such as study of human autobiographical memory, one may add a third dimension - intensity - to the 2D models from above. It must be said that this additional dimension is not as much of general emotion descriptor, but more like a representative of a subjective evaluation of experienced feeling. In previously mentioned example of the autobiographical memory, intensity of an emotion is correlated with likelihood of recall, independently of emotion’s arousal or valence [23].

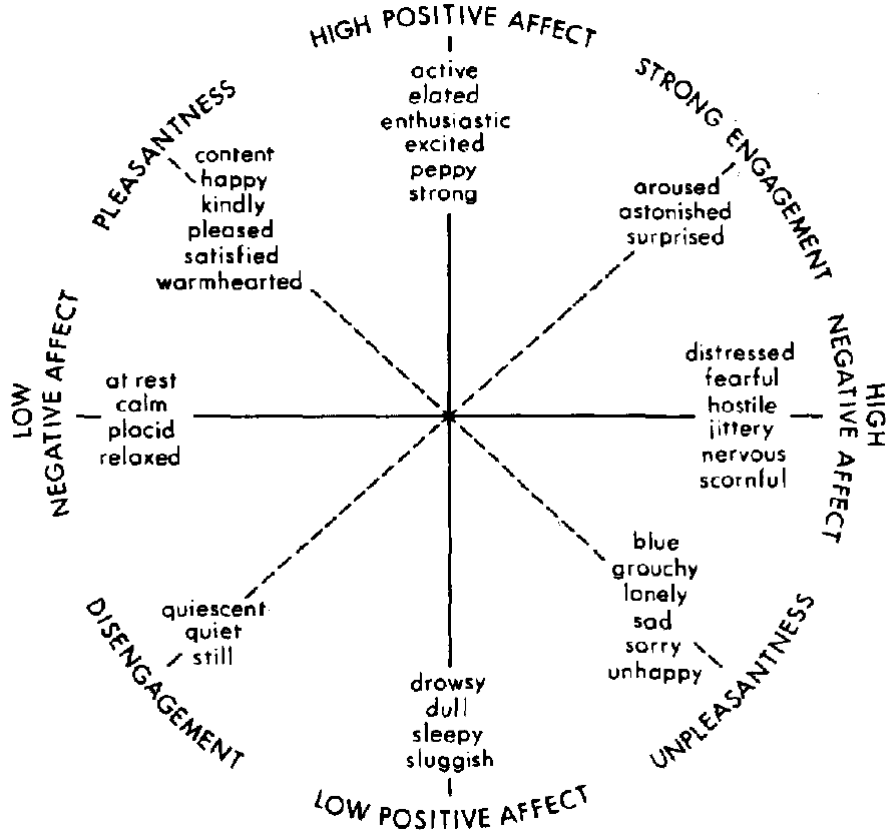


Figure 2.2: "The two-dimensional structure of affect" depicts the relation between the circumplex and the PANA model [21]

Application of dimensional models finds most of its popularity in the fields of psychiatry, neuroscience and behavior studies. It is hinted that the mesolimbic pathway of human Central Nervous System (CNS) , responsible for pleasure and reward, also plays a role in assessment of negative emotions, thus encompassing a structure for valence measurement. Also, a greater activation of the right frontal lobe can be observed in times when subject experiences intervals of negatively valenced emotions, whereas in times of positive ones the left frontal lobe enters a similar state. In a likewise fashion, activity among the Reticular Formation networks and amygdala corresponds to the degrees of arousal. Coincidentally, common psychiatric comorbidities with symptoms of hyperarousal such as Attention-deficit/hyperactivity disorder (ADHD), bipolar disorder and anxiety disorder are tied with abnormalities of Reticular Formation and amygdala [22].

The PANA model has also found its part in previously mentioned fields of studies, particularly assisting to characterise BIS and BFS. BIS stands for behavioral inhibition system is an evolu-

tionarily adaptive motivational system that dictates withdrawal behaviors. It helps an organism to avoid dangerous and possibly harmful objects, subjects, activities etc. It encourages vigilant analysis of the surroundings and cautious plan of movements. Just as the experiencing feelings tied to NA awakens the state of attention and anticipation for painful or punishing outcome. The BFS stands for behavioral facilitation system. Contrary to BIS, this system leads an organism towards beneficial to survival resources and, as one may have already guessed, links itself to PA. Feelings from the PA dimension act as a driving force of getting food, water, shelter, socialisation etc and upon achieving a goal serve as a subsequent reward as well. Links between the activations and behavioral systems one can observe in the distribution of activations over time. PA has a almost a cyclical nature in the waking hours and throughout season in order to continuously motivate an organisms survival, shifting priorities from one resource to another, whereas NA quickly peaks in the moments of trouble and vanishes suddenly along with the danger for fear of unnecessary resource depletion and physiological exhaustion [24].

Finally, it should be noted, that people find it difficult to exactly discern an emotion they are experiencing. When communicating, one is prone to use several similarly valenced emotions to describe oneself, a phrase ”feeling good” tends to be accompanied by words such as ”excited”, ”engaged”, ”cheerful” etc. This shows how we perceive emotions not as isolated states, but as a continues spectrum; like we perceive colours. All of this comprise strong arguments in favor of implementing the dimensional models for emotion classification [22].

## 2.3 Discrete models

As previously mentioned, this type of models suggest every emotion or a selected group of emotions to be an independent phenomenon possessing distinctive characteristics, e.g. facial expressions, vocal tones, behaviour. Fear compels us to flee; disgust dissuades consuming noxious substances - this provides a an intuitive background for discrete models implementation. But there are just so many emotions. Because of this Tiffany Watt Smith managed to describe 154 different emotions in her book called ”The Book of Human Emotions”. The most prominent difficulty arises with international researches due to some cultures having emotions specific only to them [18]. In this case, perhaps, not all emotions are equal in their importance and origin. Out of this arises a term ”basic emotions”: a group of emotions which are universal

to everyone and which are easily recognised by everyone, regardless of their cultural upbringing. The initial point of modern ”basicality” research has been triggered by Charles Darwin with his book titled The Expression of the Emotions in Man and Animals, where he pointed out the importance emotions have in survival as a mean of communication, thus resulting in a necessity for quick distinguishability. This kind of definition roots heavily basic emotions with evolutionary origin, i.e. basic emotions have been steadily developed throughout the course of humankind's history to subconsciously combat with fundamental life tasks [25]. Despite sharing this common framework, researchers provide a large variety of lists reciting basic emotions.

Inspired by Darwin, Silvan Tomkins in his career has proposed the nine affects, which are innate biological building blocks for emotions. These affects are Distress-Anguish, Anger-Rage, Fear-Terror, Shame-Humiliation, Disgust, Dissmell (negative affects), Surprise-Startle (neutral affect), Interest-Excitement, Enjoyment-Joy (positive affects). The affects named with two descriptive words represent the least and the most intense expression of that affect [26, 27].

The next researcher Paul Ekman has also adopted ideas from Darwin as well from Tomkins himself. While studying the nature of human facial expressions along with his colleagues, Ekman has revealed the existence of a number of basic emotions, which seem to be present in every culture all across the world, even the non-literary ones. This list includes happiness, sadness, anger, fear, surprise, disgust, with contempt joining the list later on - emotions which can be easily observed on and decoded from a person's facial expression [16, 28, 29]. Ekman has continued his investigation in the field of basic emotions. Eventually, in 1999 he proposed several characteristics in hopes of providing necessary points to help ”distinguish basic emotions from one another and from other affective phenomena”. He remarks, that non of the following traits should be treated as sine qua non:

1. Distinctive universal signals
2. Distinctive physiology
3. Automatic appraisal, tuned to:
4. Distinctive universals in antecedent events
5. Distinctive appearance developmentally

6. Presence in other primates
7. Quick onset
8. Brief duration
9. Unbidden occurrence
10. Distinctive thoughts, memories images
11. Distinctive subjective experience

Using these guidelines Ekman has further expanded his previous list of basic emotions with the ones not explicitly coded in facial expressions. An attention is also directed to the fact, that these should be viewed as ”families of related emotions”. The newly updated list comprises: amusement, anger, contempt, contentment, disgust, embarrassment, excitement, fear, guilt, pride in achievement, relief, sadness/distress, satisfaction, sensory pleasure and shame [30, 31].

As an argument in favor of proposed basic emotions, several patterns of Autonomic Neural System (ANS) activity have been traced coinciding with experience of either happiness, sadness, anger, fear or disgust. Since this patterns have been observed in a variety of different cultures, this yet again hints at the innate evolutionary origin. But a few inconsistencies should be pointed out. First of all, not every basic emotion imply possessing an ANS activity pattern. Ekman counters this by saying that there should not be any specific ANS activity tied to an emotion in the first place if the emotion lacks a specific motor behaviour purposed for performing specific actions. As an example he provides fighting as such action for anger, which includes in its ANS pattern increased blood flow into fists. Same parallel can be drawn for fear, fleeing from danger and major blood flow redirection toward large skeletal muscles [30]. Another inconsistency emerges with different sub-types within emotional families. Response of crying and non-crying sadness(es) differ in cardiovascular activity, increased and decreased respectively. Similar divergence can also be observed among the sub-types of other emotion families, which are reported to have a particular ANS activity pattern, In addition to basic emotion specific activity of ANS, the must also be present one in CNS. In the past decade a handful of studies have found some sequences associated with happiness, sadness, anger fear and disgust, however there is still much debated concerning the specific components of CNS responsible for this [25, 30].

In many aspects Robert Plutchik shared Ekman's view concerning the existence of biologically hardwired emotions. He in turn advocated for his own list of basic emotions. Moreover, he presumed that basic emotions can merge together producing secondary emotions, unlike Ekman who has doubted the notion of multiple basic emotions occurring simultaneously [30], In a work titled "A general psychoevolutionary theory of emotion" Plutchik provided his own 10 postulates regarding the basic emotions model:

1. The concept of emotion is applicable to all evolutionary levels and applies to animals as well as to humans.

2. Emotions have an evolutionary history and have evolved various forms of expression in different species.

3. Emotions served an adaptive role in helping organisms deal with key survival issues posed by the environment.

4. Despite different forms of expression of emotions in different species, there are certain common elements, or prototype patterns, that can be identified.

5. There is a small number of basic, primary, or prototype emotions.

6. All other emotions are mixed or derivative states; that is, they occur as combinations, mixtures, or compounds of the primary emotions.

7. Primary emotions are hypothethical constructs or idealized states whose properties and characteristics can only be inferred from various kinds of evidence.

8. Primary emotions can be conceptualized in terms of pairs of polar opposites.

9. All emotions vary in their degree of similarity to one another.

10. Each emotion can exist in varying degrees of intensity or levels of arousal.

[32, 33]. In order for one to understand more clearly his proposals, Plutchik has created a so called The Emotion Wheel (Figure 2.3).

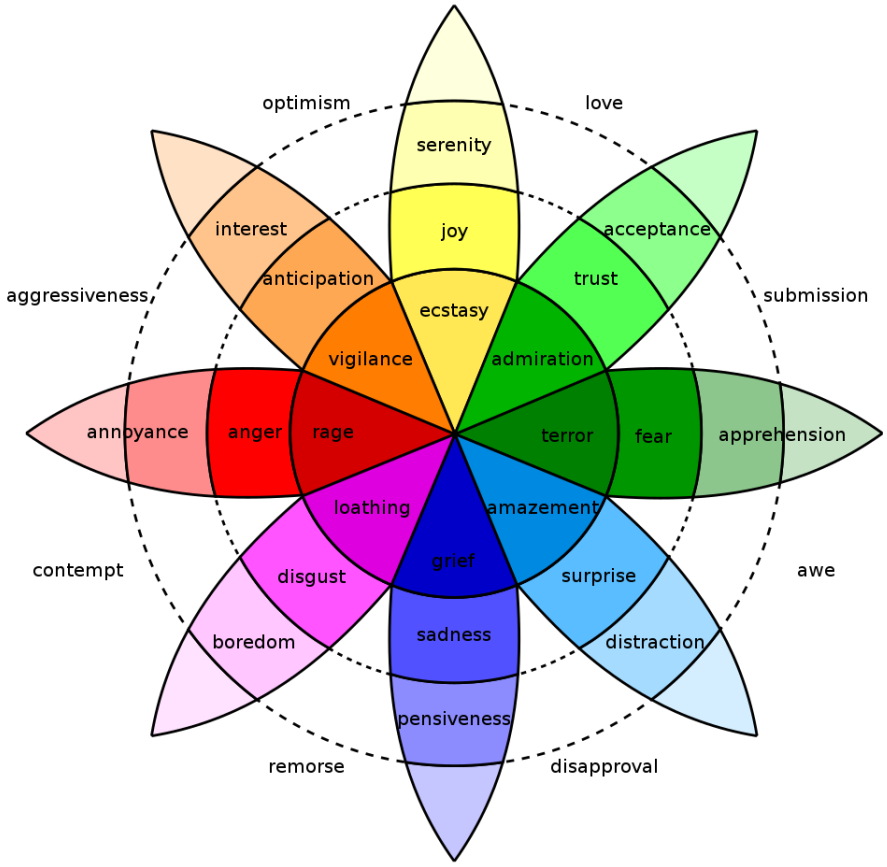


Figure 2.3: Robert Plutchik's Wheel of Emotions [34]

One can see the list of basic emotions in this case contains 8 emotions, divided into pairs of polar opposites: Joy vs Sadness, Trust vs Disgust, Fear vs Anger, Anticipation vs Surprise. One may also observe how a basic emotion adjusts with a change of intensity: more intense version of fear is terror, whilst morphing into apprehension with dropping of intensity. As mentioned previously, these basic emotions can form dyads blending into secondary, "non-basic" emotions. For instance, disgust and anger form contempt, joy and fear - guilt, fear and disgust - shame and so on. The further away a pair of basic emotions resides in the emotion wheel, the more seldom a person experiences them, to the point of when it is impossible for a dyad to be formed out of polar opposite basic emotions [32, 33, 35, 36].

It must be addressed, that Plutchik's Emotion Wheel model retains qualities similar to both the discrete and to the dimensional models (e.g.intensity and blending), which makes it stand out from the rest. Although, it is not the only nor is it the first model to represent the concept

of core emotions comprising every other emotion. A few centuries prior Descartes claimed all emotional states to be comprised of 6 basic emotions, passions as he named them, which are joy, sadness, love, desire, hatred and wonder [25,30]. In recent years, researches from the University of California conducted a self-report survey among a broad selection of participants concerning their emotional state after each view of a specific short video. As a result, 27 categories of emotions were obtained, claimed to be distinct and forming a continues intermixing space of emotions [37].

## 2.4 Selecting a suitable emotion model

Having looked at the candidates, which model should be exposed to the machine? We recall, that the most important place for denoting one's emotional state is through own facial expressions. Moreover, providing an image of a persons face to the machine would also be easier than something like scans of neural activity. Therefore, it is a logical decision, to take Ekman's 6 (or 7, whether contempt is differentiated from disgust or not) basic emotions as the basis. In addition, a number of popular data sets of people's facial expressions classifies them using exactly this list. However, we will also take a dimensional model into account, and place our chosen basic emotions along the distribution of vector model inside the space of valence and intensity. This way the machine would not only predict an emotion a user is experiencing, but also estimate user's arousal level and allocate it on the valence spectrum.

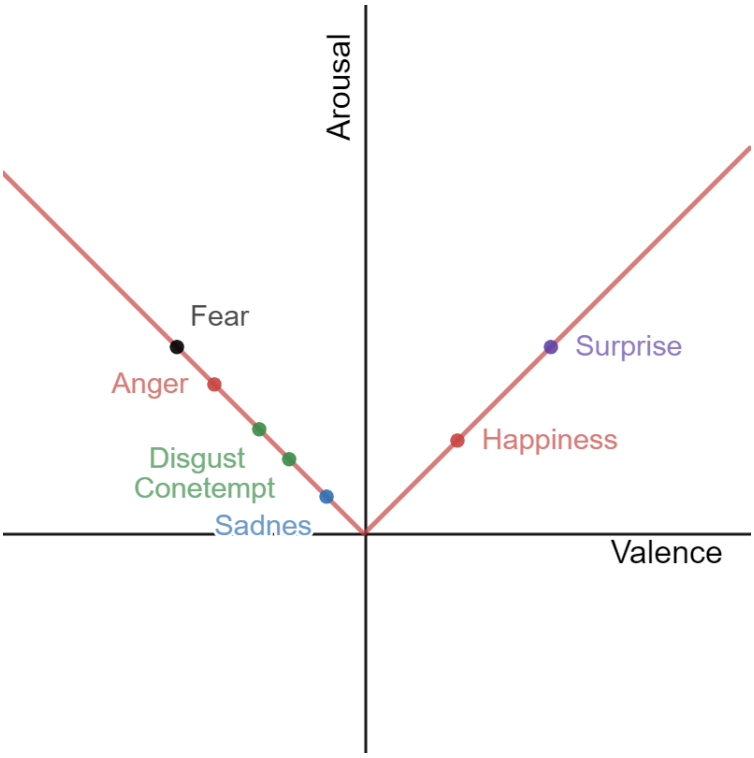


Figure 2.4: Ekman's 7 basic emotions placed along the vector emotion model: the model, which is used in this paper

# 3 Methodology

First of all, a few necessary decisions must be made as a starting point of this project. This decisions mainly encompasses the working basis of an emotion cognition machine such as hardware and software along with their dependencies. Afterwards, we can then develop applications to suit our the needs of the project based on the prior selections.

## 3.1 Hardware

Hardware is a good foundation for any project. It draws concrete limits and forms a path for further development. The previously defined scope requires the hardware to be rather small and compact. For the role of base computing hardware much attention imposed Raspberry Pi product range. It has a relatively small profile and cost, application flexibility and is generally marketed as a cheaper and mobile alternative to a standard PC. And with a recent release of Raspberry Pi 4 family, all this made an ever more compelling reason to implement it in the project, partly to challenge the claim about standard desktop computer equivalent and test its capabilities in an uneasy task, which is the aim of the paper [38, 39]. In order to obtain a brand new Raspberry Pi 4, author has purchased a Starter Kit for Raspberry Pi 4 (Model B) distributed by Labists (Notice that currently the product has been updated) [40]. The kit provided a Raspberry Pi 4 Model B board with the specifications detailed in a table bellow.

Table 3.1: Author's Raspberry Pi 4 Model B specifications [41–43]

| | |
|---|---|
| OS | Raspbian GNU/Linux 10 (buster) |
| Processor | Broadcom BCM2711, quad-core Cortex-A72 (ARM v8-A)<br>64-bit SoC @ 1.5GHz |
| Memory | 4GB LPDDR4-3200 SDRAM |
| GPU | Broadcom VideoCore VI |
| Connectivity | 2.4 GHz and 5.0 GHz IEEE 802.11b/g/n/ac wireless<br>LAN, Bluetooth 5.0, BLE<br>Gigabit Ethernet<br>2 USB 3.0 ports<br>2 USB 2.0 ports |
| GPIO | 40-pin GPIO header<br>(fully backwards-compatible with previous boards) |
| Video & sound | 2 micro HDMI ports (up to 4Kp60 supported)<br>2-lane MIPI DSI display port<br>2-lane MIPI CSI camera port<br>4-pole stereo audio and composite video port |
| Multimedia | H.265 (4Kp60 decode);<br>H.264 (1080p60 decode, 1080p30 encode);<br>OpenGL ES, 3.0 graphics |
| SD card support | Micro SD card slot for loading operating system<br>and data storage |
| Input power | 5V DC via USB-C connector (minimum 3A)<br>5V DC via GPIO header (minimum 3A)<br>Power over Ethernet (PoE)-enabled<br>(requires separate PoE HAT) |
| Dimensions | 88mm $\times$ 58mm $\times$ 19.5mm, 46g |
| Operating temperature | 0 - 50 degrees C ambient |

The kit also came with a SanDisk Ultra 32GB MicroSDHC UHS-I Card Speed Class 10 U1, which had already NOOBS pre-loaded. This card was used as a main storage device for Raspberry Pi 4. Additional storage was utilized in a form of 256GB KingSpec Z3 SCSI external SSD, connected via USB 3.1.

In order to better visualise the results of emotion analysis a display was designed and produced during a course LOTI.05.022 Computer Hardware Project. It is an STM32 microcontroller based board able to control 64 diffused RGB LED. Using colours and simple animation this board can represent users emotional state or attempt to balance out extreme cases of user's emotional arousal. A short demonstration can be viewed on youtube by accessing the following link [44]. The schematic and gerber files can be found here [45] .

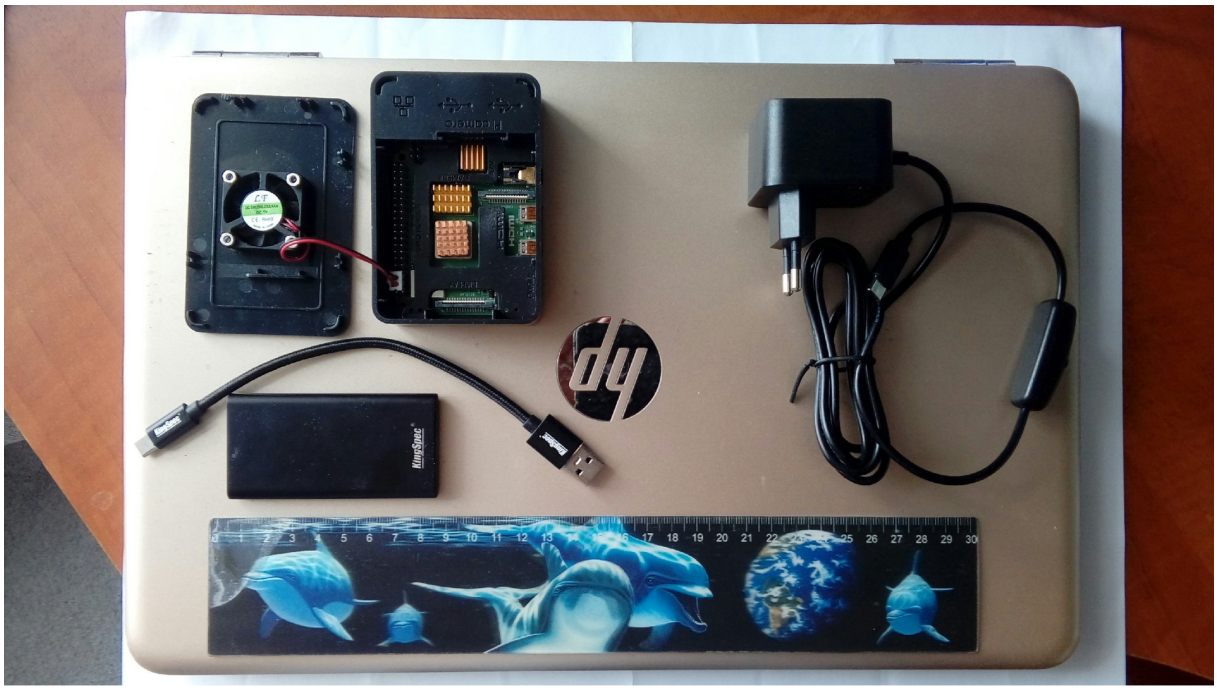


Figure 3.1: A photo of the hardware used to carry out all of the computations, described in this paper: Raspberry Pi 4 Model B, 256GB KingSpec Z3 SCSI SSD and HP Pavilion laptop; ruler for scale

Some additional computations have been performed on author's personal HP laptop. The characteristics are following:

Table 3.2: Specifications of author's personal laptop PC

| | |
|---|---|
| OS | Microsoft Windows 10 Home<br>version 10.0 18362 Build 18362 |
| System Model, Type | HP Pavilion Notebook, x64-based PC |
| Processor | AMD A9-9410 RADEON R5,<br>5 COMPUTE CORES 2C+3G, 2900 Mhz,<br>2 Core(s), 2 Logical Processor(s) |
| BaseBoard | HP 81FC<br>version 80.20 |
| Installed Physical Memory | 8.00 GB |
| Total Physical Memory | 7.45 GB |
| Total Virtual Memory | 24.5 GB |
| Storage | SSD SanDisk SD8SNAT-256G-1006 |

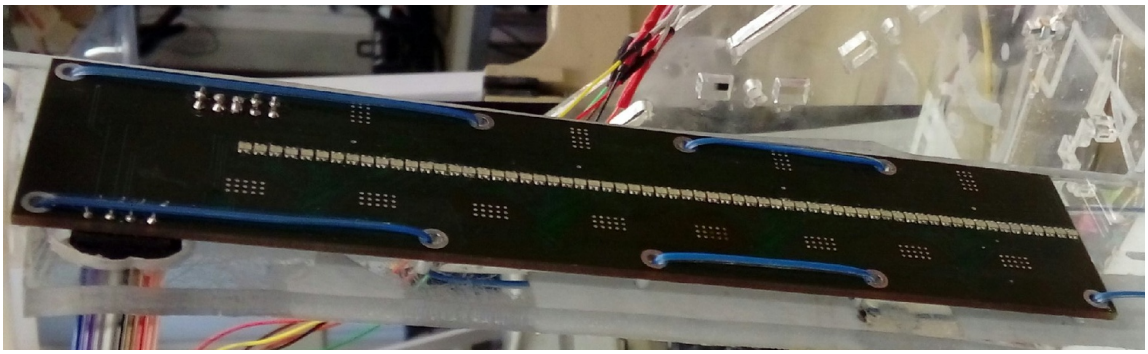

Figure 3.2: A photo of the display board created to be used in this paper

## 3.2 Machine Learning

With the hardware out of the way, we can now focus on the classification algorithms. A small reminder: the classification will be based on the analysis of a user's facial expression. A human face is not an easy pattern to convey, as we usually neglect to realise. It is full of numerous lesser shapes and figures, which come in a broad variety from one person to another. General public does not notice this matter simply because we humans are extremely adept at locating and interpreting faces of other humans. So proficient in fact, that we can human faces among inanimate objects, an occurrence called Pareidolia [46].

Another point, which needs to be addressed, is how computers perceive visual information. Images for them are nothing more than a 3 dimensional or 2 dimensional arrays of numbers, for a colour and grayscale types of image respectively. So it's not only human faces, but even the simple shapes as lines and circles, which are lacking in the repertoire of a blank machine. Therefore, the machine must be taught from the ground up.

Manually defining all the patterns and values for a machine to consider would be tedious and difficult. Luckily, there is no explicit necessity for this approach, for there are ways to make the machine teach itself all of the important computations. This is, of course, all thanks to the machine learning. This technique enables computers to generate experience in specified problem solving based on provided data without human interference whatsoever [47, 48]. This would, hopefully, alleviate greatly the burden on the programming side. Moreover, the author had neither prior knowledge nor experience in this field, so it would also provide opportunities to learn.

It must be said, that machine learning (ML) isn't one exact universal solve-all algorithm.

ML is a plethora of different algorithms banded by common goals. And as the No free lunch theorem claims, none of them can truly outperform any other in every conceivable task [47,49]. In general terms, these algorithms can be divided into two categorise: algorithms of supervised learning and algorithms of unsupervised learning. Supervised learning builds a model capable of formulating predictions based on input. To train such a model, the algorithm must be fed a mass of sample inputs along with their corresponding correct outputs. Unsupervised learning, on the other hand, does not attempt to make any sort of predictions, thus does not require an additional listing of outputs. Its task is to find various correlations, patterns, similarities etc among the elements of provided data, effectively grouping and clustering them [48].

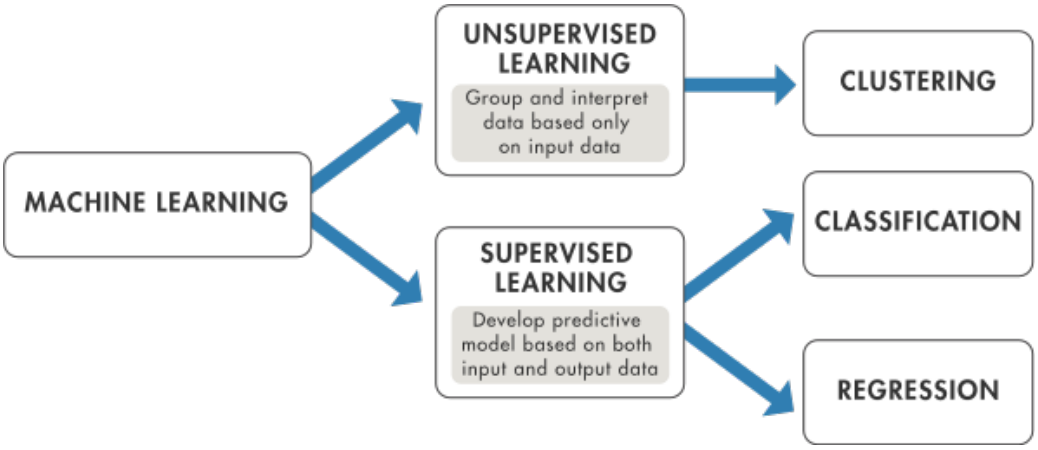


Figure 3.3: A visual representation of ML categories [48]

For this project. the software must be able to accurately perceive a user's emotional state, therefore choosing the supervised learning category is a no-brainer.

By this point the range of candidates has been narrowed down to the algorithms of supervised learning. It is still, however, quite an extensive list. After some web surfing and author's discussion with the supervisor, it was decided to use algorithms such as Support Vector Machine (SVM) and Random Forest (RF). They seem to be the ones of the most popular and efficient algorithms in terms of image processing. One should also mention Neural Networks, specifically Deep Neural Networks (DNN), which are claimed to be the best performers for such task. In fact, during a "Challenges in Representation Learning: Facial Expression Recognition Challenge" imposing a similar task, DNN based solutions generally climbed high up the scoreboard, as far as the top 10 [50, 51]. Problem with them, however, is their dependence of an efficient GPU, which the present hardware lacks.

## 3.3 Support Vector Machine

Let there be dataset, where each element belongs to either class 0 or 1 and possesses n number of features. Then we create an n-dimensional space, where each axis corresponds to one of the features. All the elements are to be placed into the space as a dot based on its features. The job of the SVM would then be to draw an (n-1)-dimensional hyperplane in order to make a clear border separating members of one class from another and the space respectively. With this, as soon as a new input will come for prediction, it can be place into the space and later classified based on its position relative to the hyperplane. For example, if the elements have only 2 features, then the space would be represented by a 2-dimensional plane, whereas the hyperplane would be a straight 1-dimensional line [52–56].

The distance between the hyperplane and the closest points of either dimension is called margin.

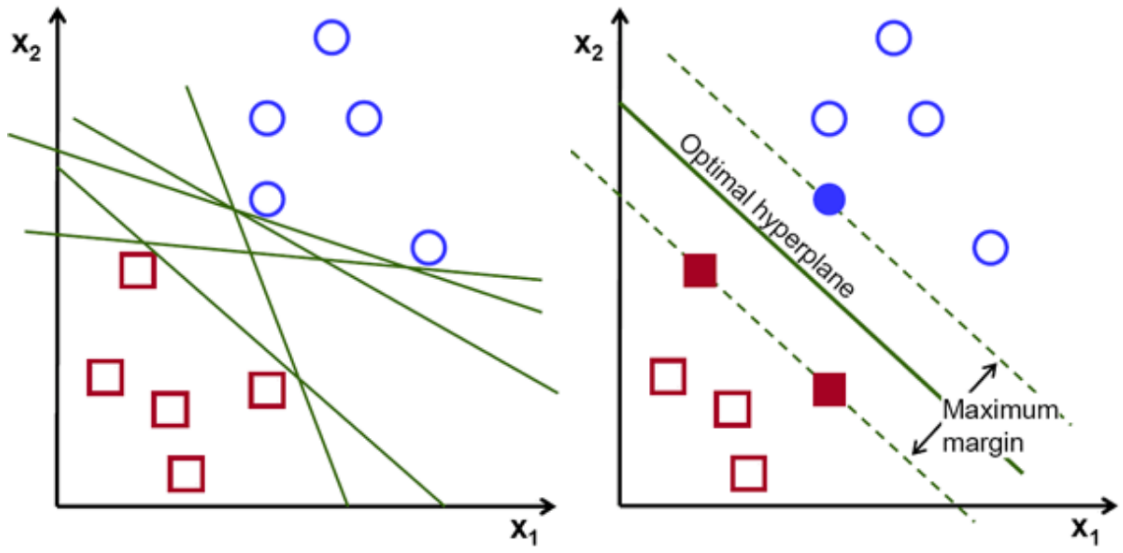


Figure 3.4: A representation of SVM decision making. On the left on may see a several possible hyperplanes, however only the hyperplane on the right provides the maximum margin, and thus is selected [55]

Those points are referred to as support vector points due to defining the hyperplane position. Removal of any other point will not affect the hyperplane, unlike a removal of a support vector point. Margin is also a crucial quality of a model, because it insures robustness of a model. Hyperplane with a maximal margin for both tends to misclassify incoming data less often. In several cases it is hard to achieve decently big margin and include all of the data points. In such cases these unfitting points could be ignored (e.g. by regarding them as noise). Managing

between a good margin and proper point disclusion is the key for achieving a high accuracy performance. Another useful technique, when dealing with tasks, where a linear solution is nil possible, is to map an existing space onto another higher dimensional space and try to draw a higher dimensional hyperplane in there; all by using an assistant function. This assistant function is named kernel and the technique itself - a kernel trick [52–56].

An SVM or working on tasks requiring grouping into more than two classes represents an ensemble of multiple binary SVMs, which follow either ”one-vs-rest” or ”one-vs-one” strategy. In case of ”one-vs-rest” each SVM takes one specific class and composes all the remaining classes together in to a single one. In the other case, each class is compared to each other class in duels [52–56].

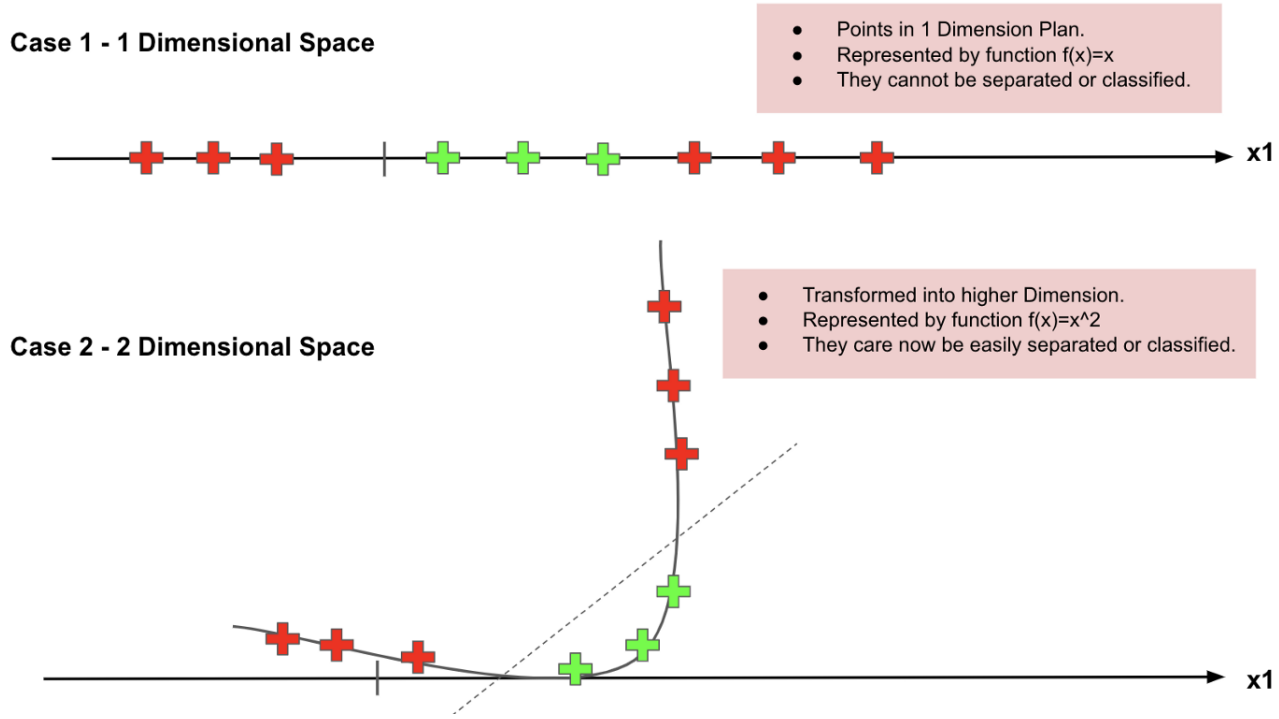


Figure 3.5: An example of how a kernel trick operates on a 1 dimensional problem, by adding a second dimension [56]

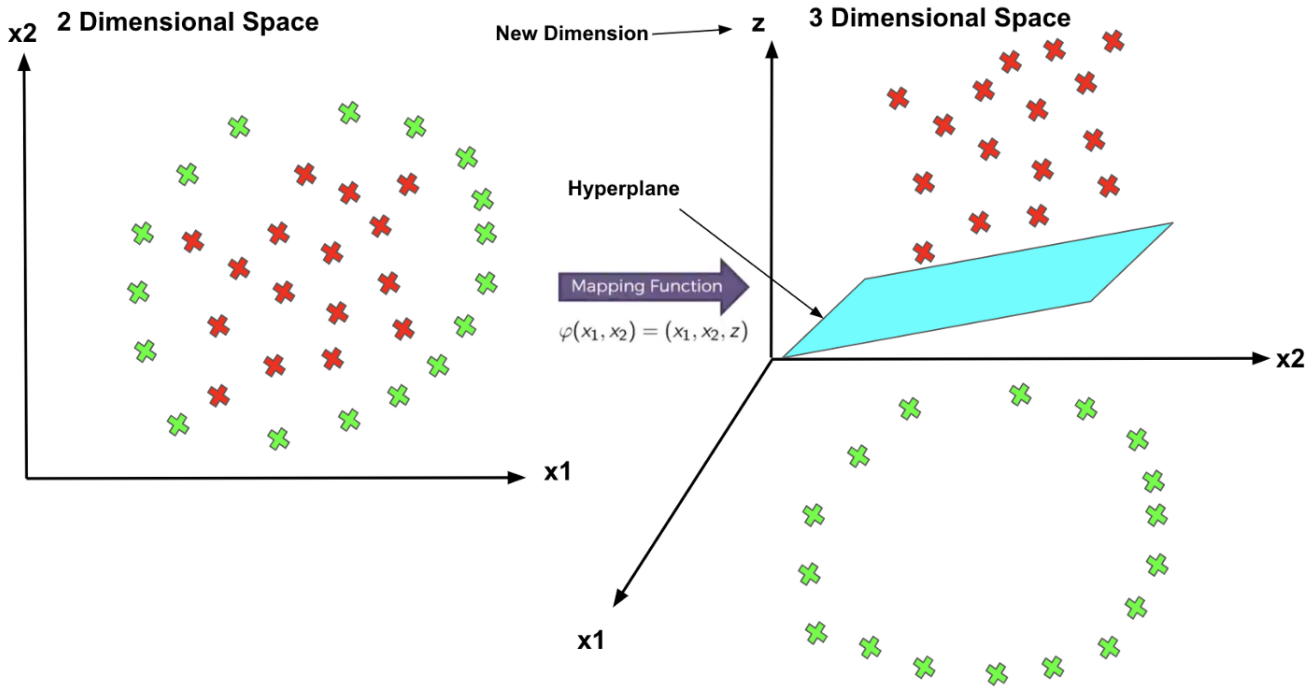


Figure 3.6: Yet another example of a kernel trick. This time the initial problem lies in a 2D domain, but can easily be projected onto a 3D one, ensuring a successful result [56]

## 3.4 Random Forest

In the base structure of every RF lie decision trees. A decision tree is similar to a flowchart: data passes through it from start to one of its exit following some inner path. A decision tree consists of nodes with a statement. Most often this statements check some feature of the incoming data in a ”if...then...else” fashion, which allows a node to branch out to other nodes and direct the data. Because of this quality, these nodes are called decision nodes. Other nodes, which do not branch out, are referred to as either terminal or leaf nodes, due to representing a final outcome of data traveling through the branches of a decision tree [57–59].

Decision trees work as ML on their own as well. I does its training by constructing a new tree starting with a single root-node with the training data in its entirety. The algorithm looks ways to split the data based on some feature of the data in order to make the new subsets to be more homogeneous, i.e. that one class would be more prevalent than the others. As soon as the best possible split is found, the node branches out in the resulted splits with corresponding nodes. The newly created nodes are called child-nodes in a relation to the currently split

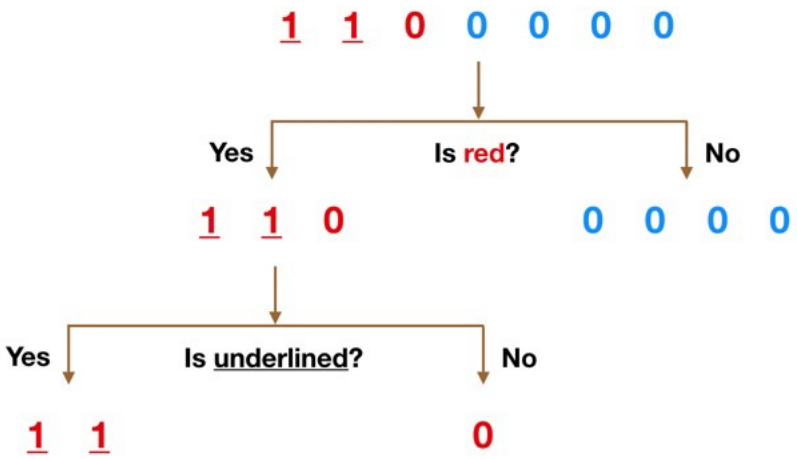


Figure 3.7: A representation of a decision tree. Here each sample represents one of a two classes, either 1 or 0. Based on the features the samples possess, the decision tree can easily classify them. [59]

parent-node. Then each subsequent child-node repeats the splitting process unless it reaches a state of absolute homogeneity, when only one class remains among the data, thus becoming a leaf node. In this fashion, the tree continues to build itself, until all branches eventually end up with leaf nodes or another specified condition is met, like maximal number of leaf nodes or maximal depth is reached and so on. The biggest concern, so far, is the split selection factor. Each time a node is reviewed, every possible split is considered and evaluated based on a prior selected function. The majority of these evaluation function are greedy; this means that they disregard any possible outcome that might appear in the future steps and only focuses on the solution which is the best in the present circumstance. We will review to members of these functions [58, 60].

The first function measures Gini impurities of nodes. As it is written in Wikipedia: "...Gini impurity is a measure of how often a randomly chosen element from the set would be incorrectly labeled if it was randomly labeled according to the distribution of labels in the subset. The Gini impurity can be computed by summing the probability $p_i$ of an item with label being chosen times the probability $\sum_{k \neq i} p_k = 1 - p_i$ of a mistake in categorizing that item." The Gini impurity can be calculated using the following expression

$$I_G(p) = \sum_{i=1}^{J} p_i \sum_{k \neq i} p_k = \sum_{i=1}^{J} p_i(1 - p_i) = \sum_{i=1}^{J} (p_i - p_i^2) = \sum_{i=1}^{J} p_i - \sum_{i=1}^{J} p_i^2 = 1 - \sum_{i=1}^{J} p_i^2$$

where $J$ is the number of classes, $i \in \{1, 2..., J\}$ and $p_i$ is a fraction of elements labeled with class $i$ in the set [61]. From this expression one may see, that lower Gini impurity implies higher homogeneity with a value of zero representing data filled with only a single class (in case of a lead node). And since the aim of a splitting a function is to strive toward more homogeneous data, from all the options it will select the splits with the lowest combined Gini impurity. [60].

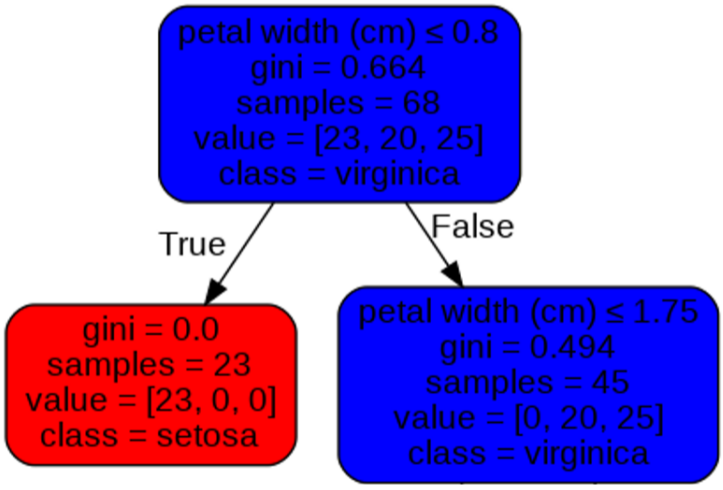


Figure 3.8: An example of split based on Gini impurity function. This particular example uses Iris database to classify flowers. As one can see the left child-node results in a absolutely homogeneous set, therefore its Gini impurity would be zero. And so, this split is selected [60]

The second principle revolves around calculating information, which reflects in entropy (units of measurement are bits) [62]. Entropy can be viewed as a measurement of disorder within a set and has a mathematical formula of

$$E(S) = -\sum_{i=1}^{J} p_i \log_2 p_i$$

where $J$ is the number of classes, $i \in \{1, 2..., J\}$ and $p_i$ is a fraction of elements labeled with class $i$ in the set [61,63]. Similar to Gini impurity, lower entropy signifies higher homogeneity, where zero is once again present in a unicategorical set. And similarly the algorithm must operate in the direction of reducing the entropy inside the dataset. This reduction represents Information Gain based of a split based on one of the features; it basically shows how much we learn about the data by looking a the feature, how good does this feature describe the data. In order to get an Information Gain of a particular split, one calculates the weighted average

of the resulting child-nodes' entropy and subtracts it from the reviewed node's entropy. In mathematical expression the Information Gain is depicted as

$$IG(S,X) = E(S) - E(S—X) = E(S) - \sum_{c \in X} P(c)E(c).$$

Logically, when all possible splits have been computed, the split with the biggest Information Gain must be chosen [58, 62–64].

Decision Tree is a valid ML algorithm, however it is prone to overfitting, i.e. perform fantastically during the training yet fail miserably working with actual data. This is when Random Forest comes in play. Essentially, RF is a group of individual Decision Trees, where each tree casts a vote in favor of a class to be predicted. Every tree analyses incoming input and provides its corresponding output, then the most resulted class is deemed the final prediction. The idea behind this logic is that errors of the minority of trees would be compensated by the successful majority. The results of RF improve with the minimising of correlation among trees and with individual increase of each tree's strength [59, 65, 66].

Several supporting methods exist for assisting RF training. One of such methods is called bootstrap aggregating (a.k.a. bagging): each individual tree is given a set of training data, where random elements are copied and replace other elements of the initial testing data. Please notice, that the size of either training data remains equal. Bagging increases variability among the trees' structure by providing different training basis. To further diversify the trees, another method implies limiting the set of features to a random subset, one tree may consider, when determining a split for its nodes. The point for this data manipulation is to stimulate different decision trees to focus their attention on different chunks of input when making a prediction [59, 65–67].

## 3.5 Scikit-learn

After selecting the ML algorithms and gathering theory about them, comes the time of implementation. Frankly, there is no need to reinvent the wheel: it is unnecessary to recreate by hand those algorithms, because multitude of libraries provide a convenient tools for ML application. Decision of which library to utilize has fallen in favor scikit-learn library for Python. It is popu-

lar, well documented and hardware universality is a top priority for the library's developers [68].

Scikit-learn provides a multitude implementations of SVM. For this paper Support Vector Machine Classifier was used, represented in the library as sklearn.svm.SVC class. The constructor for an SVC model contains several parameters for tuning:

- **C:** a regularisation parameter. It represent how much the model is willing to sacrifice largeness of margin, in favor of better classification of data points. A lower value will result in a simpler decision function, but many data points may be classified improperly and vice versa. A strictly positive float value; default is 1.0.

- **kernel:** represents a kernel function used in the algorithm. it can be selected from the list of 'linear' ( $\langle x, x' \rangle$ ), 'poly' (polynomial: $(\gamma \langle x, x' \rangle + r)^d$ ), 'rbf' (Gaussian Radial Basis Function: $\exp(-\gamma \|x - x'\|^2)$ ), 'sigmoid' ($\tanh(\gamma \langle x, x' \rangle + r)$ ). A custom kernel function can be passed into the model by either putting a standard python function as the kernel value or by setting the kernel value to 'precomputed' and providing Gram matrices to fit() and predict() methods in place of usual data. Has a value of 'rbf' by default.

- **degree:** a degree of the polynomial kernel function (poly), is represented by *d* in the formula, is of int data type, default value is 3.

- **gamma:** is a Kernel coefficient for 'rbf', 'poly' and 'sigmoid', represented by $\gamma$ in the formulas. It acts as an inverse of a kernel's radius of influence. A small value may result in possible support vector to be compared with pretty much the entire dataset and the model itself might as well perform as a linear one. It is a float value, but strings "scale" and "auto" can also be given for an actual value to be calculated by following the formulas *1 / ( number_features * training_data_variance)* and *1 / number_features* respectively. "scale" by default.

- **coef0:** a float value significant to 'poly' and 'sigmoid' kernels, is represented by *r* in their formulas. Default is 0.0.

- **shrinking:** a boolean value whether to use shrinking heuristic or not. Shrinking heuristic supposedly shortens the training time in cases, when the number of iterations reaches large numbers. True by default.

- **probability:** a boolean value which allows internal 5-fold cross-validation during a training data fit. This lets a model an ability to provide probability estimation of a class belonging for each given sample. Will slow down the training. By default is switched to False.

- **tol:** denotes the tolerance for stopping criterion; is a float value with 0.001 being the default.

- **cache_size:** a float value for kernels cash size in MB; 200 by default.

- **class_weight:** takes a dictionary, where to a key of class label corresponds a positive non-zero float weight value, which would be multiplied to C in order to get a class specific regularisation. If nothing is provided, all classes are given the weight of one. Additionally, a string 'balanced' can be provided instead, which would automatically adjust the class weight inversely proportional to the class' frequency of appearance. Default value is None.

- **verbose:** a boolean value; allows intermediary logging to appear; may not work properly with multiple threads; False by default.

- **max_iter:** a positive int value for hard capping the number of iterations within a solver. Conversely, a -1 can be given to alleviate any restrictions. -1 by default.

- **decision_function_shape:** a mostly deprecated parameter kept around for compatibility sake. Can have value either 'ovo' or 'ovr' which represent one-vs-one and one-vs-rest multi-class decision strategies. The parameter is ignored in binary classification and one-vs-one is always used for multi-class problems. Default value is 'ovr'.

- **break_ties:** a boolean value responsible for determining cases, when an input falls on the cross section of numerous classes during a multi-class prediction. If it is True, **decision_function_shape** is 'ovr' and model checks for more than two classes, then the model will resolve ties of multiple classes claiming the input to be of them own by performing a time consuming probability calculation. Else, the first class to claim the input will be returned in the output. False by default.

- **random_state:** is used to control internal pseudo random number generation for data shuffling during probability estimation; is ignored when **probability** parameter is False.

An int or a numpy.random.RandomState instance can be passed to the parameter or a None, if no randomness replication is intended.

[68–72]

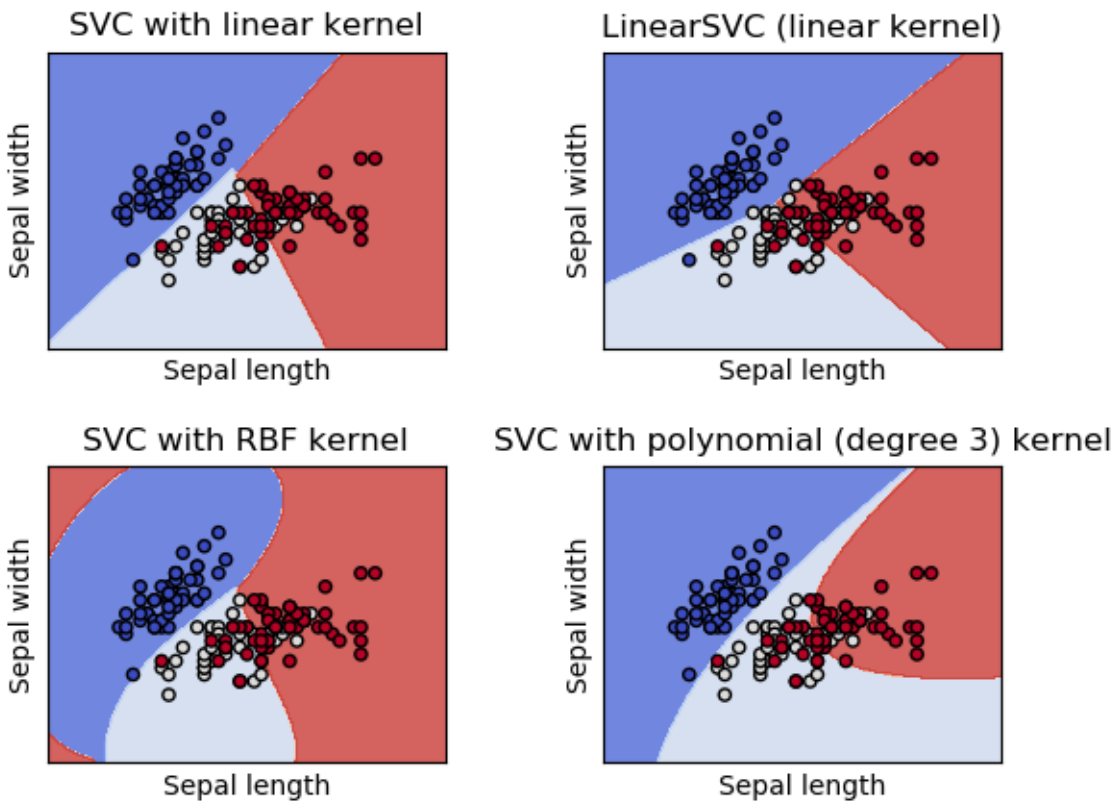


Figure 3.9: Results of differing kernel implementations of multi-class SVC on a 2D (Iris) dataset [71]

As for the RF, a Random Forest Classifier was utilized with sklearn.ensemble.RandomForestClassifier class from the library. It should be noted, that RF provided by scikit-learn do not vote for the most likely prediction, but have their probabilistic predictions averaged. In total the constructor for this class has 19 parameters available to tune:

- **n_estimators:** an int value denoting the number of trees in the forest; 100 by default.
- **criterion:** a function for split selection; can either be 'gini' (Gini impurity) or 'entropy' (Information gain); default value is 'gini'.
- **max_depth:** an int value to limit the depth of a tree. Alternatively, None can be passed to remove any restriction making the nodes split until all the branches end with leaves.

- **min_samples_split:** if the value is int, then the parameter represent the minimum number of samples of training data, which would be considered when making a split. If the value is float, then the minimum number of reviewed samples is *ceil( samples_number * **min_samples_split**)*. Default value is 2.

- **min_samples_leaf:** if the value is int, then only leafs with at least this many samples would be considered when making a split. If the value is float, then a minimal number of samples would be calculated as *ceil( samples_number * **min_samples_leaf**)*. Default value is 1.

- **min_weight_fraction_leaf:** "The minimum weighted fraction of the sum total of weights (of all the input samples) required to be at a leaf node. Samples have equal weight when sample_weight is not provided". A float value with 0.0 being the default.

- **max_features:** signifies the maximal number of features that would be considered when making a split. This number may be exceeded, if no suitable split has been found. If the value is int then the maximal number of features is the parameter itself. If the value is float then the maximal number of features is calculated following an expression *ceil( **max_features** * features_number)*. If 'sqrt' or 'auto' are given then the maximal number of features is calculated following an expression *sqrt(features_number)*. If 'log2' is given then the maximal number of features is calculated following an expression $log_2$*(features_number)*. If the value is None then there is no limit on the number of features to be considered.

- **max_leaf_nodes:** the trees are limited to having a certain maximal number of leaf nodes. Leafs are selected based on their relative reduction in impurity. An int value, but None can also be given to remove the limitation (the default value).

- **min_impurity_decrease:** nodes will be split only if the split induces a decrease of the impurity greater than or equal to this float value. The weighted impurity decrease is calculate using the following expression:

$$N_t/N * (impurity - N_{tR}/N_t * right\_impurity - N_{tL}/N_t * left\_impurity),$$

where N is the total number of samples, $N_t$ is the number of samples at the current node, $N_{tL}$ is the number of samples in the left child, and $N_{tR}$ is the number of samples in the

right child.

- **min_impurity_split:** a deprecated soon to bee removed parameter.
- **bootstrap:** a boolean value specifying whether to use bootstrap aggregating or not. If False, the entire dataset is used for tree building. True by default.
- **oob_score:** a boolean value specifying whether to use unused training data (out-of-bag samples) to estimate the generalization accuracy.
- **n_jobs:** an int value responsible for parallelisation. The computation is divided into the number of jobs equal to the value and then jobs are given the same amount of processing cores. If the value is -1, then all available cores will be used. Is None by default.
- **random_state:** is used to control internal pseudo random number generation for training data shuffling when bootstraping in on or when **max_features:** is less than features_number). An int or a numpy.random.RandomState instance can be passed to the parameter or a None, if no randomness replication is intended.
- **verbose:** a boolean value; allows intermediary logging to appear; may not work properly with multiple threads; False by default.
- **warm_start:** a boolean value, which when True allows to add more trees to the forest on a subsequent data fitting calls instead of fully retraining the model.
- **class_weight:** takes a dictionary (or a list of dictionaries in multiple output cases), where to a key of class label corresponds a positive non-zero float weight value. If nothing is provided, all classes are given the weight of one. Additionally, a string 'balanced' can be provided instead, which would automatically adjust the class weight inversely proportional to the class' frequency of appearance in the data. Another string 'balanced_subsample' can be passed, then the weights for a tree would be computed based on the tree's bootstrap data, instead of a whole dataset. Default value is None.
- **ccp_alpha:** is a non-negative float value used for Minimal Cost-Complexity Pruning, which is a method of removing nodes from a tree to reduce a tree's complexity and avoid overfitting; is 0.0 by default.
- **max_samples:** when **bootstrap** is True, this value is used to for determining the number of training samples. If the default value of None is provided, then the entire training data

is taken. If int is provided, then the number of drawn samples would be equal to the value. If a float ( must be between 0 and 1) is given, then the number of drawn samples will equal to ***max_samples*** * *samples_number*.

[72–78]

The hardest part in the ML application are the parameters of course. There is no definitive way of knowing all the correct values beforehand. Pretty much the majority of ML implement experimentation to some extent. Trial and Error approach seem to be the most widespread in contrast to other possibilities. Conveniently, scikit-learn also provides tools for looking-up the parameter. These are the GridSearchCV (GSCV) and RandomizedSearchCV (RSCV) classes from sklearn.model_selection. These classes essentially train multiple instances of specified model with a multitude of parameters to select from. Eventually, all the resulted instances are compared and the best set of parameters and the most efficient model can be accessed. The difference between them, is that GSCV checks every possible combination from the given range of parameters, whereas RSCV takes randomly but a specified number of parameter sets [79–81].

Naturally, any ML requires some training substrate for creation and testing data for performance scoring. The models initiated by RSCV and GSCV are not an exception. However, when they are used, only a single dataset should be provided, all thanks to an internal K-fold cross-validation(CV). K-fold CV is an assistance method for training and assessing ML models, which takes the most out of provided data. It breaks down the input data into K equal subsets (so in case of a 5-fold CV the data would be cut into 5 parts), and carries out 5 independent training procedures, using K-1 number of subsets for training and the remaining one for testing, thus each procedure has its own unique combination of training and testing data. Finally, all of the results are averaged to give a more general description of model's behaviour. It is quite a computationally expensive task, but nevertheless beneficial for optimising ML outputs. [79, 82]

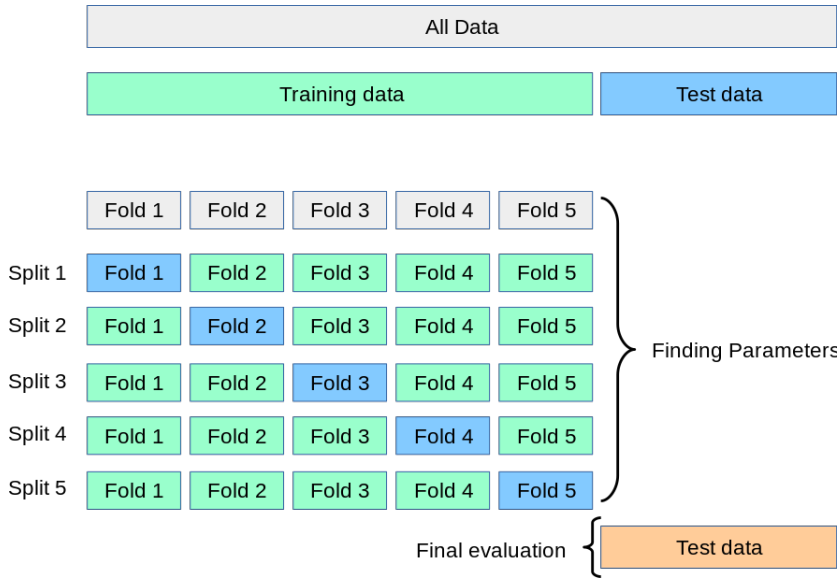


Figure 3.10: A visualisation of a 5-fold CV [82]

## 3.6 Datasets

As has been mentioned numerous times, ML needs samples to fuel its construction, therefore a good dataset is essential. The dataset should have a broad range of samples and the labels should correspond to the emotional model chosen previously in the section 2.4. For this paper several datasets have been obtained:

- **FER2013:** contains 35896 samples of grayscale images of size 48×48 pixels. The samples are labeled as either Angry, Disgust, Fear, Happy, Sad, Surprise or Neutral. Samples include pictures of people's facial expressions from different angles, along with abstract drawings [83].

- **The Japanese Female Facial Expression (JAFFE) Database:** contains 213 samples of grayscale images of size 256×256 pixels. The samples are labeled as either Anger, Disgust, Fear, Happiness, Sad, Surprise or Neutral. The samples are frontal pictures of Japanese female models posing a specific facial expression [84].

- **iCV MEFED:** contains 28718 samples of coloured images of size 5184×3456 pixels. The samples are labeled as either Anger, Contempt, Disgust, Fear, Happiness, Sadness or Surprise. The samples are all frontal pictures of a diverse group of people. Internally divided into Training, Validation and Testing subsets. Was provided by the paper's supervisor.

TODO: perhaps paste a sample for each of the mentioned databases

## 3.7 Software

The final solution for this project can be divided into steps of sub-tasks:

1. Receive a picture of a user's facial expression
2. Perform the necessary image processing
3. Classify the valence of user's emotion (Negative, Neutral or Positive)
4. Estimate the emotion's level of arousal (by either classification or regression)
5. Send a corresponding pattern to the display
6. Visualise the pattern through the display

The steps 3 and 4 seem to induce difficulty the most, therefore the practical realisation of software will commence with them. To make the steps a bit easier a few assumptions are made concerning the input data. First of all, to invoke consistency the sample must have a frontal view over a person's facial expression. Secondly, it is expected for a person's facial expression to be sincere, as not masked or suppressed, because otherwise this brings a broad variety of culture specific nuances. Since we are basing our emotional model on the concept of basic emotions, observed sincere facial expressions are universal [16].

## 3.8 ML application 1: SVM one VS one with FER2013

In order to get the initial firsthand feeling of SVM, each emotion from FER2013 was compared with each another. An SVC object was used with the parameters being default (Table 3.2) and with no class balancing. All training and scoring were performed on the Raspberry Pi. The training data made up a random 60% split of the input data, and the remaining part was used for assessment.

Table 3.3: The default parameters of an SVC model

| | |
|---|---|
| 'C' | 1 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 'auto_deprecated' |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

## 3.9 ML application 2: SVM one VS rest with FER2013

To further expand the grasp over the surface level understanding of interactions between SVM and FER2013 dataset, a number of additional binary models were trained. This time each emotion would be compared against the remaining ones grouped together. As in the previous application, only the kernel parameter was switched from default value to 'rbf'. The input data was similarly split into random subsets of 60% for training and 40% for testing. All training and scoring were performed on the Raspberry Pi.

## 3.10 ML application 3: SVM all VS all with FER2013

As a follow up, attempt at multi-class classification using SVC was carried out. The model was fed the dataset of each emotion labeled as an independent class, expecting to recieve a model able to discriminate among the 7 classes. One may recall, that SVC follows one-vs-one strategy when dealing with multi-classification tasks. Therefore, the resulting model would be an ensemble of models similar to the ones made in ML application 1. As in the previous applications, only the kernel parameter was switched from default value to 'rbf'. The input data was similarly split into random subsets of 60% for training and 40% for testing. All training and scoring were performed on the Raspberry Pi.

## 3.11 ML application 4: SVM negative VS positive + neutral with FER2013

The final attempt at training an SVM on FER2013 database. This time the plan was to segregate negative emotions from the rest, which would be happiness, surprise and neutral. In hopes of balancing the training data and not overwhelming the estimator, only 500 samples of each negative and 667 samples of the remaining emotions were fed to the SVC. The samples not used in the training were instead used for testing. This application has been carried out multiple times, each times with manually changing the 'C' and 'gamma' parameters. All training and scoring were performed on the Raspberry Pi.

## 3.12 ML application 5: RSCV for RFC negative VS neutral VS positive with JAFFE

After inspecting a handful of samples from FER2013 database, it became obvious that quite a lot of the samples do not match the first assumption made in section 3.7. With this in mind, the author has moved on to another database: JAFFE. This time an RFC was taken as an ML model, along with RSCV to help outlining the parameters. The classes were also made up to better reflect step 3 from section 3.7, negative emotions vs neutral vs positive emotions. RSCV was given a dictionary with parameters to choose from reflected in the Table 3.3; it would train 10 unrelated models simultaneously using a 3-fold CV. For the training data 70% of random samples from the JAFFE database were used, while the left out 30% were utilizing for scoring as test data. This application was carried out numerous times with identical training/test data splits. All training and scoring were performed on the Raspberry Pi.

Table 3.4: All the possible value which RSVC could use to train RFC for ML application 5, 6, 8 and 10

| Parameter | Possible Values |
|---|---|
| 'n_estimators' | {100, 200, 300...100000} |
| 'criterion' | 'gini'or 'entropy' |
| 'bootstrap' | True or False |
| 'max_depth' | None or {10, 110, 210...10000} |
| 'min_samples_split' | {2, 4, 8, 16} |
| 'min_samples_leaf' | {1, 2, 4} |

TODO: perhaps paste an unmatching sample

## 3.13 ML application 6: RSCV for RFC negative VS neutral VS positive with iCV MEFED

After proving the eligibility of RF implementation for the project, it was time to move on to another database. iCV MEFED database provides an expansive array of highly detailed samples to work with. RSCV and RFC were once again utilized with a set up identical to the one described in the previous section all but with one exception: the 15-fold CV was used instead. All training and scoring were performed on the Raspberry Pi. The entire Training and Testing subsets of the dataset was attempted to be used for training and testing the models respectively.

## 3.14 ML application 7: RSCV for SVC negative VS neutral VS positive with facial landmarks from iCV MEFED

It was suggested to look up open source solutions of tasks similar to ours. Two such projects were found, which performed data argumentation: a method of either extracting additional from already existing samples or compressing the data into bare essential [85, 86]. The idea was, that instead of providing all the pixel values of the sample to ML, only a few in number values should be given. The values represent the coordinates of points on the image, which outline the shape of a person's face/facial expression. Both projects implemented Facial Landmarks detector for receiving x and y coordinates of 68 point. The points represent the position of brows, nose, shape of eyes and mouth [87]. Not only does this helps ML to focus on the important features, but also avoids overwhelming computer's memory.

Analysing an image on the basis of Facial Landmarks is a time consuming computation. A separate pre-processing program was created in order to create child-datasets from Training and Testing subsets of iCV MEFED database. The steps of the program as followed:

1. Read a sample from the database as grayscale.

2. Detect faces using frontal face detector from dlib [88, 89].

3. Discard the sample and move on to the next one, if step 2 resulted in several faces detected (fortunately iCV MEFED database is expansive enough to neglect this loses).

4. Use Facial Landmarks detector to acquire 68 pairs of coordinates.

5. Flatten the results of step 4 into a single vector and write it to .txt file along with sample's emotion label, so a human can easily check whether any mistakes occurred.

6. Repeat steps 1 through 5 for each sample in the dataset.

Since the projects used SVC, it was decided to give SVC yet another chance. This time SVC was accompanied by an RSCV, however used slightly unconventionally. The training of a single model took several hours, training several models at once could take severely longer with a chance of stumbling across a memory error. Therefore the number of simultaneous models trained was reduced to only a single one. This made RSVC into merely a random parameter selector. CV was also reduced to 2-fold. The possible parameters are show in Table 3.5. Sizes of training sets for each class were kept approximately equal. The computations were performed on both the Raspberry Pi and author's PC.

Table 3.5: All the possible value which RSVC could use to train SVC for ML application 7 and 9

| Parameter | Possible Values |
|---|---|
| 'gamma' | {100, 200, 300...100000} |
| 'kernel' | 'rbf', 'poly' or 'sigmoid' |
| 'C' | {100, 200, 300...100000} |

## 3.15 ML application 8: RSCV for RFC negative VS neutral VS positive with facial landmarks from iCV MEFED

This procedure was nearly identical to the previous one, with differences being the model used, which is RFC, along with the dictionary of parameters, which is the same as in ML application 5 (Table 3.5).

## 3.16 ML application 9: RSCV for SVC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

A follow up application to the $7^{th}$ application. It was supposed, that perhaps SVC is unable to have any sufficiently good results because of the large resolution of pictures inside the database ($5184 \times 3456$ pixels). To test this idea, a procedure was carried out similar to the one described previously about Facial Landmarks extraction. This time the samples were resized to half of their original dimensions before passing them to detectors. Afterwards, RSCV with SCV have been implement as described in ML application 7, although all computations were performed on Raspberry Pi.

## 3.17 ML application 10: RSCV for RFC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

This procedure was nearly identical to the previous one, with differences being the model used, which is RFC, along with the dictionary of parameters, which is the same as in ML application 5 (Table 3.5).

## 3.18 Histogram of oriented gradients, data size estimation

Along with Facial Landmarks detector, the project of Amine Horseman also implements a feature descriptor method called Histogram of oriented gradients (HOG) [86]. This method is aimed at extracting the information about the shapes of objects in a picture, discarding everything else like colour, intensity etc. This, again, helps ML algorithms with locating crucial features and reduces data size.

HOG's main working aspect is not dealing with a whole image, but instead dividing the image into a multitude of little sections called cells. Specifically in this project HOG operates

over rectangular sections $N \times M$ pixels, where *N* and *M* specifies the programmer. Since HOG is a histogram after all, and as a histogram it provides a view over the frequency distribution of a set of continuous data, which is orientation in degrees in our case. To achieve this distribution of gradients is calculated. It can easily be achieved by using, for example, Sobel kernels to get the initial gradients in the x and y direction. Then using the Pythagoras theorem the overall magnitude of a gradient can be calculated, simply because gradients of x and y direction form a right-angled triangle. Using the same triangle concept the orientation of the gradient can also be received via arctangent. With the magnitude and orientation values we can move on to

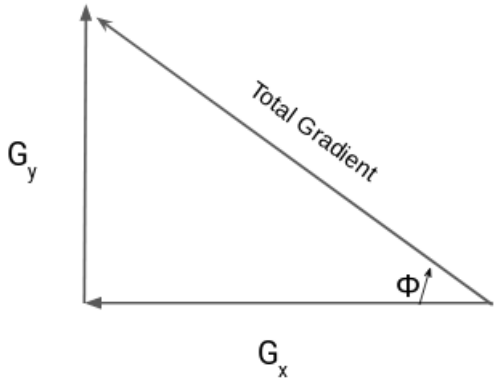


Figure 3.11: Representation of gradients in x and y directions as a right-angled triangle [91]

filling out the plot. The x-dimension will represent the angle of orientation, and y-dimension will show the the total magnitudes corresponding to the angle. Placing each magnitude value on a continues axis would be tedious and impractical. Thus the whole range of angle values is discreetly sectioned into bins of equal size. It should be noted, that in the case of gradients the angle range goes from 0 to 180 degrees, because the gradient of one direction and a gradient of the polar opposite direction (180°apart) are virtually the same, and this does not impact quality of performance. And so, the gradient magnitudes for every pixel are inserted into the designated bins. If an orientation does not strictly fall in the value of any bin, then its corresponding magnitude will be divided among the two closest bins with respect to their distances from the orientation as depicted in Figure 3.12.

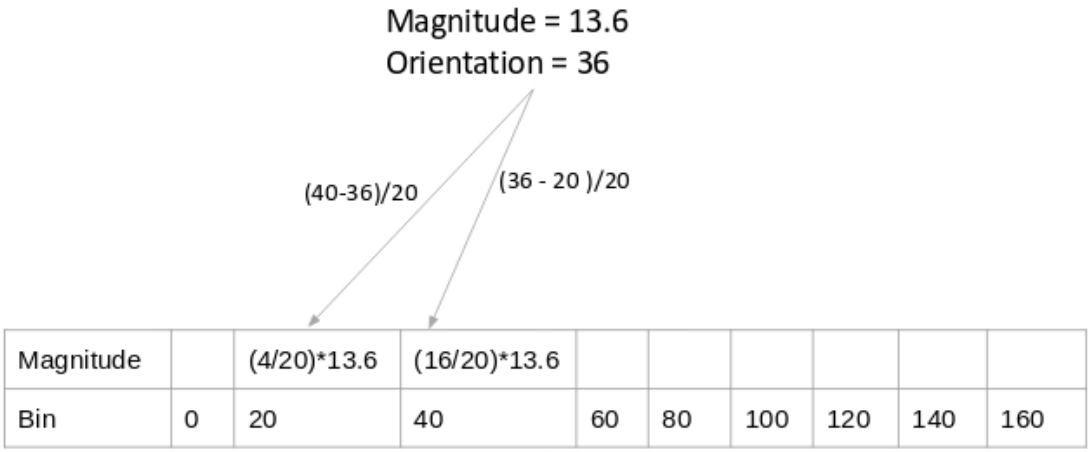


Figure 3.12: An orientation of a gradient does not fall strictly into the bin, therefore its magnitude is being shared by the two closest bins, based on how close they are. [91]

When all the computations for each cell is made, nearby cells can be grouped into (rectangular) blocks to perform normalisation among the cells inside each block. Normalisation can help avoid negative effects of extreme lighting. Finally, the result can be flattened down to a single vector for easier manipulation and possibility to be fed as an input to a ML algorithm [90, 91].

Knowing the inner workings of HOG, programming it can impose a challenge. But yet again, we are covered by external python library, which provides a ready-to-use implementation. Now it is a scikit-learn's sister library: scikit-image - specialises in functions assisting with image processing [92]. As far as HOG calculation is concerned, scikit-image offers a method called *hog()* found in the class *hog(skimage.feature).* [93] It has several parameters, out of which the most important are

- **orientations:** an int value; represents the number of orientation bins, which divide the angle axis.
- **pixels_per_cell:** a (int, int) 2-d tuple, which shows the dimensions of a cell in pixels.
- **cells_per_block:** a (int, int) 2-d tuple, which shows the size of blocks in cells.

[93] And as usual in the field of machine learning, these parameters are not easy to obtain and must be acquired by means of numerous experiments. The obvious part is the fact, that the more pixels are in the cell, the smaller resolution becomes. Since the data size is incredibly important within the constraints of a weaker hardware, it can be one of the criteria of selecting

the parameters. In order to estimate potential size of each sample numerous HOG calculations with varying values of **pixels_per_cell** and **cells_per_block** parameters were preformed over a single random image from iCV MEFED database. In total 81 procedures were carried out and their vector results saved into .npy files. This way we can estimate the effect the parameters have on the size of a single sample. As for **orientations** parameter, it is generally recommended to use a value of 9.

## 3.19 ML application 11: SVC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

It was time to implement HOG into ML. As an ML algorithm SVC was taken with the default parameters (Table 3.4). HOG vectors for training were computed from iCV MEFED Training dataset using the parameters: **orientations** = 9, **pixels_per_cell** = 72, **cells_per_block** = 1. Similarly, HOG vectors for testing were calculated. Each iteration of ML training had a limit on a maximum number of samples representing each class. All calculations were performed on the author's PC.

## 3.20 ML application 12: RFC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

Identical to the previous application, albeit RFC was used instead of SVC.

## 3.21 ML application 13: RFC negative VS neutral VS positive with HOG vectors of smallest resolution from iCV MEFED

Perhaps, the problem with the previous application was due to the lost features, which happened during HOG function execution with a big value of **pixels_per_cell** parameter. This implementation would use HOG vectors, which back up the best resolution achieved with **pixels_per_cell** value of 8. Other parameters: **cells_per_block** = 1, **orientations** = 9. In order to not lose time much time in between the training sessions, the vectors were pre-calculated and then were

stored using numpy library's memmap. This made a binary file of a numpy array, which then could be accessed by code in slices, and (hopefully) not overwhelm the system's memory [94].

# 4 Results

## 4.1 Results of ML application 1: SVM one VS one with FER2013

In total 21 SVC models were trained for each possible pair among the 6 emotions + neutral. The models are named after the pair of emotions they were trained to discern in a format of: $emotion_1$ VS $emotion_2$. The training time for each model was around 2 to 4 hours.

The list of finished models is following:
anger VS neutral, anger VS disgust, anger VS fear, anger VS happiness, anger VS sadness, anger VS surprise, neutral VS disgust, neutral VS fear, neutral VS happiness, neutral VS sadness, neutral VS surprise, disgust VS fear, disgust VS happiness, disgust VS sadness, disgust VS surprise, fear VS happiness, fear VS sadness, fear VS surprise, happiness VS sadness, happiness VS surprise and sadness VS surprise.

In order to assess and compare the models' performance each model was given a set of test data, which the model tried to predict. The predicted class was then checked with the sample's actual class. Using *.score()* method of a model a float number has been returned, which shows the fraction of successful predictions in relation to the number of total predictions made. In addition, a method *sklearn.metrics.confusion_matrix()* was used to visualise the distribution between the successful and unsuccessful predictions in a matrix.

The results for each model is given in a format of:

---

*model's name* ($emotion_1$ VS $emotion_2$)
scoring: *returned value of .score() method*

Table 4.1: An example of a confusion matrix for the model $emotion_1$ VS $emotion_2$

| | predicted $emotion_1$ | predicted $emotion_2$ |
|---|---|---|
| true $emotion_1$ | number of $emotion_1$ samples predicted as $emotion_1$ | number of $emotion_1$ samples predicted as $emotion_2$ |
| true $emotion_2$ | number of $emotion_2$ samples predicted as $emotion_1$ | number of $emotion_2$ samples predicted as $emotion_2$ |

---

anger VS neutral

scoring: 0.5711723828737951

Table 4.2: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and neutral data

| | predicted anger | predicted neutral |
|---|---|---|
| true anger | 113 | 1910 |
| true neutral | 3 | 2435 |

---

anger VS disgust

scoring: 0.9172727272727272

Table 4.3: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and disgust data

| | predicted anger | predicted disgust |
|---|---|---|
| true anger | 1981 | 0 |
| true disgust | 182 | 37 |

---

anger VS fear

scoring: 0.5196029776674937

Table 4.4: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and fear data

| | predicted anger | predicted fear |
|---|---|---|
| true anger | 114 | 1933 |
| true fear | 3 | 1980 |

---

anger VS happiness

scoring: 0.6634391249775865

Table 4.5: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and happiness data

| | predicted anger | happiness |
|---|---|---|
| true anger | 109 | 1876 |
| true happiness | 1 | 3591 |

---

anger VS sadness

scoring: 0.5677697189483227

Table 4.6: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and sadness data

| | predicted anger | predicted sadness |
|---|---|---|
| true anger | 115 | 1904 |
| true sadness | 3 | 2390 |

---

anger VS surprise

scoring: 0.6510329424902289

Table 4.7: Confusion matrix resulted from a binary SVC model trained on FER2013 anger and surprise data

| | predicted anger | predicted surprise |
|---|---|---|
| true anger | 2011 | 1 |
| true surprise | 1249 | 321 |

---

neutral VS disgust

scoring: 0.9347664936990363

Table 4.8: Confusion matrix resulted from a binary SVC model trained on FER2013 neutral and disgust data

| | predicted neutral | predicted disgust |
|---|---|---|
| true neutral | 2479 | 0 |
| true disgust | 176 | 51 |

---

neutral VS fear

scoring: 0.592756183745583

Table 4.9: Confusion matrix resulted from a binary SVC model trained on FER2013 neutral and fear data

| | predicted neutral | predicted fear |
|---|---|---|
| true neutral | 2533 | 2 |
| true fear | 1842 | 151 |

---

neutral VS happiness

scoring: 0.6064197530864197

Table 4.10: Confusion matrix resulted from a binary SVC model trained on FER2013 neutral and happiness data

| | predicted neutral | predicted happiness |
|---|---|---|
| true neutral | 76 | 2391 |
| true happiness | 0 | 3608 |

---

neutral VS sadness

scoring: 0.525254582484725

Table 4.11: Confusion matrix resulted from a binary SVC model trained on FER2013 neutral and sadness data

| | predicted neutral | predicted sadness |
|---|---|---|
| true neutral | 2500 | 1 |
| true sadness | 2330 | 79 |

---

neutral VS surprise

scoring: 0.7026960784313725

Table 4.12: Confusion matrix resulted from a binary SVC model trained on FER2013 neutral and surprise data

| | predicted neutral | predicted surprise |
|---|---|---|
| true neutral | 2519 | 0 |
| true surprise | 1213 | 348 |

---

disgust VS fear

scoring: 0.9069664902998237

Table 4.13: Confusion matrix resulted from a binary SVC model trained on FER2013 disgust and fear data

| | predicted disgust | predicted fear |
|---|---|---|
| true disgust | 38 | 210 |
| true fear | 1 | 2019 |

---

disgust VS happiness

scoring: 0.946526867627785

Table 4.14: Confusion matrix resulted from a binary SVC model trained on FER2013 disgust and happiness data

| | predicted disgust | predicted happiness |
|---|---|---|
| true disgust | 40 | 204 |
| true happiness | 0 | 3571 |

---

disgust VS sadness

scoring: 0.919622641509434

Table 4.15: Confusion matrix resulted from a binary SVC model trained on FER2013 disgust and sadness data

| | predicted disgust | predicted sadness |
|---|---|---|
| true disgust | 42 | 213 |
| true sadness | 0 | 2395 |

---

disgust VS surprise

scoring: 0.8978021978021978

Table 4.16: Confusion matrix resulted from a binary SVC model trained on FER2013 disgust and surprise data

| | predicted disgust | predicted surprise |
|---|---|---|
| true disgust | 43 | 186 |
| true surprise | 0 | 1591 |

---

fear VS happiness

scoring: 0.6614103472714387

Table 4.17: Confusion matrix resulted from a binary SVC model trained on FER2013 fear and happiness data

| | predicted fear | predicted happiness |
|---|---|---|
| true fear | 152 | 1911 |
| true happiness | 0 | 3581 |

---

fear VS sadness

scoring: 0.5689732142857142

Table 4.18: Confusion matrix resulted from a binary SVC model trained on FER2013 fear and sadness data

| | predicted fear | predicted sadness |
|---|---|---|
| true fear | 166 | 1931 |
| true sadness | 0 | 2383 |

---

fear VS surprise

scoring: 0.6673972602739726

Table 4.19: Confusion matrix resulted from a binary SVC model trained on FER2013 fear and surprise data

| | predicted fear | predicted surprise |
|---|---|---|
| true fear | 2086 | 6 |
| true surprise | 1208 | 350 |

---

happiness VS sadness

scoring: 0.6089264974282396

Table 4.20: Confusion matrix resulted from a binary SVC model trained on FER2013 happiness and sadness data

| | predicted happiness | predicted sadness |
|---|---|---|
| true happiness | 3600 | 0 |
| true sadness | 2357 | 70 |

———————————————————-

happiness VS surprise

scoring: 0.7687127188762748

Table 4.21: Confusion matrix resulted from a binary SVC model trained on FER2013 happiness and surprise data

| | predicted happiness | predicted surprise |
|---|---|---|
| true happiness | 3610 | 1 |
| true surprise | 1201 | 385 |

———————————————————-

sadness VS surprise

scoring: 0.703125

Table 4.22: Confusion matrix resulted from a binary SVC model trained on FER2013 sadness and surprise data

| | predicted sadness | predicted surprise |
|---|---|---|
| true sadness | 2469 | 0 |
| true surprise | 1197 | 366 |

## 4.2 Results of ML application 2: SVM one VS rest with FER2013

In total 7 SVC models were trained for each of the 6 emotions + neutral. The models are named after the emotion the model tried to discern from the others: $emotion$ VS rest (*list of all other emotions*). The training time for each model was around 6 hours.

The list of finished models is following:

anger VS rest (neutral + disgust + fear + happiness + sadness + surprise)

neutral VS rest (anger + disgust + fear + happiness + sadness + surprise)

disgust VS rest (anger + neutral + fear + happiness + sadness + surprise)

fear VS rest (anger + disgust + neutral + happiness + sadness + surprise)

happiness VS rest (anger + disgust + neutral + fear + sadness + surprise)

sadness VS rest (anger + disgust + neutral + fear + happiness + surprise)

surprise VS rest (anger + disgust + neutral + fear + happiness + sadness)

In order to assess and compare the models' performance each model was given a set of test data,

which the model tried to predict. Once again the methods *.score()* and *sklearn.metrics.confusion_matrix()* were utilized for performance quality depiction.

The results for each model is given in a format described in section 4.1:

---

anger VS rest (neutral + disgust + fear + happiness + sadness + surprise)

scoring: 0.868617206548241

Table 4.23: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish anger from the rest of emotions, which are neutral, disgust, fear, happiness, sadness and surprise

| | predicted anger | predicted rest |
|---|---|---|
| true anger | 92 | 1880 |
| true rest | 6 | 12377 |

---

neutral VS rest (anger + disgust + fear + happiness + sadness + surprise)

scoring: 0.8329501915708812

Table 4.24: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish neutral from the rest of emotions, which are anger, disgust, fear, happiness, sadness and surprise

| | predicted neutral | predicted rest |
|---|---|---|
| true neutral | 76 | 2398 |
| true rest | 0 | 11881 |

---

disgust VS rest (anger + neutral + fear + happiness + sadness + surprise)

scoring: 0.9877394636015325

Table 4.25: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish disgust from the rest of emotions, which are anger, neutral, fear, happiness, sadness and surprise

| | predicted disgust | predicted rest |
|---|---|---|
| true disgust | 47 | 175 |
| true rest | 1 | 14132 |

---

fear VS rest (anger + disgust + neutral + happiness + sadness + surprise)

scoring: 0.8665273423894113

Table 4.26: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish fear from the rest of emotions, which are anger, neutral, disgust, happiness, sadness and surprise

| | predicted fear | predicted rest |
|---|---|---|
| true fear | 133 | 1910 |
| true rest | 6 | 12306 |

---

happiness VS rest (anger + disgust + neutral + fear + sadness + surprise)

scoring: 0.7538836642284918

Table 4.27: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish happiness from the rest of emotions, which are anger, neutral, disgust, fear, sadness and surprise

| | predicted happiness | predicted rest |
|---|---|---|
| true happiness | 94 | 3532 |
| true rest | 1 | 10728 |

---

sadness VS rest (anger + disgust + neutral + fear + happiness + surprise)

scoring: 0.8352490421455939

Table 4.28: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish sadness from the rest of emotions, which are anger, neutral, disgust, fear, happiness and surprise

| | predicted sadness | predicted rest |
|---|---|---|
| true sadness | 69 | 2358 |
| true rest | 7 | 11921 |

---

surprise VS rest (anger + disgust + neutral + fear + happiness + sadness)

scoring: 0.9116684082201324

Table 4.29: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish surprise from the rest of emotions, which are anger, neutral, disgust, fear, happiness and sadness

| | predicted surprise | predicted rest |
|---|---|---|
| true surprise | 353 | 1260 |
| true rest | 8 | 12734 |

## 4.3 Result of ML application 3: SVM all VS all with FER2013

A single model was trained for this application. The method *sklearn.metrics.confusion_matrix()* along with the testing data were used to assess the model's accuracy. The resulted confusion matrix for this model is provided below in the Table 4.30.

Table 4.30: Confusion matrix resulted from a multi-classification SVC model trained on FER2013 data in order to distinguish every emotion from each other

| | predicted anger | predicted neutral | predicted disgust | predicted fear | predicted happiness | predicted sadness | predicted surprise |
|---|---|---|---|---|---|---|---|
| true anger | 106 | 0 | 0 | 1 | 1864 | 1 | 0 |
| true neutral | 1 | 71 | 0 | 1 | 2433 | 1 | 0 |
| true disgust | 0 | 0 | 34 | 0 | 173 | 0 | 0 |
| true fear | 4 | 2 | 0 | 134 | 1933 | 6 | 4 |
| true happiness | 0 | 1 | 0 | 0 | 3512 | 0 | 0 |
| true sadness | 6 | 2 | 0 | 3 | 2367 | 73 | 0 |
| true surprise | 1 | 0 | 0 | 5 | 1238 | 0 | 378 |

## 4.4 Result of ML application 4: SVM negative VS positive + neutral with FER2013

Several models were trained for this application. Despite tweaking the 'C' and 'gamma' parameters each iteration of this application, all outcomes seemed the same. Due to a mistake in the code, each result was over-writing the previous one. Thus only one assessment of the latest model of 'C' parameter 1000 and 'gamma' parameter 100000 is provided, which includes a *.score()* method's returned float number and a *sklearn.metrics.confusion_matrix()* method's returned confusion matrix.

scoring: 0.5500972612467966

Table 4.31: Confusion matrix resulted from a binary SVC model trained on FER2013 data in order to distinguish negative emotions (anger, disgust, fear, sadness) from the rest, which are neutral, happiness and surprise. The 'gamma' parameter is 100000 and 'C' is 1000

| | predicted negative | predicted rest |
|---|---|---|
| true negative | 134 | 14564 |
| true rest | 7 | 17682 |

## 4.5 Results of ML application 5: RSCV for RFC negative VS neutral VS positive with JAFFE

In total 6 models were trained using RSCV. All these models are named in a format of: TREE search *i*, where *i* stands for the iteration of this application.

The list of finished models is following:
TREE search -2, TREE search -1, TREE search 0, TREE search 1, TREE search 2, TREE search 3.
Notice, that numeration of the RFCs begins with a negative number; all RFCs marked with a negative number have their models lost.

Just like in the previous applications, methods *sklearn.metrics.confusion_matrix()* and *.score()* were used to assess models' performance. However this time another score is given in addition, which represents the successful instances of predictions during CV inside of the RSCV training. It can be accessed from an RSCV object's attribute *best_index_* after training.

The results of assessments for each model along with a table with their parameters are provided below in a format of:

———————————————————-

*model's name* (TREE search *i*)

Table 4.32: An example of a table containing parameters of the given model

| $parameter_1$ | value of $parameter_1$ |
|---|---|
| $parameter_2$ | value of $parameter_2$ |
| ... | ... |
| $parameter_n$ | value of $parameter_n$ |

search score: *RSCV's best_index_ attribute after training*

test score: *returned value of .score() method*

Table 4.33: An example of a confusion matrix for the given model

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | number of negative samples predicted as negative | number of negative samples predicted as neutral | number of negative samples predicted as positive |
| true neutral | number of neutral samples predicted as negative | number of neutral samples predicted as neutral | number of neutral samples predicted as positive |
| true positive | number of positive samples predicted as negative | number of positive samples predicted as neutral | number of positive samples predicted as positive |

———————————————————-

TREE search -2

Table 4.34: The parameters of RFC model: TREE search -2

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 2835 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 75781 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7578125

test score: 0.7818181818181819

Table 4.35: Confusion matrix from test data of RFC model: TREE search -2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 4 | 4 | 2 |
| true positive | 3 | 0 | 12 |

———————————————————-

TREE search -1

Table 4.36: The parameters of RFC model: TREE search -1

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 413 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 62663 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7734375

test score: 0.8181818181818182

Table 4.37: Confusion matrix from test data of RFC model: TREE search -1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 3 | 5 | 2 |
| true positive | 2 | 0 | 13 |

———————————————————-

TREE search 0

Table 4.38: The parameters of RFC model: TREE search 0

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 2633 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 81836 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7734375

test score: 0.8181818181818182

Table 4.39: Confusion matrix from test data of RFC model: TREE search 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 3 | 5 | 2 |
| true positive | 2 | 0 | 13 |

———————————————————-

TREE search 1

Table 4.40: The parameters of RFC model: TREE search 1

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 9899 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 2 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 55600 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7734375

test score: 0.8363636363636363

Table 4.41: Confusion matrix from test data of RFC model: TREE search 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 3 | 5 | 2 |
| true positive | 1 | 0 | 14 |

———————————————————-

TREE search 2

Table 4.42: The parameters of RFC model: TREE search 2

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 8789 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 35418 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.765625

test score: 0.8181818181818182

Table 4.43: Confusion matrix from test data of RFC model: TREE search 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 4 | 5 | 1 |
| true positive | 2 | 0 | 13 |

———————————————————-

TREE search 3

Table 4.44: The parameters of RFC model: TREE search 3

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 6468 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 16 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 67709 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.765625

test score: 0.7818181818181819

Table 4.45: Confusion matrix from test data of RFC model: TREE search 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 27 | 0 | 3 |
| true neutral | 3 | 4 | 3 |
| true positive | 3 | 0 | 12 |

## 4.6 Results of ML application 6: RSCV for RFC negative VS neutral VS positive with iCV MEFED

In total 3 models were trained using RSCV. All these models are named in a format of: iCV tree Search *i*, where *i* stands for the iteration of this application.

The list of finished models is following:
iCV tree Search 0, iCV tree Search 1, iCV tree Search 2.

The assessment and format of result representation is exactly identical as the one described in

section 4.5.

———————————————————-

iCV tree Search 0

Table 4.46: The parameters of RFC model: iCV tree Search 0

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 5560 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 47527 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.700857912204897

test score: 0.7000521648408973

Table 4.47: Confusion matrix from test data of RFC model: iCV tree Search 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4026 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1610 | 0 | 0 |

———————————————————-

iCV tree Search 1

Table 4.48: The parameters of RFC model: iCV tree Search 1

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 4349 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 72754 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.700857912204897

test score: 0.7000521648408973

Table 4.49: Confusion matrix from test data of RFC model: iCV tree Search 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4026 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1610 | 0 | 0 |

———————————————————-

iCV tree Search 2

Table 4.50: The parameters of RFC model: iCV tree Search 2

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 3844 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 47527 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5884605228412991

test score: 0.27995131281516256

Table 4.51: Confusion matrix from test data of RFC model: iCV tree Search 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4026 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1610 |

## 4.7 Results of ML application 7: RSCV for SVC negative VS neutral VS positive with facial landmarks from iCV MEFED

In total 11 models were trained using RSCV. All these models are named in a format of: Landmark SVC search *p i*,

where *P* denotes the hardware upon which the training of the model was carried out (can be either Pi meaning the training was performed on Raspberry Pi 4 model B or PC meaning the training was instead performed on the author's laptop),

and *i* stands for the iteration of this application.

The list of finished models is following:
Landmark SVC search Pi 0, Landmark SVC search Pi 1, Landmark SVC search Pi 2, Landmark SVC search Pi 3, Landmark SVC search Pi 4, Landmark SVC search PC 0, Landmark SVC search PC 1, Landmark SVC search PC 2, Landmark SVC search PC 3, Landmark SVC search PC 4 and Landmark SVC search PC 5.

The assessment and format of result representation is exactly identical as the one described in section 4.5.

---

Landmark SVC search Pi 0

Table 4.52: The parameters of SVC model: Landmark SVC search Pi 0

| | |
|---|---|
| 'C' | 85300 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 90500 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.7005277044854882
test score: 0.7014767932489452

Table 4.53: Confusion matrix from test data of SVC model: Landmark SVC search Pi 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

---

Landmark SVC search Pi 1

Table 4.54: The parameters of SVC model: Landmark SVC search Pi 1

| | |
|---|---|
| 'C' | 32800 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 32200 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.7005277044854882

test score: 0.7014767932489452

Table 4.55: Confusion matrix from test data of SVC model: Landmark SVC search Pi 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search Pi 2

Table 4.56: The parameters of SVC model: Landmark SVC search Pi 2

| | |
|---|---|
| 'C' | 94400 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 38200 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.7005277044854882

test score: 0.7014767932489452

Table 4.57: Confusion matrix from test data of SVC model: Landmark SVC search Pi 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search Pi 3

Table 4.58: The parameters of SVC model: Landmark SVC search Pi 3

| | |
|---|---|
| 'C' | 53000 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 19400 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.7005277044854882

test score: 0.7014767932489452

Table 4.59: Confusion matrix from test data of SVC model: Landmark SVC search Pi 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search Pi 4

Table 4.60: The parameters of SVC model: Landmark SVC search Pi 4

| | |
|---|---|
| 'C' | 30200 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 84700 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.7005277044854882

test score: 0.7014767932489452

Table 4.61: Confusion matrix from test data of SVC model: Landmark SVC search Pi 4

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 0

Table 4.62: The parameters of SVC model: Landmark SVC search PC 0

| | |
|---|---|
| 'C' | 82800 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 28300 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4719207173194903

test score: 0.7014767932489452

Table 4.63: Confusion matrix from test data of SVC model: Landmark SVC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 1

Table 4.64: The parameters of SVC model: Landmark SVC search PC 1

| | |
|---|---|
| 'C' | 68000 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 82600 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4719207173194903

test score: 0.7014767932489452

Table 4.65: Confusion matrix from test data of SVC model: Landmark SVC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 2

Table 4.66: The parameters of SVC model: Landmark SVC search PC 2

| | |
|---|---|
| 'C' | 8200 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 87600 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4719207173194903

test score: 0.7014767932489452

Table 4.67: Confusion matrix from test data of SVC model: Landmark SVC search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 3

Table 4.68: The parameters of SVC model: Landmark SVC search PC 3

| | |
|---|---|
| 'C' | 31900 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 23700 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4719207173194903

test score: 0.7014767932489452

Table 4.69: Confusion matrix from test data of SVC model: Landmark SVC search PC 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 4

Table 4.70: The parameters of SVC model: Landmark SVC search PC 4

| | |
|---|---|
| 'C' | 16500 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 87300 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4719207173194903

test score: 0.7014767932489452

Table 4.71: Confusion matrix from test data of SVC model: Landmark SVC search PC 4

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

———————————————————-

Landmark SVC search PC 5

Table 4.72: The parameters of SVC model: Landmark SVC search PC 5

| | |
|---|---|
| 'C' | 69700 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 41200 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

rch score: 0.3918918918918919

test score: 0.7014767932489452

Table 4.73: Confusion matrix from test data of SVC model: Landmark SVC search PC 5

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3990 | 0 | 0 |
| true neutral | 114 | 0 | 0 |
| true positive | 1584 | 0 | 0 |

## 4.8 Results of ML application 8: RSCV for RFC negative VS neutral VS positive with facial landmarks from iCV MEFED

In total 10 models were trained using RSCV. All these models are named in a format of: Landmark RFC Search *p i*,
where *P* denotes the hardware upon which the training of the model was carried out (can be either Pi meaning the training was performed on Raspberry Pi 4 model B or PC meaning the training was instead performed on the author's laptop),
and *i* stands for the iteration of this application.

The list of finished models is following:

Landmark RFC search Pi 0, Landmark RFC search Pi 1, Landmark RFC search Pi 2, Landmark RFC search Pi 3, Landmark RFC search Pi 4, Landmark RFC search Pi 5, Landmark RFC search Pi 6, Landmark RFC search PC 0, Landmark RFC search PC 1 and Landmark RFC search PC 2.

The assessment and format of result representation is exactly identical as the one described in section 4.5.

---

Landmarks RFC Search Pi 0

Table 4.74: The parameters of RFC model: Landmark RFC Search Pi 0

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 2026 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 64681 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7312425565700675

test score: 0.3785161744022503

Table 4.75: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 710 | 0 | 3280 |
| true neutral | 18 | 0 | 96 |
| true positive | 139 | 2 | 1443 |

---

Landmarks RFC Search Pi 1

Table 4.76: The parameters of RFC model: Landmark RFC Search Pi 1

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 4193 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 31381 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7283313484186846

test score: 0.3790436005625879

Table 4.77: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 716 | 2 | 3272 |
| true neutral | 20 | 0 | 94 |
| true positive | 141 | 3 | 1440 |

———————————————————-

Landmarks RFC Search Pi 2

Table 4.78: The parameters of RFC model: Landmark RFC Search Pi 2

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 2983 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 16 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 10190 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.7320365224295355

test score: 0.37746132208157523

Table 4.79: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 700 | 0 | 3290 |
| true neutral | 18 | 0 | 96 |
| true positive | 135 | 2 | 1447 |

———————————————————-

Landmarks RFC Search Pi 3

Table 4.80: The parameters of RFC model: Landmark RFC Search Pi 3

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 2026, |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 21290 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.679984922728986

test score: 0.6680731364275668

Table 4.81: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3244 | 0 | 746 |
| true neutral | 106 | 0 | 8 |
| true positive | 1028 | 0 | 556 |

———————————————————-

Landmarks RFC Search Pi 4

Table 4.82: The parameters of RFC model: Landmark RFC Search Pi 4

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 4899 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 16 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 19272 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.6836285965573564

test score: 0.6689521800281294

Table 4.83: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 4

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3254 | 0 | 736 |
| true neutral | 108 | 0 | 6 |
| true positive | 1033 | 0 | 551 |

———————————————————-

Landmarks RFC Search Pi 5

Table 4.84: The parameters of RFC model: Landmark RFC Search Pi 5

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 4092 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 43490 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.6855132554340998

test score: 0.6726441631504922

Table 4.85: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 5

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3325 | 0 | 665 |
| true neutral | 109 | 0 | 5 |
| true positive | 1083 | 0 | 501 |

———————————————————-

Landmarks RFC Search Pi 6

Table 4.86: The parameters of RFC model: Landmark RFC Search Pi 6

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 1068 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 22300 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.6687608318890814

test score: 0.409985935302391

Table 4.87: Confusion matrix from test data of RFC model: Landmark RFC Search Pi 6

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 973 | 4 | 3013 |
| true neutral | 101 | 0 | 5 |
| true positive | 1083 | 0 | 501 |

———————————————————-

Landmarks RFC Search PC 0

Table 4.88: The parameters of RFC model: Landmark RFC Search PC 0

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 1219 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 32390 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.587069372345446

test score: 0.6060126582278481

Table 4.89: Confusion matrix from test data of RFC model: Landmark RFC Search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 2692 | 52 | 1246 |
| true neutral | 81 | 10 | 23 |
| true positive | 836 | 3 | 745 |

———————————————————-

Landmarks RFC Search PC 1

Table 4.90: The parameters of RFC model: Landmark RFC Search PC 1

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 2227 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 8 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 49545 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.6775039328788673

test score: 0.5497538677918424

Table 4.91: Confusion matrix from test data of RFC model: Landmark RFC Search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 2317 | 049 | 1624 |
| true neutral | 61 | 15 | 38 |
| true positive | 783 | 6 | 795 |

———————————————————-

Landmarks RFC Search PC 2

Table 4.92: The parameters of RFC model: Landmark RFC Search PC 2

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 665 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 2 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 70736 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5489864864864865

test score: 0.5070323488045007

Table 4.93: Confusion matrix from test data of RFC model: Landmark RFC Search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 2157 | 544 | 1289 |
| true neutral | 40 | 50 | 24 |
| true positive | 726 | 181 | 677 |

## 4.9 Results of ML application 9: RSCV for SVC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

In total 12 models were trained using RSCV. All these models are named in a format of: Landmark SVC search 0.5 Pi *i*, where *i* stands for the iteration of this application.

The list of finished models is following:

Landmark SVC search 0.5 Pi 0, Landmark SVC search 0.5 Pi 1, Landmark SVC search 0.5 Pi 2, Landmark SVC search 0.5 Pi 3, Landmark SVC search 0.5 Pi 4, Landmark SVC search 0.5

Pi 5, Landmark SVC search 0.5 Pi 6, Landmark SVC search 0.5 Pi 7, Landmark SVC search 0.5 Pi 8, Landmark SVC search 0.5 Pi 9, Landmark SVC search 0.5 Pi 10 and Landmark SVC search 0.5 Pi 11.

The assessment and format of result representation is exactly identical as the one described in section 4.5.

---

Landmark SVC search 0.5 Pi 0

Table 4.94: The parameters of SVC model: Landmark SVC search 0.5 Pi 0

| | |
|---|---|
| 'C' | 500 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 40600 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4839580857729286

test score: 0.6998780700226441

Table 4.95: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

---

Landmark SVC search 0.5 Pi 1

Table 4.96: The parameters of SVC model: Landmark SVC search 0.5 Pi 1

| | |
|---|---|
| 'C' | 49500 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 59100 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4814734795290051

test score: 0.28009057655460723

Table 4.97: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 2

Table 4.98: The parameters of SVC model: Landmark SVC search 0.5 Pi 2

| | |
|---|---|
| 'C' | 17500 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 92300 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4814734795290051

test score: 0.28009057655460723

Table 4.99: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 3

Table 4.100: The parameters of SVC model: Landmark SVC search 0.5 Pi 3

| | |
|---|---|
| 'C' | 24600 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 67400 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4814734795290051

test score: 0.28009057655460723

Table 4.101: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 4

Table 4.102: The parameters of SVC model: Landmark SVC search 0.5 Pi 4

| | |
|---|---|
| 'C' | 45900 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 56200 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.37735849056603776

test score: 0.28009057655460723

Table 4.103: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 4

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 5

Table 4.104: The parameters of SVC model: Landmark SVC search 0.5 Pi 5

| | |
|---|---|
| 'C' | 39100 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 33800 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4716981132075472

test score: 0.6998780700226441

Table 4.105: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 5

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

———————————————————-

Landmark SVC search 0.5 Pi 6

Table 4.106: The parameters of SVC model: Landmark SVC search 0.5 Pi 6

| | |
|---|---|
| 'C' | 57600 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 86900 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.37735849056603776

test score: 0.28009057655460723

Table 4.107: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 6

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 7

Table 4.108: The parameters of SVC model: Landmark SVC search 0.5 Pi 7

| | |
|---|---|
| 'C' | 69500 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 76800 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4716981132075472

test score: 0.6998780700226441

Table 4.109: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 7

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

———————————————————-

Landmark SVC search 0.5 Pi 8

Table 4.110: The parameters of SVC model: Landmark SVC search 0.5 Pi 8

| | |
|---|---|
| 'C' | 69400 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 90700 |
| 'kernel' | 'poly' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.37735849056603776

test score: 0.28009057655460723

Table 4.111: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 8

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 4018 |
| true neutral | 0 | 0 | 115 |
| true positive | 0 | 0 | 1608 |

———————————————————-

Landmark SVC search 0.5 Pi 9

Table 4.112: The parameters of SVC model: Landmark SVC search 0.5 Pi 9

| | |
|---|---|
| 'C' | 78300 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 52700 |
| 'kernel' | 'sigmoid' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4716981132075472

test score: 0.6998780700226441

Table 4.113: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 9

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

———————————————————-

Landmark SVC search 0.5 Pi 10

Table 4.114: The parameters of SVC model: Landmark SVC search 0.5 Pi 10

| | |
|---|---|
| 'C' | 57000 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 74700 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4716981132075472

test score: 0.6998780700226441

Table 4.115: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 10

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

———————————————————-

Landmark SVC search 0.5 Pi 11

Table 4.116: The parameters of SVC model: Landmark SVC search 0.5 Pi 11

| | |
|---|---|
| 'C' | 68700 |
| 'cache_size' | 200 |
| 'class_weight' | None |
| 'coef0' | 0.0 |
| 'decision_function_shape' | 'ovr' |
| 'degree' | 3 |
| 'gamma' | 77300 |
| 'kernel' | 'rbf' |
| 'max_iter' | -1 |
| 'probability' | False |
| 'random_state' | None |
| 'shrinking' | True |
| 'tol' | 0.001 |
| 'verbose' | False |

search score: 0.4716981132075472

test score: 0.6998780700226441

Table 4.117: Confusion matrix from test data of SVC model: Landmark SVC search 0.5 Pi 11

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 4018 | 0 | 0 |
| true neutral | 115 | 0 | 0 |
| true positive | 1608 | 0 | 0 |

## 4.10 Results of ML application 10: RSCV for RFC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

In total 4 models were trained using RSCV. All these models are named in a format of: Landmark RFC Search 0.5 Pi *i*, where *i* stands for the iteration of this application.

The list of finished models is following:
Landmark RFC Search 0.5 Pi 0, Landmark RFC Search 0.5 Pi 1, Landmark RFC Search 0.5 Pi 2, Landmark RFC search 0.5 Pi 3.

The assessment and format of result representation is exactly identical as the one described in section 4.5.

———————————————————-

Landmarks RFC Search 0.5 Pi 0

Table 4.118: The parameters of RFC model: Landmark RFC Search 0.5 Pi 0

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'gini' |
| 'max_depth' | 413 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 4 |
| 'min_samples_split' | 16 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 34409 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5240566037735849

test score: 0.5796899494861523

Table 4.119: Confusion matrix from test data of RFC model: Landmark RFC Search PC 0.5 Pi 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 2678 | 42 | 1298 |
| true neutral | 81 | 0 | 34 |
| true positive | 94 | 17 | 650 |

---

Landmarks RFC Search 0.5 Pi 1

Table 4.120: The parameters of RFC model: Landmark RFC Search 0.5 Pi 1

| | |
|---|---|
| 'bootstrap' | True |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 2933 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 4 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 9181 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5679245283018868

test score: 0.646577251349939

Table 4.121: Confusion matrix from test data of RFC model: Landmark RFC Search 0.5 Pi 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3259 | 45 | 714 |
| true neutral | 111 | 0 | 4 |
| true positive | 1143 | 12 | 453 |

---

Landmarks RFC Search 0.5 Pi 2

Table 4.122: The parameters of RFC model: Landmark RFC Search 0.5 Pi 2

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 1018 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 2 |
| 'min_samples_split' | 16 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 64681 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5556603773584906

test score: 0.6568542065842188

Table 4.123: Confusion matrix from test data of RFC model: Landmark RFC Search 0.5 Pi 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 3062 | 30 | 926 |
| true neutral | 79 | 0 | 36 |
| true positive | 886 | 13 | 709 |

———————————————————-

Landmarks RFC Search 0.5 Pi 3

Table 4.124: The parameters of RFC model: Landmark RFC Search 0.5 Pi 3

| | |
|---|---|
| 'bootstrap' | False |
| 'class_weight' | None |
| 'criterion' | 'entropy' |
| 'max_depth' | 5000 |
| 'max_features' | 'auto' |
| 'max_leaf_nodes' | None |
| 'min_impurity_decrease' | 0.0 |
| 'min_impurity_split' | None |
| 'min_samples_leaf' | 1 |
| 'min_samples_split' | 2 |
| 'min_weight_fraction_leaf' | 0.0 |
| 'n_estimators' | 14227 |
| 'n_jobs' | None |
| 'oob_score' | False |
| 'random_state' | None |
| 'verbose' | 0 |
| 'warm_start' | False |

search score: 0.5509433962264151

test score: 0.5983278174534054

Table 4.125: Confusion matrix from test data of RFC model: Landmark RFC Search 0.5 Pi 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 2748 | 50 | 1220 |
| true neutral | 75 | 2 | 38 |
| true positive | 907 | 16 | 685 |

## 4.11 Results of Histogram of oriented gradients, data size estimation

Having calculated all 81 HOG vectors, they each were saved as a binary .npy file. The sizes of files was then measured using *path.getsize()* method found in python's *os* library and then translated into either MiB or GiB depending on the result. The file size's dependence in relation to **cells_per_block** and **pixels_per_cell** parameters are depicted in the Tables 4.126 and 4.127.

Table 4.126: The correspondence of resulted HOG vectors' size in relation to the value of **pixels_per_cell** parameter (x axis) and the value of **cells_per_block** parameter (y axis), part 1

| | **8 pixels** | **16 pixels** | **24 pixels** | **32 pixels** | **40 pixels** |
|---|---|---|---|---|---|
| **1 cell** | 19.22MiB | 4.81MiB | 2.14MiB | 1.20MiB | 0.76MiB |
| **2 cells** | 76.59MiB | 19.07MiB | 8.44MiB | 4.73MiB | 2.99MiB |
| **3 cells** | 0.17GiB | 42.58MiB | 18.78MiB | 10.48MiB | 6.59MiB |
| **4 cells** | 0.30GiB | 75.12MiB | 33.00MiB | 18.34MiB | 11.49MiB |
| **5 cells** | 0.46GiB | 0.11GiB | 50.95MiB | 28.21MiB | 17.60MiB |
| **6 cells** | 0.66GiB | 0.16GiB | 72.50MiB | 39.97MiB | 24.83MiB |
| **7 cells** | 0.90GiB | 0.22GiB | 0.10GiB | 53.54MiB | 33.11MiB |
| **8 cells** | 1.17GiB | 0.28GiB | 0.12GiB | 68.80MiB | 42.35MiB |
| **9 cells** | 1.47GiB | 0.36GiB | 0.15GiB | 85.65MiB | 52.49MiB |

Table 4.127: The correspondence of resulted HOG vectors' size in relation to the value of **pixels_per_cell** parameter (x axis) and the value of **cells_per_block** parameter (y axis), part 2

| | **48 pixels** | **56 pixels** | **64 pixels** | **72 pixel** |
|---|---|---|---|---|
| **1 cell** | 0.53MiB | 0.39MiB | 0.30MiB | 0.24MiB |
| **2 cells** | 2.09MiB | 1.50MiB | 1.16MiB | 0.92MiB |
| **3 cells** | 4.59MiB | 3.28MiB | 2.54MiB | 1.99MiB |
| **4 cells** | 7.96MiB | 5.67MiB | 4.37MiB | 3.41MiB |
| **5 cells** | 12.14MiB | 8.61MiB | 6.61MiB | 5.14MiB |
| **6 cells** | 17.06MiB | 12.04MiB | 9.21MiB | 7.12MiB |
| **7 cells** | 22.65MiB | 15.91MiB | 12.11MiB | 9.33MiB |
| **8 cells** | 28.85MiB | 20.17MiB | 15.28MiB | 11.71MiB |
| **9 cells** | 35.60MiB | 24.76MiB | 18.68MiB | 14.24MiB |

## 4.12 Results ML application 11: SVC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

In total 7 models were trained following this application. All these models are named in a format of: HOG 72 op 1 pc*m* SVC search PC *i*,
where *m* denotes the maximal number of samples a class could get,
and *i* represents the iteration of an iteration of this application with respective *m* value.

The list of finished models is following:
HOG 72 p 1 pc400 SVC search PC 0, HOG 72 p 1 pc400 SVC search PC 1, HOG 72 p 1 pc400 SVC search PC 2, HOG 72 p 1 pc400 SVC search PC 3, HOG 72 p 1 pc4000 SVC search PC 0, HOG 72 p 1 pc4000 SVC search PC 1 and HOG 72 p 1 pc4000 SVC search PC 2.

Once again for assessment methods *sklearn.metrics.confusion_matrix()* and *.score()* were implemented.
The results are represented in a following format:

———————————————————-

*model's name* (HOG 72 op 1 pc*m* SVC search PC *i*)
maximal number of samples per class: *m*
scoring: *returned value of .score() method*

Table 4.128: An example of a confusion matrix for the given model

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | number of negative samples predicted as negative | number of negative samples predicted as neutral | number of negative samples predicted as positive |
| true neutral | number of neutral samples predicted as negative | number of neutral samples predicted as neutral | number of neutral samples predicted as positive |
| true positive | number of positive samples predicted as negative | number of positive samples predicted as neutral | number of positive samples predicted as positive |

---

HOG 72 p 1 pc400 SVC search PC 0

maximal number of samples per class: 400

scoring: 0.33482142857142855

Table 4.129: Confusion matrix from test data of SVC model: HOG 72 p 1 pc400 SVC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 75 | 0 | 0 |
| true neutral | 67 | 0 | 0 |
| true positive | 82 | 0 | 0 |

---

HOG 72 p 1 pc400 SVC search PC 1

maximal number of samples per class: 400

scoring: 0.35267857142857145

Table 4.130: Confusion matrix from test data of SVC model: HOG 72 p 1 pc400 SVC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 79 | 0 | 0 |
| true neutral | 60 | 0 | 0 |
| true positive | 85 | 0 | 0 |

---

HOG 72 p 1 pc400 SVC search PC 2

maximal number of samples per class: 400

scoring: 0.3482142857142857

Table 4.131: Confusion matrix from test data of SVC model: HOG 72 p 1 pc400 SVC search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 85 |
| true neutral | 0 | 0 | 61 |
| true positive | 0 | 0 | 78 |

---

HOG 72 p 1 pc400 SVC search PC 3

maximal number of samples per class: 400

scoring: 0.32589285714285715

Table 4.132: Confusion matrix from test data of SVC model: HOG 72 p 1 pc400 SVC search PC 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 93 |
| true neutral | 0 | 0 | 58 |
| true positive | 0 | 0 | 73 |

---

HOG 72 p 1 pc4000 SVC search PC 0

maximal number of samples per class: 4000

scoring: 0.47596153846153844

Table 4.133: Confusion matrix from test data of SVC model: HOG 72 p 1 pc4000 SVC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 792 | 0 | 0 |
| true neutral | 72 | 0 | 0 |
| true positive | 800 | 0 | 0 |

---

HOG 72 p 1 pc4000 SVC search PC 1

maximal number of samples per class: 4000

scoring: 0.4753605769230769

Table 4.134: Confusion matrix from test data of SVC model: HOG 72 p 1 pc4000 SVC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 809 |
| true neutral | 0 | 0 | 64 |
| true positive | 0 | 0 | 791 |

---

HOG 72 p 1 pc4000 SVC search PC 2

maximal number of samples per class: 4000

scoring: 0.47896634615384615

Table 4.135: Confusion matrix from test data of SVC model: HOG 72 p 1 pc4000 SVC search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 0 | 0 | 808 |
| true neutral | 0 | 0 | 59 |
| true positive | 0 | 0 | 757 |

## 4.13 Results of application 12: RFC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

In total 7 models were trained following this application. All these models are named in a format of: HOG 72 op 1 pc*m* RFC search PC *i*,
where *m* denotes the maximal number of samples a class could get,
and *i* represents the iteration of an iteration of this application with respective *m* value.

The list of finished models is following:
HOG 72 p 1 pc400 RFC search PC 0, HOG 72 p 1 pc400 RFC search PC 1, HOG 72 p 1 pc400 RFC search PC 2, HOG 72 p 1 pc400 RFC search PC 3, HOG 72 p 1 pc4000 RFC search PC 0, RFC 72 p 1 pc4000 SVC search PC 1 and RFC 72 p 1 pc4000 SVC search PC 2.

The assessment of models and result representation follows the format described in section 4.12.

---

HOG 72 p 1 pc400 RFC search PC 0

maximal number of samples per class: 400

scoring: 0.40625

Table 4.136: Confusion matrix from test data of RFC model: HOG 72 p 1 pc400 RFC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 32 | 12 | 27 |
| true neutral | 22 | 25 | 20 |
| true positive | 35 | 17 | 29 |

---

HOG 72 p 1 pc400 RFC search PC 1

maximal number of samples per class: 400

scoring: 0.4375

Table 4.137: Confusion matrix from test data of RFC model: HOG 72 p 1 pc400 RFC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 32 | 9 | 27 |
| true neutral | 14 | 40 | 9 |
| true positive | 36 | 10 | 33 |

---

HOG 72 p 1 pc400 RFC search PC 2

maximal number of samples per class: 400

scoring: 0.53125

Table 4.138: Confusion matrix from test data of RFC model: HOG 72 p 1 pc400 RFC search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 46 | 9 | 27 |
| true neutral | 14 | 40 | 9 |
| true positive | 36 | 10 | 33 |

---

HOG 72 p 1 pc400 RFC search PC 3

maximal number of samples per class: 400

scoring: 0.5044642857142857

Table 4.139: Confusion matrix from test data of RFC model: HOG 72 p 1 pc400 RFC search PC 3

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 57 | 8 | 20 |
| true neutral | 19 | 26 | 19 |
| true positive | 30 | 15 | 30 |

---

HOG 72 p 1 pc4000 RFC search PC 0

maximal number of samples per class: 4000

scoring: 0.6243990384615384

Table 4.140: Confusion matrix from test data of RFC model: HOG 72 p 1 pc4000 RFC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 562 | 0 | 225 |
| true neutral | 33 | 10 | 35 |
| true positive | 332 | 0 | 467 |

---

HOG 72 p 1 pc4000 RFC search PC 1

maximal number of samples per class: 4000

scoring: 0.6358173076923077

Table 4.141: Confusion matrix from test data of RFC model: HOG 72 p 1 pc4000 RFC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 607 | 1 | 217 |
| true neutral | 29 | 3 | 23 |
| true positive | 336 | 0 | 448 |

---

HOG 72 p 1 pc4000 RFC search PC 2

maximal number of samples per class: 4000

scoring: 0.6207932692307693

Table 4.142: Confusion matrix from test data of RFC model: HOG 72 p 1 pc4000 RFC search PC 2

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 553 | 0 | 251 |
| true neutral | 30 | 4 | 30 |
| true positive | 320 | 0 | 476 |

## 4.14 Results of ML application 13: RFC negative VS neutral VS positive with HOG vectors of smallest resolution from iCV MEFED

Only 2 models were trained following this application. All these models are named in a format of: HOG 8 op 1 pc*m* RFC search PC *i*,

where *m* denotes the maximal number of samples a class could get,

and *i* represents the iteration of an iteration of this application with respective *m* value.

The list of finished models is following:

HOG 8 p 1 pc1000 RFC search PC 0, HOG 8 p 1 pc1000 RFC search PC 1.

The assessment of models and result representation follows the format described in section 4.12.

---

HOG 8 p 1 pc1000 RFC search PC 0

maximal number of samples per class: 1000

scoring: 0.41

Table 4.143: Confusion matrix from test data of RFC model: HOG 8 p 1 pc1000 RFC search PC 0

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 51 | 14 | 35 |
| true neutral | 0 | 0 | 0 |
| true positive | 56 | 13 | 31 |

---

HOG 8 p 1 pc1000 RFC search PC 1

maximal number of samples per class: 1000

scoring: 0.47

Table 4.144: Confusion matrix from test data of RFC model: HOG 8 p 1 pc1000 RFC search PC 1

| | predicted negative | predicted neutral | predicted positive |
|---|---|---|---|
| true negative | 58 | 12 | 30 |
| true neutral | 0 | 0 | 0 |
| true positive | 54 | 10 | 36 |

# 5 Analysis

## 5.1 Analysis of ML application 1: SVM one VS one with FER2013

Despite the fact of high scoring result (even as far as being over 0.9 for anger VS disgust, neutral VS disgust, disgust VS fear, disgust VS sadness), confusion matrices show the absolute chaos of classifications and the high scoring values are nothing more than a coincidence with class imbalance. This proves the inability of the default parameters to efficiently influence a training model in the scope of a presented task.

## 5.2 Analysis of ML application 2: SVM one VS rest with FER2013

Just like in the previous case, scoring outputs are deceptively high, most exceeding 0.83 success rate. Confusion matrices shed light on the actual cause of such performance. The models most of the time predict the *rest* class, due to the sheer size of the data belonging to this class. Its dataset can be 3 to 6 (if not even more) times bigger than the database of a single emotion. The *rest* class is simply more likely to appear, based on the training data. Moreover, even if during the testing a model fails to predict instances of a singled out emotion, the number of successful *rest* responses will greatly outweigh them, which is exactly the case here.

## 5.3 Analysis of ML application 3: SVM all VS all with FER2013

We can see, that the overwhelming majority of predictions result in *happiness* class (quite an optimistic model, one can say). The reason for this may be that the default parameters do not

activate tie breaking based on the probability. Instead the model would return the first prediction made when the input falls onto shared regions of sub-classifiers. Apparently, happiness would be predicted before any other computation could be made.

## 5.4 Analysis of ML application 4: SVM negative VS positive + neutral with FER2013

Unfortunately, controlling the shape of training data did not bolster the results in any meaningful way. Neither has parameter tuning yielded any significant impact.

## 5.5 Analysis of ML application 5: RSCV for RFC negative VS neutral VS positive with JAFFE

Here we can see an obvious spike in performance. This shows how efficient RF can be in terms of being an estimator in the field of image processing. The improved results may also be attributed to the database used, for its uniformity and consistency. However, it is also fair to mention the narrow range of diversity and the general small size of the JAFFE database. The application 5 gives a bit of understanding concerning the parameters of a RFC, although no concrete correlation between the combination of parameters and performance is observed. Coupled with the downsides of the database, the resulted models are not suitable to be implemented as a solution to step 3 of section 3,7.

## 5.6 Analysis of ML application 6: RSCV for RFC negative VS neutral VS positive with iCV MEFED

It was obvious something was wrong in the code. Turns out there was a bug which read the samples incorrectly, constantly returning None instead of an array. Quite peculiar, that RFC accept an array of None as a training data. The results depicted prior now stand as a testimony to the lost time, because each of failed models took around a week to train, let alone all the unfinished models, which took considerably more training time and had to be stopped. The bug was not the only issue, however. After the bug has been removed, the training data could not fit

into RAM, periodically raising a memory error. An alternative method should be found instead.

## 5.7 Analysis of ML application 7: RSCV for SVC negative VS neutral VS positive with facial landmarks from iCV MEFED

No matter the parameter, every model was constantly predicting *negative* class. It may be due to the samples' value being extremely similar and SVC did not compute probability, instead only returning the first predicted class.

## 5.8 Analysis of ML application 8: RSCV for RFC negative VS neutral VS positive with facial landmarks from iCV MEFED

The results are visibly better than the results from the previous approach. It must be noted that all the assessments from CV during parameter search are above 0.5, some even are going as high as 0.7, although most of the models fail at efficiently predicting classes of the test dataset. Only Pi 3, Pi 4 and Pi 5 have their test score consistently high enough with their search score. An interesting common point of these models is the fact of never having returned any *neutral* class instance, neither right nor wrong. Moreover, these models are more biased towards predicting *negative* class. However, this is where the similarities end. The saddest part is, that their parameters do not seem to match among each other, which makes it hard to discern meaningfully useful values for them.

## 5.9 Analysis of ML application 9: RSCV for SVC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

Results practically resemble the results of ML application 7: now in addition to the models returning *negative* class for any input, are also models always predicting only *positive* class.

## 5.10 Analysis of ML application 10: RSCV for RFC negative VS neutral VS positive with facial landmarks from twice shrunk iCV MEFED

The result are similar to the ones from the ML application 8, albeit slightly worse. Perhaps, some essential features were lost after downsizing.

Around this time author has noticed that Raspberry Pi is using a 32-bit version Python. A following query has revealed, that the default operating system for Raspberry Pi 4 is but a 32-bit OS, which is not able to fully utilize the 64-bit processor and 4 GB RAM installed on the board [95, 96].This explains the constant memory errors and long training times. With this information in mind, any future ML on the Raspberry ceased, since the Raspberry is not tested to its fullest and handling with the 32-bit OS is but a waste of time.

## 5.11 Analysis of Histogram of oriented gradients, data size estimation

We can see that HOG vector's size correlates negatively with **pixels_per_cell** parameter, but positively with **cells_per_block**. Moreover, the **pixels_per_cell** parameter's correlation is stronger, than of **cells_per_block**. The smallest size, which is less than a third of a MiB, is achieved using **pixels_per_cell** of 72 and **cells_per_block** of just 1, while the largest (ca one and a half GiB) size is achieved by specifying 8 **pixels_per_cell** and 9 **cells_per_block**.

## 5.12 Analysis ML application 11: SVC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

Results practically resemble the results of ML application 7 and 9.

## 5.13 Analysis of application 12: RFC negative VS neutral VS positive with HOG vectors of smallest size from iCV MEFED

Yet again, RFC was outperforming SVC at the similar task. Models which could have up to 4000 samples per class had a higher accuracy than the models with a stricter limitation. The best models possessed a score quite similar to the score of best models trained on Facial Landmarks feature from ML application 8.

## 5.14 Analysis of ML application 13: RFC negative VS neutral VS positive with HOG vectors of smallest resolution from iCV MEFED

The models performed noticeably worse than the models from the previous application. This can be addressed to strict a limitations of only 1000 samples per class, however any larger training set caused memory errors.

## 5.15 Analysis conclusions

Numerous attempts were carried out in order to create an ML model able to accurately classify a human's current valence of emotion based on an image of said human's current facial expression. Unfortunately, none of the attempts resulted with a sufficiently high performance.

Out of all the models the equally best results gave RFC models, which were trained on either Facial Landmarks features or HOG vectors. Those results were slightly higher than 60%. Coincidentally, somewhat similar results were achieved in the project, form where the idea of using Facial Landmarks detector and HOG vectors was adapted from [86].

Without finding a working solution for valence of human emotion classification (step 3; Section 3.7) the project cannot move further and because of the approaching deadline shall remain at this stage of development.

# 6 Conclusion

This paper touches many various fields of study: clinical psychiatry, neural biology, mathematical statistics etc. Most importantly, it ventures on a bridge linking human and computer recognition. A little step to bring human-machine interaction ever closer to each over. An extremely ambitious direction, one may say. Especially for an undergrad student. Author had to learn many new topics, acquire fresh practical experience, try out multiple branches of problem solving. Despite not reaching the final goal, the overall intermediary process of exploring left a beneficial mark. Having that said, the author wishes to reflect back on some aspects of the project.

The hardware used in for this project uncovered as unsuitable. The new out-of-the-box Raspberry Pi 4, although marketed as a rightful alternative to a desktop computer, was unable to efficiently deal with the difficult tasks provided by the project. In its place, an average laptop could not operate skillfully with the imposing assignments, it never meant for, as well. Most likely, an introduction of better hardware could expand the project further in its wake. At least, in ML model training position and leave the prepared model for the smaller embedded system to utilize in the final implementation.

Machine learning is a promising, yet chaotic methodology. The results can potentially surpass anything a human can produce in terms of sheer computational competence, but it balances it out with the multitude of mostly unguessable attributes, which can only be obtained through trial and error, when each error is extortionately expensive. Countless models took days of training time without any foreseeable outcome and had to be stopped. As the point of better hardware rises up again, algorithms with reportedly better performance than of Support Vector Machine and Random Forest such as Deep Neural Networks could be investigated.

In the end, all these possibilities remain in the realms of theoretical future, outside of the given paper's scope. This paper, however, offers a look at different methods and solutions with a deeper than surface level examination of their inner workings.

# Acknowledgements

The author would like to express gratitude toward the supervisor of this paper, Prof. Gholamreza Anbarjafari, for the advises and guidance throughout this project and for providing the iCV MEFED database. Also the author wishes to thank the University of Tartu for granting the ability to study and the reason of writing this very paper.

# References


[1] & Company blog. - SHOW ME THE MAYFAIR (FILTER!): A QUICK GUIDE TO INSTAGRAM.
https://andcompanysaid.com/show-me-the-mayfair-filter-a-quick-guide-to-instagram/ 19.05.2020.

[2] Y. Rabhi, M. Mrabet, F. Fnaiech and M. Sayadi, "A real-time emotion recognition system for disabled persons," textit2018 4th International Conference on Advanced Technologies for Signal and Image Processing (ATSIP), Sousse, 2018, pp. 1-6, doi: 10.1109/ATSIP.2018.8364339.

[3] D. Yang, T. Kunihiro, H. Shimoda and H. Yoshikawa, "A study of real-time image processing method for treating human emotion by facial expression," *IEEE SMC'99 Conference Proceedings. 1999 IEEE International Conference on Systems, Man, and Cybernetics (Cat. No.99CH37028)*, Tokyo, Japan, 1999, pp. 360-364 vol.2, doi: 10.1109/ICSMC.1999.825285.

[4] CD Project Red, The Witcher 3 Forum. - FACIAL EXPRESSIONS IN WITCHER 3.
https://forums.cdprojektred.com/index.php?threads/facial-expressions-in-witcher-3.34493/ 10.12.2018.

[5] Destructoid. - Uncharted animator explains wacky Mass Effect: Andromeda facial expressions.
https://www.destructoid.com/uncharted-animator-explains-wacky-mass-effect-andromeda-facial-expressions-427083.phtml 10.12.2018.

[6] Adobe Blog. - Woo Hoo! The Simpsons TV Show and Adobe Make Live Animation Television History.
https://theblog.adobe.com/the-simpsons-tv-show-and-adobe-make-live-animation-television-history 19.05.2020.

[7] Youtube, A.I.Channel.
https://www.youtube.com/channel/UC4YaOt1yT-ZeyB0OmxHgolA 19.05.2020.

[8] Wikipedia, the free encyclopedia. - Bipolar Disorder.
https://en.wikipedia.org/wiki/Bipolar_disorder 19.05.2020.

[9] Wikipedia, the free encyclopedia. - Social-emotional agnosia.
https://en.wikipedia.org/wiki/Social-emotional_agnosia 19.05.2020.

[10] Lexico, Powered by OXFORD. - emotion.
https://www.lexico.com/definition/emotion 19.05.2020.

[11] Oxford Learner's Dictionaries. - emotion.
https://www.oxfordlearnersdictionaries.com/definition/english/emotion 19.05.2020.

[12] Dictionary.com. - emotion.
https://www.dictionary.com/browse/emotion 19.05.2020.

[13] Collins dictionary. - emotion.
https://www.collinsdictionary.com/dictionary/english/emotion 19.05.2020.

[14] Merriam-Webster. - emotion.
https://www.merriam-webster.com/dictionary/emotion 19.05.2020.

[15] Wikipedia, the free encyclopedia. - Emotion.
https://en.wikipedia.org/wiki/Emotion 19.05.2020.

[16] P. Ekman."Universal Facial Expressions of Emotions". *California Mental Health Research Digest*, **8(4)**, 1970, 151-158.

[17] Ekman, P. (1972). Universals and Cultural Differences in Facial Expressions of Emotions. In Cole, J. (Ed.), *Nebraska Symposium on Motivation* (pp. 207-282). Lincoln, NB: University of Nebraska Press.

[18] Wikipedia, the free encyclopedia. - Emotion classification.
https://en.wikipedia.org/wiki/Emotion_classification 19.05.2020.

[19] Classics in the History of Psychology. - "Outlines of Psychology", Wilhelm Max Wundt (1897), Translated by Charles Hubbard Judd (1897), II. PSYCHICAL COMPOUNDS,

12. COMPOSITE FEELINGS.
https://psychclassics.yorku.ca/Wundt/Outlines/sec12.htm 19.05.2020.

[20] J. A. Russell, "Affective space is bipolar", *Journal of Personality and Social Psychology* **37(3)**, 1979, 345356, DOI:10.1037/0022-3514.37.3.345.

[21] D. Watson, A. Tellegen, "Toward a consensual structure of mood", *Psychological Bulletin*, **98(2)**, 1985, 219235, DOI:10.1037/0033-2909.98.2.219.

[22] J. Posner, J. A. Russell, BS Peterson, "The circumplex model of affect: an integrative approach to affective neuroscience, cognitive development, and psychopathology", *Development and psychopathology*, **17(3)**, 2005, 715734, DOI:10.1017/S0954579405050340.

[23] D. C. Rubin, JM. Talarico, "A comparison of dimensional models of emotion: evidence from emotions, prototypical events, autobiographical memories, and words", *Memory (Hove, England)*, **17(8)**, 2009, 802808, DOI:10.1080/09658210903130764.

[24] D. Watson, D. Wiese, J. Vaidya, A. Tellegen, "The Two General Activation Systems of Affect: Structural Findings, Evolutionary Considerations, and Psychobiological Evidence", *Journal of Personality and Social Psychology*, **76**, 1999 ,820-838, DOI:10.1037/0022-3514.76.5.820.

[25] M. Pirkowska M. Wrobel, "Basic Emotions", 2017, DOI:10.1007/978-3-319-28099-$8495 - 1$.

[26] The Tomkins institute. - Nine affects, present at birth, combine with life experience to form emotion and personality.
http://www.tomkins.org/what-tomkins-said/introduction/nine-affects-present-at-birth-combine-to-form-emotion-mood-and-personality/ 19.05.2020.

[27] International Institute for Restorative Practices. - Defining Restorative; 4.4. Nine Affects.
https://www.iirp.edu/defining-restorative/nine-affects 19.05.2020.

[28] P. Ekman, W. V. Friesen, "Constants across cultures in the face and emotion", *Journal of Personality and Social Psychology*, **17(2)**, 1971, 124129. DOI:10.1037/h0030377.

[29] P. Ekman, W. V. Friesen, The Repertoire of Nonverbal Behavior: Categories, Origins, Usage, and Coding, *Semiotica*, **1**, 1969, 49-98, DOI:10.1515/semi.1969.1.1.49.

[30] P. Ekman, W. V. Friesen, Basic Emotions in *Handbook of Cognition and Emotion*, T. Dalgleish, M. Power, ed. Academic, University of California, San Francisco, 1999, pp. 45-60.

[31] The Emotion Machine. - Classification of Emotions. http://www.theemotionmachine.com/classification-of-emotions/ 19.05.2020.

[32] Personality Research. - basicemotions, Plutchik. http://www.personalityresearch.org/basicemotions/plutchik.html 19.05.2020.

[33] Adliterate. - archive, Robert Plutchik's. PSYCHOEVOLUTIONARY THEORY OF BASIC EMOTIONS. http://www.adliterate.com/archives/Plutchik.emotion.theorie.POSTER.pdf 19.05.2020.

[34] Wikimedia Commons. - File:Plutchik-wheel.svg. https://commons.wikimedia.org/wiki/File:Plutchik-wheel.svg 19.05.2020.

[35] Toolshero. - Emotion Wheel by Robert Plutchik. https://www.toolshero.com/psychology/emotion-wheel-robert-plutchik/ 19.05.2020.

[36] Positive psychology. - The Emotion Wheel: What It Is and How to Use It. https://positivepsychology.com/emotion-wheel/ 19.05.2020.

[37] A. S. Cowen, and D. Keltner. Self-report captures 27 distinct categories of emotion bridged by continuous gradients. *Proceedings of the National Academy of Sciences of the United States of America*, **114,38**, 2017 E7900-E7909, DOI:10.1073/pnas.1702247114.

[38] Raspberry Pi products. - Raspberry Pi 4 Model B. https://www.raspberrypi.org/products/raspberry-pi-4-model-b/ 19.05.2020.

[39] Raspberry Pi blog, - Raspberry Pi 4 on sale now from $35. https://www.raspberrypi.org/blog/raspberry-pi-4-on-sale-now-from-35/ 19.05.2020.

[40] Labists, products. - LABISTS Raspberry Pi 4 4GB Complete Starter Kit with 32GB Micro SD Card. https://labists.com/collections/best-seller/products/labists-raspberry-pi-4g-ram-32gb-card 19.05.2020.

[41] Raspberry Pi, product. - Raspberry Pi 4 Tech Specs.
https://www.raspberrypi.org/products/raspberry-pi-4-model-b/specifications/ 19.05.2020.

[42] Raspberry Pi, files. - Raspberry Pi4 Computer Model B brief.
https://static.raspberrypi.org/files/product-briefs/200206+Raspberry+Pi+4+1GB+2GB+4GB+Product+B
19.05.2020.

[43] Raspberry Pi, MagPi. - Raspberry Pi 4 specs and benchmarks.
https://magpi.raspberrypi.org/articles/raspberry-pi-4-specs-benchmarks 19.05.2020.

[44] Youtube - Display board test.
https://www.youtube.com/watch?v=uXTC86OBOzM 19.05.2020.

[45] Google drive. - Zip file with Gerber files and schematic for the display board.
https://drive.google.com/file/d/1iHimIezZrT7o61-xzEsDlJY6Fgw0df5v/view?usp=sharing
19.05.2020.

[46] Wikipedia, the free encyclopedia. - Pareidolia.
https://en.wikipedia.org/wiki/Pareidolia 19.05.2020.

[47] Medium. - Artificial Intelligence vs. Machine Learning vs. Deep Learning: Whats the Difference.
https://medium.com/ai-in-plain-english/artificial-intelligence-vs-machine-learning-vs-deep-learning-whats-the-difference-dccce18efe7f 19.05.2020.

[48] Mathworks. - What Is Machine Learning?
https://www.mathworks.com/discovery/machine-learning.html 19.05.2020.

[49] Wikipedia, the free encyclopedia. - No free lunch theorem.
https://en.wikipedia.org/wiki/No_free_lunch_theorem 19.05.2020.

[50] Github. - DeepEmotion.
https://github.com/SkyAndCloud/DeepEmotion 19.05.2020.

[51] Kaggle - Challenges in Representation Learning: Facial Expression Recognition Challenge.
https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/leaderboard 19.05.2020.

[52] Analytics Vidhya. - Understanding Support Vector Machine(SVM) algorithm from examples (along with code).
https://www.analyticsvidhya.com/blog/2017/09/understaing-support-vector-machine-example-code/ 19.05.2020.

[53] Medium. - Machine learning 101, Chapter 2 : SVM (Support Vector Machine) Theory.
https://medium.com/machine-learning-101/chapter-2-svm-support-vector-machine-theory-f0812effc72 19.05.2020.

[54] Towards Data Science. - Support Vector Machine: Kernel Trick; Mercers Theorem.
https://towardsdatascience.com/understanding-support-vector-machine-part-2-kernel-trick-mercers-theorem-e1e6848c6c4d 19.05.2020.

[55] Towards Data Science, - Support Vector Machine Introduction to Machine Learning Algorithms.
https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47 19.05.2020.

[56] Towards Data Science. - A Top Machine Learning Algorithm Explained: Support Vector Machines (SVMs).
https://towardsdatascience.com/one-of-the-top-machine-learning-algorithms-for-supervised-learning-support-vector-machines-svms-fc45ac0667f4 19.05.2020.

[57] Grey Atom. - Introduction to Decision Trees.
https://medium.com/greyatom/decision-trees-a-simple-way-to-visualize-a-decision-dc506a403aeb 19.05.2020/

[58] Medium. - Decision Tree. It begins here.
https://medium.com/@rishabhjain_22692/decision-trees-it-begins-here-93ff54ef134 19.05.2020/

[59] Towards Data Science. - Understanding Random Forest.
https://towardsdatascience.com/understanding-random-forest-58381e0602d2 19.05.2020/

[60] Towards Data Science. - Understanding Decision Trees (once and for all!).
https://towardsdatascience.com/understanding-decision-trees-once-and-for-all-2d891b1be579 19.05.2020.

[61] Wikipedia, the free encyclopedia - Decision tree learning.
https://en.wikipedia.org/wiki/Decision_tree_learningGini_impurity 19.05.2020.

[62] I. H. Witten, E. Frank, M. A. Hall, *Data Mining Practical Machine Learning Tools and Techniques Third Edition*, Elsevier Inc, 2011.

[63] Towards Data Science. - Entropy: How Decision Trees Make Decisions
https://towardsdatascience.com/entropy-how-decision-trees-make-decisions-2946b9c18c8 19.05.2020.

[64] Wikipedia, the free encyclopedia. - Information gain in decision trees.
https://en.wikipedia.org/wiki/Information_gain_in_decision_trees

[65] Analytics Vidhya. - Powerful Guide to learn Random Forest (with codes in R Python).
https://www.analyticsvidhya.com/blog/2015/09/random-forest-algorithm-multiple-challenges/?utm$_source = blogutm_medium = understandingsupportvectormachinearticle$ 19.05.2020.

[66] Berkley. - Random Forests, Leo Breiman and Adele Cutler.
https://www.stat.berkeley.edu/ breiman/RandomForests/cc$_home.htmooberr$ 19.05.2020.

[67] Wikipedia, the free encyclopedia. - Bootstrap aggregating.
https://en.wikipedia.org/wiki/Bootstrap$_aggregating$ 19.05.2020.

[68] Scikit-learn. - FAQ.
https://scikit-learn.org/stable/faq.html 19.05.2020.

[69] Scikit-learn. - RBF SVM parameters.
https://scikit-learn.org/stable/auto$_examples/svm/plot_rbf_parameters.html$ 19.05.2020.

[70] Scikit-learn. - sklearn.svm.SVC.
https://scikit-learn.org/stable/modules/generated/sklearn.svm.SVC.htmlsklearn-svm-svc 19.05.2020.

[71] Scikit-learn. - Support Vector Machines https://scikit-learn.org/stable/modules/svm.html 19.05.2020.

[72] Scikit-learn. - Glossary of Common Terms and API Elements.
https://scikit-learn.org/stable/glossary.html 19.05.2020.

[73] Scikit-learn. - sklearn.ensemble.RandomForestClassifier.
https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
19.05.2020.

[74] Scikit-learn. - Decision Tree, Minimal Cost-Complexity Pruning.
https://scikit-learn.org/stable/modules/tree.htmlminimal-cost-complexity-pruning
19.05.2020.

[75] Scikit-learn. - Post pruning decision trees with cost complexity pruning.
https://scikit-learn.org/stable/auto$_e xamples/tree/plot_c ost_c omplexity_p runing.html sphx-glr-auto-examples-tree-plot-cost-complexity-pruning-py$ 19.05.2020.

[76] Scikit-learn - Ensamble models, forest.
https://scikit-learn.org/stable/modules/ensemble.htmlforest 19.05.2020.

[77] Medium. - In Depth: Parameter tuning for Random Forest.
https://medium.com/all-things-ai/in-depth-parameter-tuning-for-random-forest-d67bb7e920d 19.05.2020.

[78] Towards Data Science - Optimizing Hyperparameters in Random Forest Classification.
https://towardsdatascience.com/optimizing-hyperparameters-in-random-forest-classification-ec7741f9d3f6 19.05.2020.

[79] Scikit-learn. - Tuning the hyper-parameters of an estimator.
https://scikit-learn.org/stable/modules/grid$_s earch.html$ 19.05.2020.

[80] Scikit-learn. - sklearn.model_selection.GridSearchCV.
https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.GridSearchCV.html$
19.05.2020.

[81] Scikit-learn. - sklearn.model_selection.RandomizedSearchCV https://scikit-learn.org/stable/modules/generated/sklearn.model$_s election.RandomizedSearchCV.html$
19.05.2020.

[82] Scikit-learn. - Cross-validation: evaluating estimator performance.
https://scikit-learn.org/stable/modules/cross$_v alidation.html$ 19.05.2020.

[83] Kaggle. - Challenges in Representation Learning: Facial Expression Recognition Challenge, Data https://www.kaggle.com/c/challenges-in-representation-learning-facial-expression-recognition-challenge/data 19.05.2020.

[84] Lyons, Michael, Kamachi, Miyuki, Gyoba, Jiro. (1998). The Japanese Female Facial Expression (JAFFE) Database [Data set]. Zenodo. http://doi.org/10.5281/zenodo.3451524.

[85] GitHub. - emotion-recognition-with-svm.
https://github.com/phaphuang/emotion-recognition-with-svm 20.05.2020.

[86] GitHub - Facial expression recognition using SVM.
https://github.com/amineHorseman/facial-expression-recognition-svmfacial-expression-recognition-using-svm 20.05.2020.

[87] van Gent, P. (2016). Emotion Recognition Using Facial Landmarks, Python, DLib and OpenCV. A tech blog about fun things with Python and embedded electronics. Retrieved from: http://www.paulvangent.com/2016/08/05/emotion-recognition-using-facial-landmarks/ 20.05.2020.

[88] Dlib. - get_frontal_face_detector().
http://dlib.net/python/index.htmldlib.get$_{f}rontal_{f}ace_{d}etector$ 20.05.2020.

[89] Dlib. - face_detector.py.
http://dlib.net/face$_{d}etector.py.html$ 20.05.2020.

[90] Learn OpenCV - Histogram of Oriented Gradients.
https://www.learnopencv.com/histogram-of-oriented-gradients/ 20.05.2020.

[91] Analytics Vidhya - Feature Engineering for Images: A Valuable Introduction to the HOG Feature Descriptor.
https://www.analyticsvidhya.com/blog/2019/09/feature-engineering-images-introduction-hog-feature-descriptor/ 20.05.2020.

[92] Scikit-image. - Image processing in Python.
https://scikit-image.org 20.05.2020.

[93] Scikit-image. - skimage.feature.hog().
https://scikit-image.org/docs/dev/api/skimage.feature.htmlskimage.feature.hog
20.05.2020.

[94] Numpy. - numpy.memmap.
https://het.as.utexas.edu/HET/Software/Numpy/reference/generated/numpy.memmap.html
20.05.2020.

[95] Raspberry Pi, Forum. - https://www.raspberrypi.org/forums/viewtopic.php?t=252369.
https://www.raspberrypi.org/forums/viewtopic.php?t=252369 20.05.2020.

[96] Medium. - Why you should run a 64 bit OS on your Raspberry Pi4.
https://medium.com/@matteocroce/why-you-should-run-a-64-bit-os-on-your-raspberry-pi4-bd5290d48947 20.05.2020.

# Non-exclusive licence to reproduce thesis and make thesis public

I, Kirill Rodionov,

1. herewith grant the University of Tartu a free permit (non-exclusive licence) to reproduce, for the purpose of preservation, including for adding to the DSpace digital archives until the expiry of the term of copyright,

   **"Embedded system for real-time emotional arousal classification"**,

   supervised by Gholamreza Anbarjafari.

2. I grant the University of Tartu a permit to make the work specified in p. 1 available to the public via the web environment of the University of Tartu, including via the DSpace digital archives, under the Creative Commons licence CC BY NC ND 3.0, which allows, by giving appropriate credit to the author, to reproduce, distribute the work and communicate it to the public, and prohibits the creation of derivative works and any commercial use of the work until the expiry of the term of copyright.

3. I am aware of the fact that the author retains the rights specified in p. 1 and 2.

4. I certify that granting the non-exclusive licence does not infringe other persons' intellectual property rights or rights arising from the personal data protection legislation.

Kirill Rodionov

**02.06.2020**