Instituto Tecnologico y de Estudios Superiores de Monterrey

Monterrey Campus

School of Engineering and Sciences

Front-End Modeling for Emotional State Recognition

A thesis presented by

Oliver Alejandro Velázquez Flores

Submitted to the
School of Engineering and Sciences
in partial fulfillment of the requirements for the degree of

Master of Science

in

Computer Science

Monterrey, Nuevo León, December, 2021

Dedication

Mundo Escondido
Es el lugar de las computadoras
y de las ciencias infalibles.
Ante mis ojos te evaporas
-y creo en las cosas invisibles.
José Emilio Pacheco (1969)

A mis padres, Jaime(†) y mis demás amigos.

Acknowledgements

I would like to thank my fellow colleagues for their time and knowledge in the period we shared, specially because of COVID-19 Pandemic. This work could not be possible without Tecnológico de Monterrey's finantial support as well as CONACyT scholarship along the 23-month-period program.

Front-End Modeling for Emotional State Recognition
by
Oliver Alejandro Velázquez Flores

Abstract

Morphological biometrics have proved to be important contributors to e-security and e-health alike. One of its subdivisions, *behavioral biometrics,* focuses on understanding an individual based on several activities, in this case, a user with drawing and handwritten tasks. Following up the *EMOTHAW* methodology, this work focuses on extracting and generating new features from the original raw attributes of the aforementioned database. These techniques range from signal processing, physics, and statistics from which important feature vectors and models are created. The signal processing techniques focuses on calculating the logarithmic energy of a signal and then generate both the spectral and cepstral domains to create new numerical features for the resulting vector. In the physics department, traditional kinematic variables are calculated from the position of each task, and lastly the statistical features includes momentum, descriptive statistics, and the different means available in the literature (arithmetic, harmonic, etc.). Furthermore, the implementation of a combined dimensionality reduction (PCA) and feature selection (a novel correlation-based filtering algorithm) pipeline methodology is key for the continuous improvement of the results presented in this research thesis project, going from 60.5 - 71.6 % to 100% accuracy on a binary problem, and reaching accuracy of 82.45% in a ternary problem. This was accomplished by generating synthetic observations to compensate the imbalance distribution of classes intrinsic to health databases with added *Gaussian White Noise* to a sample number of real observations. Finally, by implementing a Machine Learning frame of several classification algorithms with the library H2O, a considerable testing efficiency was reached, guarantying the best performance.

List of Figures

List of Tables

Contents

Chapter 1

Introduction

The context of the problem presented in this thesis relates to the psychological field. Emotions are an important part of our lives, they manage how our reaction is going to be in a particular situation, and they are present in how we interact with other people. Whenever sadness, anger, indifference or any other similar feeling are constantly present in an individual, they can be classified as emotional states [97]. These illnesses affect human interactions, the ability to solve problems and cause imbalance in the patient's life [100]. The emotional states that have been linked to chronic diseases are depression, anxiety, and stress[4] [25] [111].

Machine learning techniques can provide effective results in the recognition of emotional states with the implementation of signal processing techniques [53] [66]. In particular, the feature extraction methods used for speech emotion recognition provide good performance [66]. Nonetheless, in the case of handwritten (sensors altogether) data, it is a newly researched field that can use signal processing techniques to identify emotional state with the advantage of being a less invasive process to collect data.

In this thesis work, however, the construction of a new database is not feasible. So a continuation on the EMOTHAW (EMOTion recognition from HAndWriting and DraWing) [73] methodology will be followed with the goals to improve it and find different approaches with the database to propose different, new and better features to classify. This methodology has samples from 129 participants that have been assessed with the Depression-Anxiety-Stress Scale (DAS Scale). All of them required to have a seven-task test (Drawing a house, pentagons, houses, writing words and a sentence) and their principal features are pen on-air and in-paper time, pressure, pen position, azimuth, and altitude, and with the help of Machine Learning models.

A considerable amount of research has been done in recognizing emotions through speech [66][98] but requires signal processing to filter any noise in the signal, The advantages of the handwritten data are that it can be analyzed as any signal, but without the need of filtering it. The EMOTHAW methodology [73] has established is a path to further investigate into the handwritten world.

The solution model will be based on the EMOTHAW work done by Likforman-Sulem, Esposito, Faundez-Zanuy, Clémençon, & Cordasco [73] and on signal processing techniques [66] and to expand the feature selection and generation process in the classifier algorithm.

This work is focused on finding and applying the best feature generation, selection, and dimensionality reduction algorithms available in the literature to enhance the performance in

classification methods; in other words, it goes beyond the scope of this work's objective to find improvement through Machine Learning algorithms, but only through the aforementioned techniques.

The digest version of the methodology would be:

- Constant testing with different feature vectors and their respective comparison between the original results from the EMOTHAW paper [73] or the best results available.
- Identify the behavior of the database features in relation to time and frequency.
- The implementation of signal feature extraction to the database, modifying sample test, primarily.
- Addition of different feature-types, exploring possibilities within the statistical and physics worlds.
- Continuous testing afterwards.

The overall goal of this thesis is to generate a reliable method that could be used in further applications like chronic disease diagnosis and prevention, suicidal tendencies' recognition, and suicide prevention. In other words, in similar databases like EMOTHAW.

In this chapter, the reader can identify different sections that will clarify the research intentions and how the problem is going to be solved. First, the *Problem Definition*, where the obstacles and applications are described. Furthermore, the *Hypothesis* section marks the direction this work will follow, in *Objectives*, which results and accomplishments will be obtained at the end.

In chapter 2, *Reference Review,* there is an in-depth review of different methods, algorithms and proceedings that are found in the literature, programming libraries, and open-source repositories that contributed to this thesis work. In the first section 2.1, *Imbalanced databases and Data Augmentation,* there is a brief explanation on what an imbalance database is and how to solve this distribution problem. Furthermore, in section 2.2, *Signal Processing*, details the spectral and cepstral domain processing that could be drawn from a signal. Later on in sections 2.3 and 2.4 focus on *Feature Extraction* and *Dimensionality Reduction*. They both have important techniques explained. The next section, 2.5 is *Feature Selection,* has several techniques from which this work benefits. The last two sections, 2.6 and 2.7 are oriented towards machine learning: modeling and validation.

Chapter 3, *Front-End Design,* presents the detailed pipeline procedure followed in this work. This process explains the original raw features into each feature extraction methodology, and feature selection technique, followed by the balancing/data augmentation algorithm and finally the Machine Learning classification models and validation tests. Furthermore, all results from the previous chapter are presented in chapter 4 where, there is a detail explanation of the database *EMOTHAW*, including how it was created, by how, and which demography constitutes it. Then in sections 4.2 and 4.3 have result summary tables each, with their own objectives. Lastly, in chapter 5, *Shared Thoughts and Conclusions* there is a recap on the lessons learned and the ideas that work best in the project and also relevant ideas are discussed that could be of used in continuing this research project

1.1 Problem definition and Motivation

It has been proved that there is a strong relationship between handwriting and emotion recognition [73]; "EMOTHAW: A novel database for emotional state recognition from handwriting and drawing" [73] provided an analysis with Random Forest and SVM Classifier algorithms. The problem at hand is that this methodology can be further improved through the implementation of signal feature extraction techniques and different classifier algorithms based upon their performance using the same database.

The EMOTHAW methodology states that the drawing part of the test had a little-to-none impact on the identification of depression in the users [73]. One of the first problems is to determine which classifier algorithm best fits an improved version. Outs such as Random Forests, Bayesian approaches, Support Vector Machines (SVM), and Deep Learning are the first candidates for finding the one that fits best. The next step is the testing process, in which, the mentioned classifiers will be trained and tested with different models (task-oriented, emotion-oriented) and robustness will be added with the help of Signal feature extraction techniques [66].

Handwritten signals have present similar characteristics than traditional signals that are feasible in recognizing emotions, but one clear difference and advantage is that handwriting is a non-invasive approach to acquire reliable data that can be easily linked to diseases like Parkinson and Alzheimer [73]. Signal techniques, procedures, and methodologies can be applied to a handwritten analysis, one of the main differences, however, is the adjustment in the sampling rate each technique requires to work properly. A handwriting signals are bounded to the amount of traces the individual requires to fulfill a task (either write a sentence or draw a house), whereas a voice signal have a sampling rate of 8 kHz for a telephone [117].

One important aspect to consider is that the depression emotion is detectable to drawing aspects rather than handwriting, when both features appeared in the classifier's results, handwriting does not provide enough information to be considered in the depression detection, this could be something to put more attention to when comparing results, if graphology modifies this results then it will be a confirmation that the original method was not accurate.

Although, studies, for example [4] [25] [111]., link these emotions with chronic disease generation, there is not a clear way if there is a shorter path between handwriting and these diseases, but there is enough proof with the references that there is the possibility to identify such diseases and the method at hand.

All the research regarding emotion recognition work with a specific database that was either build from scratch to work on it or the methodology adapted to an already existing one. If graphology proofs to be a relevant technique for the feature selection, there is a possibility that those databases can be as simple as a sentence, and the information required can be extracted from medical institution forms or even any work done by handwriting.

This method can be used by government and medical institutions, schools or any other organizations that gather information through handwriting. One concern, though, is to build the generalized database, because there will be need to acquire specialized tablets, such as the INTUOS WACOM series 4 used in the EMOTHAW research [73], other outcome could be the advances in the tablet industry that will fit the applications of this method in the future.

1.2 Hypothesis and Research Questions

Diversified feature vectors drawn from written activities have a significant impact on Machine Learning Frameworks' performance.

This proposed work aims to successfully answer the following research questions:

- Can signal processing techniques help in creating improved features in this database?
- Does the current database have enough information to work with? If not, how to successfully broaden it?

1.3 Objectives (General and Particular)

The general objective of this thesis is to develop a feature extraction pipeline that yields a higher emotional state identification accuracy than other models, using voice signal processing techniques and dimensionality reduction algorithms, enabling its applicability in suicidal tendencies' detection, chronic disease prediction, and biosecurity.

The goals to be achieved are:

- First goal: Evaluate the feasibility of feature extraction techniques available in the literature for handwritten databases.
- Second goal: Evaluate the relationship between tasks, emotions and features and how they impact the overall results.
- Third goal: Have improved results in comparison to the EMOTHAW methodology [73], with the implementation of the mentioned techniques.
- Fourth goal: Define the methodology for online handwritten data with signal feature extraction techniques.

Chapter 2

Reference Review

2.1 Imbalanced databases and Data Augmentation

As far as databases are concern, there are mainly two types of data structures when classifying them based on the number of instances in each class: balance and imbalance databases. It is said that a database is balance, if all the classes it has had the same number of instances [6] or at least the difference in-between classes is not significant. In figure 2.1 we can see the distribution of balanced 4-class database.

It is not always the case where the data we look or gather in a real-life problem will have an equal distribution. In figures 4.9 and 4.10 it can be seen how the classes are distributed and that there are important implications off of it: generalization is a clear problem when training a Machine Learning or Statistical problem.

An imbalance database consists of one or more class(es) that have some instances or examples, also called outliers, to a given population. There are, commonly, two types of Under sampling in a given population: when there is not enough representation of the predicting variables (attributes), and the uneven distribution of dependent variables (classes) in a data set [101]. The latter is the case for the medical and psychological fields, because it is more frequent that people do not present a given disease or disorder [99], in other words, it is more common to have healthy people than disease people.

In figures 2.2 and 2.3 we can observe further examples of imbalance class distribution, specially for binary and ternary classification, respectively.

Figure 2.1: Synthetic example of a balance database's class distribution

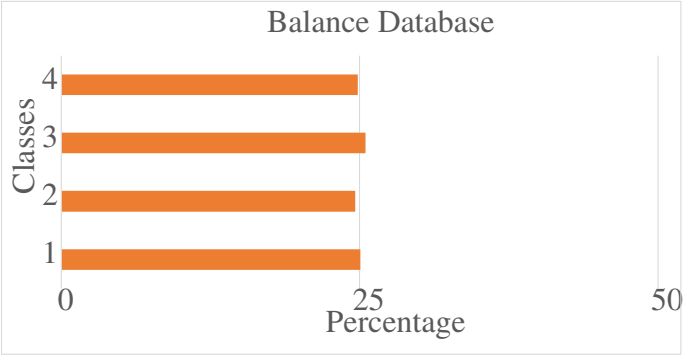


Figure 2.2: Example of Binary imbalanced distribution

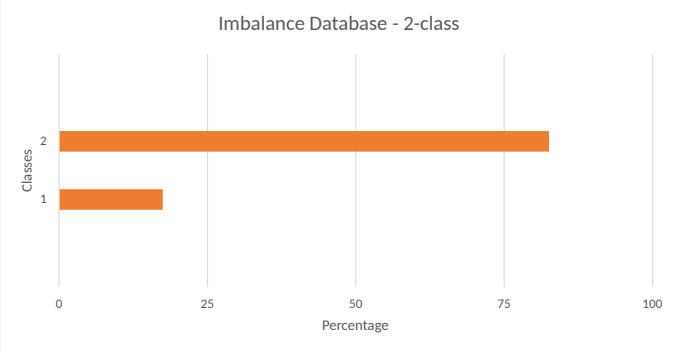
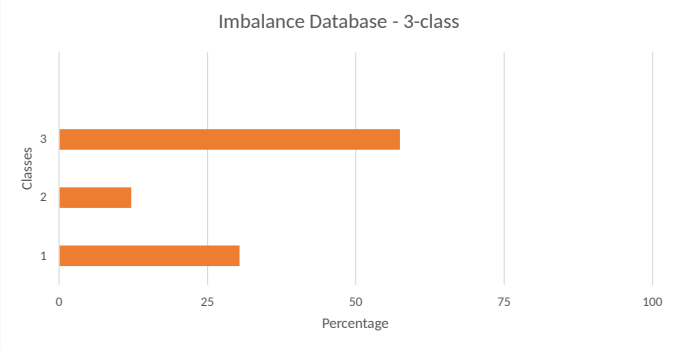


Figure 2.3: Example of Ternary imbalanced distribution



Even though imbalance classes are a problem for generalization, there are several techniques that solve this problem [51]: synthetic generation of data (e.g., Under sampling, Oversampling), expanding the database creation methodology to more similar population, restructuring the database construction, and others.

2.1.1 Under sampling and Oversampling

Under sampling and Oversampling are traditional data augmentation techniques in classification problems, specially in imbalanced databases [81]. These proceedings are the first attempt on solving the imbalance problem and because of this, they have a lot of shortcomings. They are generated based on the distribution of the classes, adjusting enough to create new synthetic samples to compensate the imbalance issue [121].

Oversampling is more complicated to execute than *Under sampling*, because the latter can only require to remove samples from the most-represented class in a given data set and usually requires more human effort to perform, whereas the former is only possible by applying statistical, signal processing, or other similar technique to the data at hand [121]. This said, it does not mean that Under sampling cannot use the aforementioned techniques, is that it does not require to perform them. Their main application is in Statistical and Machine Learning applications [81].

Under sampling without a clear plan or knowledge about the data at hand can represent a considerable problem, removing data will cause the data set to transform into an even different

structure and might not be useful anymore to the original problem [121], this is because of *statistical confidence*. Nonetheless, in the popularization of the *Big Data* culture and industry, it might be feasible to consider [110].

The available synthetic methods for sampling traditional data are implemented in open-source libraries such as SK-learn, one of which is the *Synthetic Minority Over-sampling Technique* (SMOTE) [22]. These set of algorithms let the user over and under sample, as well as creating random set of data for a given distributed data. The underlying methodology is to sample the data and consider its nearest neighbors to obtain a hypothetical data point to then calculate the vector between this new data point to a neighbor and multiply it by a random number between [0,1], then it adds the resulting value to the current real point of data to create a synthetic observation [22].

2.1.2 Imaging: Transformation, synthetic, etc.

Data augmentation is fairly common in imaging processing techniques, specially in neural networks (e.g., 2D-Deep Learning) [103]. Images are a type of unstructured data that allow a certain ease to augment the data pool, because a given system will consider two different images even though there are the same but mirrored [12].

The principal reason why data augmentation is popular and works well with images is because obtaining new real data is not efficient, i.e., that it takes considerable effort and time to obtain new imaging of the given application the models are built for [103]. Synthetic imaging is good enough to have a certain variety of data without sacrificing the possibility of over or under fitting the model.

The most popular data augmentation techniques for images in deep learning applications are (consider that all these techniques take one image in particular and apply these techniques to them): rotation, cropping, mirroring, color modification, added noise, among others [103].

2.1.3 Noise Sampling

In statistics, a data set is considered noisy when it contains added unintended information, usually corrupted that, if anything, invalidates the data as a whole [34], but when this concept is used appropriately it can be a useful as a data augmentation technique for traditional data [105].

It is a mathematically constructed noise that constitutes zero mean, constant variance and no serial correlation, which allows for enough synthetic data to compensate the imbalance problem and not change the nature of the overall data set. More information in section 3.4 in chapter 3.

2.2 Signal Processing

Signal Processing has its own utility in physical, electronic, and electric applications, but in computer science, specially for Sensors Data applications, it can be used as a Feature extraction technique that works with the characteristics of the signal-like features presented in medical and psychological databases [67], including images and sound.

Signals are information emitted from a source that are usually mathematical modeled for analysis. This so-called source can be mechanical, optical, magnetic, electrical, electronic, discrete, or any other physical or digital nature [118]. Overall, there are three different types of signals: *analog*, *digital*, and *discrete*.

Analog signals are the continuous representation of a variable that depends on time (also continuous), usually electric, audio, and any other signal that is generated from older devices. This type of signal uses a physical mean to carry on information; in electrical signals, for instance, amplitude (voltage), current, and frequency are parameters that change over time [71]. Due to the physical mean in which they are present, they have inherited noise within the information they carry. In other words, noise can be understood as the loss of information within a signal [16]. Analog signal processing techniques include time, frequency, and time-frequency domain.

Discrete signals are another type of signal in the literature. One of the most important characteristic is that now it has one discrete variable that conforms the signal, usually the independent variable is the one that is discrete. It can be obtained from an analog signal by sampling it with a fixed frequency called *sampling frequency*. The dependent variable remains as a continuous magnitude [90]. It can be seen as the bridge between analog and digital signals. Its processing is reliant on electronic apparatus in its core, but can be simulated by computing.

The third classification is digital signals. They are similar to discrete signals, but both the independent and dependent variable are discontinuous. This process further discretize an already discrete signal with mostly digital signal processors or computers [93]. Filtering and signal transformations (e.g. fast Fourier Transform) are algorithms typical in digital signals [87].

2.2.1 Signal Energy

In signal processing, energy of a signal, denoted E_s, is the measurement of a signal for "strength" or the amount of information, electricity or any other physical scale [58]. The first approach on how to quantify it in a given signal was to calculate the area under the curve, with possible negative and positive values throughout, but this represented a strong problem given that there would be cases where the result would be zero. Nonetheless, to fix this problem, the calculation of an energy's signal is:

$$E_s(t) = \int_{-\infty}^{\infty} |s(t)|^2 dt \tag{2.1}$$

By elevating the signal $s(t)$ by the power 2, the problem is solved. Furthermore, it can be used on continuous and discrete signals alike.

2.2.2 Spectral Density

Furthermore, the energy of a signal can be mapped with frequency to understand the distribution of the latter component in the time series. Usually, this approach is used in finite signals (pulses, digital, etc.), and from 2.1 can be defined as [10]:

$$S(f) = \int_{-\infty}^{\infty} | \int_{-\infty}^{\infty} e^{-i2\pi ft} s(t)dt|^2 df \tag{2.2}$$

We can simplify equation 2.2 in notation by calling the inner integral as $X(f)$:

$$S(f) = \int_{-\infty}^{\infty} |X(f)|^2 df \tag{2.3}$$

Equation 2.3 is considered a density function and $X(f)$ the *Fourier transform* [78]; it can also be applied to discrete signals by means of the *Discrete Fourier Transform* (DFT). Rewriting equation 2.3 we have:

$$S_d(f) = \sum_{n=0}^{N-1} s(n) e^{-\frac{2\pi i}{N} kn} \tag{2.4}$$

Similarly to equation 2.2, energy can be calculated by applying the logarithmic operation to the DFT of a signal (eq. 2.4) [86]:

$$E_d(f) = \log_2 |S_d(f)|^2 \tag{2.5}$$

$E_d(f)$ is called the *log energy spectrum* or *spectrum* for short, and it helps on identifying better the energy distribution within the signal in terms of frequency: In the first measures of the aforementioned signal will contain most of its energy [86]. The signal $S_d(f)$ could also be divided by two, since it usually contains the same information twice. It also helps set up for a further step called Quefrency Analysis, which will be explained in the next section (2.2.3).

2.2.3 Quefrency Analysis

The words *Quefrency* and *Cepstrum* derive from *frequency* and *spectrum*, for the former is just a change in the first and second set of three letters and for the latter, it is a permutation of the first four words. This summarizes the connection between spectrum and cepstrum. The latter is obtained from the former by applying the Discrete Fourier Transform (i.e., the spectrum of a spectrum), and from equation 2.5 it can follow [86]:

$$C_d(f) = \mathcal{F}(E_d(f)) = \sum_{n=0}^{N-1} E_d(n) e^{-\frac{2\pi i}{N} kn} \tag{2.6}$$

This type of analysis has been used in signal processing since its conception, and has led to important discoveries and applications ever since [89]. *Cepstral Coefficients* can be obtained from the Cepstrum, by applying a set of filters (i.e., filterbank) of a given sampling rate f_s, filter bandwidth bwf, a number of linear filters Q and a filterbank bandwidth $FBBW$ [86]. This methodology helps in the creation of numerical features for time series of the nature of the *EMOTHAW* Database, see more in section 4.1.

2.3 Feature Extraction

Feature Extraction (FE) is a Machine Learning Technique that focuses on derive or find newer attributes within the raw information offered in the data set of a given problem [61]. It is a technique related to Feature Selection (Section 2.5) and Dimensionality Reduction (Section 2.4) when FE is used as a method of explaining a large data set with the minimum information possible.

One of the most used techniques is to combine already existing attributes into a new one by multiplying, summing, or perform some statistical measure for them [48]. Another approach is to use Principal Component Analysis (any of its variance: Multilinear PCA, Kernel PCA), Isomap, Independent Component Analysis, or any other Dimensionality Reduction Technique [14]. Nonetheless, FE is still part of a bigger chain of action for model generalization and better understanding of Machine Learning in general, but it is a key aspect of it and experts argue that its optimization might be the most important part of the process [112]. This process is also called *Feature Engineering*, where the goal is to conceptualize a feature vector that maximizes the explanation of the original data set within it [112].

The set of new features can be called *feature vector* and the complete set of features (new and old) *Front-End* features [85]. The type of features that would be looked in this work are: Time (Section 2.3.1) [73], Kinematic (Section 2.3.2) [56], Statistical (Section 2.3.3) [33], Spectral (Section 2.2.2) [85] [86], and Cepstral features (Section 2.2.3) [85] [86]. All of them add important information to the resulting feature vector.

2.3.1 Time features

Time Features are related to several raw attributes from the EMOTHAW data set (see Section 4.1.2), specifically timestamp and On/Off Status. In the EMOTHAW work [73], they extracted 4 new features called timing-based and ductus-based features. which are:

1. F_1: Amount of time that the pen stayed on air during task ($\sum_{h=1}^{H-1} d_{2h}$),
2. F_2: Amount of time that the pen was over the paper during task ($\sum_{h=0}^{H} d_{2h+1}$),
3. T: Time to complete the task ($F_1 + F_2$),
4. NSt: Number of strokes (ductus-based feature).

One thing to note from F_1 and F_2 is that d_i is the duration of the i-th stroke, and by calculating $i\%2$ it could be determined whether the stroke was on-air (No residue) or in-paper (otherwise). The number of strokes H can be defined as a set of strokes A, where $A = \{St^1, St^2, ..., St^H\}$, which in-paper and on-air strokes are alternated in A, starting and finishing with paper strokes (i.e., St^1 and St^H) [86].

They might seem insufficient features, but there are seven tasks to draw them from, summing for potential 28 new features. Nonetheless, they decided to not use the circle-drawing tasks because of the lack of pen status in them [73]. In the end, they work with 20 features total.

The formal definition of the time feature vector (TF) is [73]:

$$TF = \{F_1, F_2, T, NSt\} \tag{2.7}$$

2.3.2 Kinematics

Kinematic Features refers to the physical characteristics of an object in movement (i.e., velocity, position, acceleration, and so on) without considering the force or outside phenomenon that caused the object to move [15]. Even though it's a physics concept in itself, when the initial conditions of a problem satisfy the criteria of Kinematics, then it can be applied to the problem regardless of its nature, including feature generation/extraction [56].

The complete *Kinematic Feature* and *kinematic signal* vectors has a considerable amount of new attributes that help take advantage of the nature of positional-type features [85]. These new characteristics can be defined as follows:

1. Discrete displacement: $d_{x,y}(n) = \sqrt{x(n)^2 + y(n)^2}$,
2. The set of displacements: $w(n) = \{d_{x,y},\ x(n),\ y(n)\}$,
3. Velocity: $v_w(n) = \frac{w(n) - w(n-1)}{T_s}$,
4. Acceleration: $a_w(n) = \frac{v(n) - v(n-1)}{T_s}$,
5. Jerk: $j_w(n) = \frac{a(n) - a(n-1)}{T_s}$,
6. Trajectory of pen divided by duration of task: S,
7. Changes in Velocity: $NCV = \frac{1}{(n-1)} \sum_{i=1}^{N-1} |v(i) - v(i+1)|$ (local extrema velocity),
8. Change in Acceleration: $NCA = \frac{1}{(n-2)} \sum_{i=1}^{N-2} |a(i) - a(i+1)|$ (acceleration's local extrema),
9. Relative velocity to writing duration: $NCV_r = \frac{NCV}{T-1}$,
10. Relative acceleration to writing duration: $NCA_r = \frac{NCA}{(T-2*T_s)}$, where T_s is the sampling time.

These features draw important information about the position of a pen in a problem that depends upon the data acquisition through a tablet. Furthermore, these features can be used to further create new features (see Section 2.3.3).

2.3.3 Statistical Analysis

Statistical features are one of the most common type of features there is in Data Analytics and Machine Learning, specially in the Feature Extraction department [33], because they can summarize important information of numerical attributes in a data set. There is a wide number of different statistical parameters or measurements that could be calculated from an attribute, but not all of them are best fitted to them; this is because all data is not the same.

Descriptive statistics provides measurements that do not go in-depth, but show relevant information of the attributes at hand. Some of them are: mean (average), median, mode, standard deviation. Other type of statistical features are the so-called *mean* measurements, e.g., the arithmetic mean (which is the average), geometric mean, harmonic, and the truncated

mean. This last one is a derivation of the average, where it removes the lowest and highest value from the calculation and proceeds normally with the rest of the data. From there, quartiles, percentiles, kurtosis, and momentum are also popular among statistical features [1] [85].

From this, it could be said that there are three important statistical features [85]:

1. Descriptive Statistics group: $\mathcal{D}_{\mathfrak{g}}^{\tau,u} = \{\overleftrightarrow{\mathfrak{g}}^{\tau,u}, \breve{\mathfrak{g}}^{\tau,u}, \ddot{\mathfrak{g}}^{\tau,u}, \dddot{\mathfrak{g}}^{\tau,u}, \widetilde{\mathfrak{g}}^{\tau,u}\}$

   Where,

   - $\overleftrightarrow{\mathfrak{g}}^{\tau,u}$ is the range of the attribute,
   - $\breve{\mathfrak{g}}^{\tau,u}$ is the median,
   - $\ddot{\mathfrak{g}}^{\tau,u}$ is the mode,
   - $\dddot{\mathfrak{g}}^{\tau,u}$, is the standard deviation,
   - $\widetilde{\mathfrak{g}}^{\tau,u}$ is the percentile $99^{th} - 1^{st}$.

2. Mean Statistics group: $\mathfrak{M}_{\mathfrak{g}}^{\tau,u} = \{\overline{\mathfrak{g}}^{\tau,u}, \overline{\overline{\mathfrak{g}}}^{\tau,u}, \overbrace{\mathfrak{g}^{\tau,u}}^{\text{tri}}\}$

   Where:

   - $\overline{\mathfrak{g}}^{\tau,u}$ is the arithmetic mean,
   - $\overline{\overline{\mathfrak{g}}}^{\tau,u}$ is the geometric mean,
   - $\overbrace{\mathfrak{g}^{\tau,u}}^{\text{tri}}$ is the truncated mean.

3. Momentum Statistics group: $\mathcal{M}_{\mathfrak{g}}^{\tau,u} = \{\overbrace{\mathfrak{g}^{\tau,u}}^{\text{qua}}, \overbrace{\mathfrak{g}^{\tau,u}}^{\text{per}}, \overbrace{\mathfrak{g}^{\tau,u}}^{\text{mom}}, \mathfrak{h}^{\tau,u}\}$

   Where:

   - $\overbrace{\mathfrak{g}^{\tau,u}}^{\text{qua}}$ is the Quartile features ($Q_1 = 75$ and $Q_3 = 25$),
   - $\overbrace{\mathfrak{g}^{\tau,u}}^{\text{per}}$ is the Percentile features (1, 5, 10, 20, 30, 90, 95, 100),
   - $\overbrace{\mathfrak{g}^{\tau,u}}^{\text{mom}}$ is the Momentum features (3^{th}, 4^{th}, 5^{th}, 6^{th}).
   - $\mathfrak{h}^{\tau,u}$ is the Kurtosis.

Statistical features, alongside Time, Kinematic, Spectral, and Cepstral, compose a wide feature vector that has a considerable amount of information drawn from a given raw data set [85].

2.4 Dimensionality Reduction

Today's data is increasing in complexity and size, specially unstructured data. One of its characteristics is that it has considerably high data dimensions (e.g.: global climate data, human genes, signal processing data, neuroinformatics, among others), and it demands an important computational time to be process and appropriately used for modeling or predicting [114]. Dimensionality reduction intends to preserve the meaningful information of a high-dimensional database without having all the attributes or instances of the high-dimensional vector space. This task helps in reducing computational, classification and training time, visualization of the patterns and overall data, cluster analysis, among other applications [114]. It could also be used as a pre-processing step in another process, such as preparation for machine learning models.

There are two types of dimensionality reduction techniques: *linear* and *nonlinear*. The usage of either of them depends upon the characteristics of the data that one is working with. Traditional applications usually work best with linear algorithms and complex applications (i.e.: human behavior, handwriting, social phenomenon, natural observations, etc.) with non-linear techniques [114]. It is also worth mentioning that dimensionality reduction techniques could also be used as part of other feature-oriented analyses, such as feature extraction and selection, both of which are explored in sections 2.3 and 2.5, respectively.

One of the main inconvenience with high-dimensional data and dimensionality reduction altogether is the *curse of dimensionality* problem which states that, for databases/problems with increasing dimensions, significant information found within a given space in the overall database requires considerable amounts of data the higher the dimension of the problem [57]. Furthermore, the way data is sparse over these spaces is not constant, and it is bounded to the nature of the problem itself, making efficient search algorithms complicated to design.

As mentioned before, Dimensionality Reduction can also be used as part of another process within a workload pipeline, specially in conjunction with Feature Extraction [102]. Within the methods that transform a given vector space (in this case, the high-dimensional database) into a different one constructed according to some methodology, *Principal Component Analysis* is a common algorithm paired with and even used as a Dimensionality Reduction technique [19].

2.4.1 Principal Component Analysis

Principal Component Analysis (PCA) is a traditional linear method used to reduce complexity in high-dimensional databases, exploratory data analysis or for predictive models. It can also be used as a pre-processing algorithm for machine learning classification models [91]. PCA works by maintaining the variance as much as possible of a given database, but by transforming the feature space into orthogonal components. In other words, is a projection into a new feature space that explains the variance of the entire database without its entire dimensionality (i.e., the principal component) [91]. Furthermore, the other components are then orthogonally position from the principal component for maximization of variance.

The first thing in the process is to calculate the Principal component (PC) (The first component and the one that holds maximum variance), and then the rest, where they lose variance in comparison to the original data set. This is possible by centering the original

instances by making each column's mean 0 [2]. In essence, PCA methods are similar up until this point, there are two types of analyses: *Correlation* and *Covariance*; The former requires the data to be normalized, and the latter to divide the centralized data by the square root of the observations of the problem.

The number of components of the resulting feature vector space is bounded to the original size of the data set [2]. Let's say that the original database had N observations and M original features, then the number of maximum components will be $min(N, M)$ and sometimes $min(N, M) - 1$ (depends on software used). Nonetheless, having all the PCs in the resulting model will not ensure good performance, most of them contain no significant information overall. Python's SciKit-Learn library offers the option of calculating components from a significance threshold (e.g., if n_components is set to 99%, then the package will only calculate the PCs that have said significance) [92].

2.4.2 Independent Component Analysis

Independent Component Analysis (ICA) is a statistical technique that helps reconstruct or take apart a signal into its whole or components, respectively [55]; because of the nature of the algorithm, it is also used in signal processing techniques to filter out noise and audio [80].

This technique decomposes a given signal (i.e., continuous data, audio, image, video, etc.) into a multisource independent signals that are statistically independent of each other. It is not an easy task to do, but there are two different heuristics that fulfill these conditions: *Minimizing mutual information* or *Maximizing non-Gaussianity* [55]. The latter means not having a normal distribution [24].

There is not a clear or correct way to guarantee that the multivariate results will be optimal, but iterating through n-times can compensate for that. Although, doing so will demand considerable computational cost [19]. Nonetheless, three important pre-processing steps help this burden: *zero mean centering*, *whitening*, which is the transformation of a set of data with known covariance matrix into a new set but with the identity matrix as its covariance [60], and reduce dimensionality [55].

The mathematical representation of ICA is by having a vector $\mathbf{x} = (x_1, x_2, ..., x_m)^\top$ and hidden components $\mathbf{s} = (s_1, s_2, ...s_m)^\top$ so that, in its general form:

$$\mathbf{s} = \mathbf{W}\mathbf{x} \tag{2.8}$$

Where $\mathbf{W}$ is the linear transformation that maximizes independence.

2.4.3 Locally Linear Embedding

Another important dimensionality reduction technique is the *Locally Linear Embedding* model. It is an unsupervised learning algorithm that focuses on preserving the embeddings of the high dimensional data set while also reducing dimensionality [95]. It started as an idea that came from speech signals and images in particular, where usually their content is compressed and reduced without substantially taking information from them. It also came from the comparison of other linear dimensionality reduction methods, in particular Principal Component Analysis (PCA) and Multidimensional Scaling (MDS), which do well enough to be the most popular algorithms, even though they have important nonlinear constrictions [95].

Locally Linear Embedding is said to be by the authors a similar algorithm to PCA and MDS, in the usage of eigenvectors in the projections of data to reduce dimensions, but it successfully maps high-dimension points in a fixed coordinate system by preserving as much as possible the relationship among points in the data (creating locally linear areas of data and nearest neighbor distancing) [95]; mathematically is defined as:

$$\varepsilon(W) = \sum_i |\vec{X}_i - \sum_j W_{ij}\vec{X}_j|^2 \tag{2.9}$$

Where W are the weights, which is a numerical mapping of the relationship between the point j and its reconstruction j and the overall equation is the cost function. This equation is based upon two conditions: the first is that the point $\vec{X}_i$ is only constructed by its nearest neighbors, and that $\sum_i W_{ij} = 1$. It follows to solve a least square problem for finding the optimal weights W_{ij} [95].

2.4.4 Isometric Mapping

Another non-linear embedding-based algorithm is *Isometric Mapping* (Isomap), which is similar to multidimensional scaling (MDS) but uses a different distance metric. MDS traditionally uses Euclidean distance, whereas Isomap uses and calculates from a weighted graph the geodesic distance metric [83].

The algorithm has three different set of steps which go from calculating nearest neighbor into shortest-path calculation and finishes with eigenvalue decomposition:

1. Nearest neighbors

2. Shortest-path search: Uses the *Dijkstra's Algorithm* (any other one could be used), but the implementation in sklearn can decide which algorithm is best for the specific input data [92].

3. Partial eigenvalue decomposition: in N points of data and given a $N \times N$ grid/kernel, it calculates the largest eigenvalues in the eigenvectors of the embedding.

2.5 Feature Selection

Feature Selection techniques have encounter complications in newer databases, because their dimensionality has increase considerably, and the traditional algorithms do not meet the computational efficiency needed in today's applications [122]. Nonetheless, their benefits still holds in reducing dimensionality, reducing unwanted noise when generalizing machine learning models, they could be used also to reduce training time, reduce dimensionality, learning accuracy, among other reasons [96] [122]. Based on given criteria, they sort of all the different attributes and remove those that did not made the cut [30]. It could be, for instance, a correlation-based feature selection technique that will only select attributes with 0.4 correlation factor or less in-between characteristics.

There are three main different groups of Feature Selection methods: *filters*, *wrappers*, and *embedded* [47]. They are heavy influenced by the evaluation metrics used for understanding how good an algorithm is.

Wrapping algorithms are computationally expensive, but are the best fitted for a given application or model, because they subdivide the database into subgroups and each is then trained, and the error rate metric will tell which groups produced the better model overall [47].

Embedded methods are not algorithms on their own, but they are part of the model process. They are particularly efficient because they are not constantly splitting data into several groups, they only take advantage of the actual process and the metrics to evaluate if a given attribute is relevant or not [47]. The best example is the *LASSO* algorithm in regression methods that penalizes coefficients with the L1 metric [109]. Many other algorithms are based upon making a better version of the LASSO method.

In the other hand, *filter algorithms* are less computational expensive than wrapping's, because they are not fitted towards a specific use (they still hold relevant information, nonetheless). There are different types of measures for filters (e.g.: correlation-based, mutual information, distance-types), which allows reaching the point of a somewhat good subset of features and good training computational time [47]. In section 2.5.3 the filtering algorithm used in this work can be found.

2.5.1 Variance Threshold

Variance Threshold is one of the most simple techniques for feature selection. As the name implies, the user sets a given threshold and removes all attributes that do not comply with said variance limit. This includes the variables with the same value across all observations [47].

The mathematical expression is:

$$Var[X] = \mathbf{p}(1 - \mathbf{p}) \tag{2.10}$$

Which is the Bernoulli random variable variance expression.

Now, this being a so-called simple technique does not say it is not useful; there are problems that could benefit from this algorithm, some databases are simple enough to perform it to them, which will save computational runtime. The parameter of the variance threshold needs to be tune and performance can be affected because of a wrong threshold value. In usage with other algorithms, Variance Threshold can improve the results of this new model of feature selection [47].

2.5.2 Relief-Based Algorithms

As mentioned in the Feature Selection introduction (2.5), there are three different types of feature selection algorithms: wrapper, filter, and embedded methods. *Relief-Based* algorithms belong in the filter methods category, which takes advantages of feature interactions [63].

In general, Relief uses pair-nearest neighbors approach to calculate a score for each feature in a data set, to then rank all the scores and establish a threshold to then remove the features that do not comply with the conditions [63]. This methodology can also be used as

a pre-processing step in a larger data pipeline. The advantages of these types of algorithms is that they do not require any heuristic to work, they are cost-efficient (computationally), and noise tolerant, but the issues with it is that data sets with low observations will not make the algorithm to work appropriately [63].

Kira and Rendell proposed the first relief algorithm in 1992, although since then other relief-based algorithms have been created: ReliefF, RRELIEFF, Relieved-F and others, but they work fundamentally the same, they just add new applications (i.e., incomplete data) or prepare it for different types of problems (e.g., regression, multi-class, etc.) [63].

The fundamental proceeding is: let's assume there is a data set D with n observations, f features, and a feature weight zero vector W with a binary target variable. All numerical data should be scaled between [0, 1]. In an iterative process, the vector W will be updated from the feature vector F of each observation from the data set, so that [63]:

$$W_i = W_i - (f_i - nearHit_i)^2 + (f_i - nearMiss_i)^2 \tag{2.11}$$

Where $nearHit_i$s are the closest same class observation from f_i and $nearMiss_i$s are the closest different class observation.

2.5.3 Fast Correlation-Based Filtering

As mentioned in Section 2.5, attaining considerable computational efficiency in high dimensional databases, and Fast Correlation-Based Filtering (FCBF) is a filtering algorithm based on correlational analysis that effectively reduces a given database complexity but without sacrificing important information for modeling [122]; it also takes care of redundancy of attributes.

An attribute on its own has no connotation. It requires context to be classified as *good* or *bad* feature. In particular, it can be said that an attribute is important if it holds relationship to the output (class, target variable, etc.) and that it does not have it with other inputs (attributes, dependent variables, etc.). This relationship is called *correlation* [122].

There are two ways to calculate correlation between variables: the *Linear Correlation Coefficient*, and *Information Theory*, both have their advantages and disadvantages, but Linear Correlation Coefficient disadvantages outweigh the Information Theory advantages [122]. This is due to two important concepts, *information gain* and *symmetrical uncertainty*. The former holds that when the entropy of a variable X decreases because of the effect of another variable Y, X now reflects more information because of Y [122]. Nonetheless, this measure is considerable bias to variables with more values; Symmetrical Uncertainty fixes this issue by normalizing the value range from [0,1], where 0 indicates that the variables are uncorrelated and 1 is perfectly correlated.

The algorithm starts by selecting the most relevant features with the threshold γ, and they are sorted by Symmetric Uncertainty. The next part consists of three different heuristics so that it removes redundant attributes. The first heuristic is that if a feature F is considered predominant, it could be considered to be a threshold for the lower ranked features. Heuristic two removes redundant attributes if they are redundant peers to F, and lastly, the third heuristic established the higher ranked Symmetric Uncertainty feature to be always consider the predominant attribute [122].

This methodology proved to be efficient and effective in creating relevant feature spaces without sacrificing computational resources.

2.6 Training-Testing-Validation Sets

2.6.1 Cross-Validation

In Machine Learning and Statistics, the overfitting or under fitting is fairly common among models in both fields, specially when there is a lack of process design and data preparation [123]. One clear step that helps solve these problems is to validate the model with a set of data that resembles the one used for training and testing to evaluate actual performance in-never-seen-before observations [106]. There are several methods and techniques in the literature, but the one that has proved to be one of the more reliable is *Cross-Validation*.

Is a resampling method, in which the goal is to verify that the model at hand is not overfitted to the train set and does not show any bias selection, which both are important issues to solve before formally deploy a new model for solving a given problem [106]. This is achieved by segmenting the original data set into subgroups: training set, test set, and sometimes a third, called validation set. The overall idea is to use the training set to let the model learn, test its generalizations with the test set (first seen data), and if the user decided to have a validation set, it could work as new data for the model and further verify its performance [106].

The most common form of Cross-Validation is the K-fold methodology, which follows the partition process into K subsets along different rounds of training/testing; meaning that, the partitions are made at the beginning of each round, and at the end of all training and testing, the performance results are usually averaged to reach a better model that avoids overfitting [3].

There are two types of Cross-Validation techniques: *Exhaustive* and *Non-exhaustive* Cross-Validation. The latter type does not require testing on every possible way the data could be split, instead it only focuses on a given fixed partition, whereas the former does train on every possible ways of splitting the data. *Holdout Methods* are non-exhaustive and *leave-one-out* are exhaustive [3].

2.6.2 Holdout Method

Holdout Method is a non-exhaustive cross-validation type, where the data is split into two different groups, one is the test set and the other the training set, in which a model is going to be created [120]. The main difference between a k-fold cross-validation is that Holdout does not average the results on the splits, because it does not have multiple subsets of data. That is the reason why Holdout is usually the simplest and usually the first approach for a validation technique [35].

The lack of multiple runs might lead to inconsistencies in the model's results, including any performance metric used to evaluate the model, after all, it is the simplest cross-validation model there is [120].

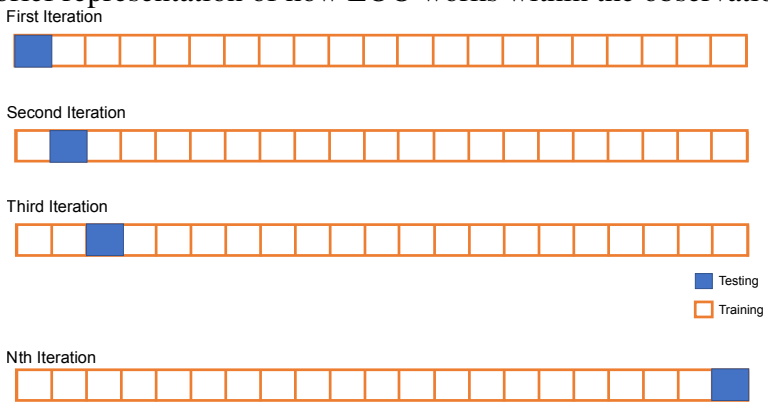
2.6.3 Leave-one-out

Leave-one-out (LOO) is a particular case of Training/testing exhaustive cross-validation where, it will only consider one observation for testing and the rest for training, then this process is repeated N times, N being the total number of observations within the data set [116].

It can be easily explained graphically. In figure 2.4 we can observe how in each iteration the algorithm only takes into account a piece of information (in this case, a square represents an observation, or row vector) and in the next moves to a new one. The blue squares represent the subset that will be used for testing. The lined-orange squares are the rest of observations used for training. At the end of the Nth iteration, the metrics used for the model are averaged, and it is the overall resulting performance of the model [86].

The computational time of this approach may vary depending on the number of observations in the data set, usually is more efficient than Leave-Percentage-Out or any other k-Fold Cross-Validation; Nonetheless, it might not be the case and K-fold could run faster [82].

Figure 2.4: A brief representation of how LOO works within the observations of a data set



2.7 Auto-Machine Learning

Machine Learning is constantly being used in different types of businesses and researches as its tools become widely available and significant easier to implement. From one of these ideas comes *Auto-Machine Learning* (AutoML), which integrates different machine learning models without the time-consuming tasks that most models require working efficiently (Hyper and parameter tuning). The AutoML alternatives provide an automated parameter tuning based on Bayesian optimization, genetic programming, or Random Search [44]. It is also efficient, and guarantees productivity off of the AutoML implementations [69].

There are several implementations of AutoML in the literature, the five more used are: Auto-WEKA [108], Auto-sklearn [39], TPOT [88], AutoGluon [36], and H2O [69]. All of them provide a platform for experienced data scientists, researchers and newcomers in any given field or career path. Auto-WEKA and Auto-sklearn work with Bayesian optimization for hyperparameter tuning, TPOT with genetic programming, and AutoGluon and H2O with Random Search. The OpenML AutoML benchmark analysis [44] provides common ground comparison for AutoML open-source tools, because they are often compared incorrectly. Although, comparisons are not used to find out which methodology is the absolute best, because

there is no such thing, it helps users to identify the model that will work best on their given circumstances and framework [44]. 39 different datasets were used, as well as the same performance metrics across the board: area under the receiver operator curve (AUROC) for binary classification and log loss for multi-class; resource usage and allocation were homogenized across methodologies [44].

This benchmark approach is still going and updates in real time the performance of each framework and is constantly updating datasets [69]. On one of those updates, H2O outperformed other methodologies in a variety of datasets [69]. The benefits of using H2O rather than any other platform, is its capacity of scalability, speed and parallelism, because of the multi-layer stacked ensembles. It can also use different machine learning algorithms with different parameters and ranks them in a *leaderboard*; it can be sorted out by any performance metric, training time or per-row prediction speed [69].

H20 is, then, an online open source tool available for Python, R, Scala, and Java that provides competitive results for numerous datasets using Random Search and Super Learning [69]. For this work, the Python implementation was used. Furthermore, a brief description of each Machine Learning used in the Stacked Ensemble models or Leaderboard system.

2.7.1 Support Vector Machine

Support Vector Machine, or SVM, is a widely known Machine Learning and Statistical model used mostly for classification purposes, but it is not limited to regression [27]. Statistical Learning Theory is substantially used in SVM; it means that through the algorithm finds a predictive function from the data used for modeling [115]. This is manageable by creating a feature space by maximizing the margin distance between or among classes. In other words, the creation of an optimal hyper-space [86].

The feature space can be expressed given an n-feature database and its respective labeling represented by the pair: (X_u, y_u) where $u = 1, 2, 3, ..., i$, $X_u \in R_n$, and $y_u \in \{-1, 1\}$. Each pair can be linear separated if there is a vector w and a scalar b that fulfill the inequality [86]:

$$w \cdot X_u + b \leq -1, \; when \; y_i = -1 \tag{2.12}$$

Multiplying equation 2.12 by y_u, we obtain:

$$y_i(w \cdot X_u + b) \geq 1, \; i = 1, 2, 3, ..., I \tag{2.13}$$

After this, the optimal hyperspace can be found when equation 2.13 equals to zero, for w_0 and b_0, which stand for the weight vector and the bias (scalar), respectively. Then, to maximize the space in-between boundary, the weight vector **w**'s module should be minimized [86]. So, the relationship is given by:

$$min(\frac{1}{2}\|\mathbf{w}\|^2) \tag{2.14}$$

The new condition described in equation 2.14 changes the nature of the problem into an optimization-type. One course of action is to use Lagrange multipliers $a_u \geq 0$ for each (X_u, y_u) pair [86]. The resulting equation is a Lagrange function of the form:

$$L(\mathbf{w}, b, \mathbf{a}) = \frac{1}{2}\|\mathbf{w}\|^2 - \sum_{u=1}^{i} a_u\{y_u(\mathbf{w} \cdot \mathbf{X}_u + b) - 1\} \tag{2.15}$$

The resulting equation, also known as the linear decision function [27], is then express as:

$$L(\mathbf{w}, b, \mathbf{a}) = sign(\sum_{u=1}^{i} a_u\{y_u(\mathbf{w}_0 \cdot \mathbf{X}_u + b_0)\}) \tag{2.16}$$

For the case that the data is not linearly separable, then different kernels are used to compensate for this issue. These kernels can linearly separate the data, but in higher dimension spaces [46]. Each kernel has its own definition and function, the most used are *linear*, *polynomial*, and *Radial Basis Function*. The general kernel function is defined as:

$$k(\mathbf{X}_u, \mathbf{X}'_u) = \phi(\mathbf{X}_u)^\top \phi(\mathbf{X}'_u) \tag{2.17}$$

Not only that, from equation 2.17 we can also obtain a classification function for two classes similar to equation 2.16 [86]:

$$f(\mathbf{X}) = \sum_{u=1}^{i} a_u y_u k(\mathbf{X}_u, \mathbf{X}'_u) + b \tag{2.18}$$

Support Vector Machine models can be computed with these formulations and the open source tools available online.

One thing to notice is that standardization and/or normalization can help in stabilizing the overall model when training and testing, but it does not guarantee better results in few cases [13].

The *Radial Basis Function* (RBF) kernel is widely used in Support Vector Machines for non-linear problems, because it provides better accuracy results and lower testing costs than other kernels [21]. Furthermore, this kernel works best with datasets with reduced number of instances or feature vectors; Nonetheless, different kernel approximations can help in solving this issue.

The formal definition of the RBF kernel is:

$$k(\mathbf{X}_u, \mathbf{X}'_u) = \exp(-\gamma\|\mathbf{X}_u - \mathbf{X}'_u\|^2),\ \gamma > 0 \tag{2.19}$$

The special value in equation 2.19 is γ, which is a regularization hyperparameter that represents how much influence a single training instance has to respect the rest. The higher values γ have, the more together each example have to be of each other to be considered support vectors [21]. Inversely, it could represent a circumference that selects the samples as support vectors [92].

Another regularization parameter that is used in SVM but not included in the formulations is C. It represents a penalty factor that capitalizes on the decision function margin [92]. In other words, there is an inverse relationship between said margin and correct classification of examples: the smaller the margin, the better classification a model has and C controls

this trade off; the larger C is the smaller the margin and the better the model in classifying, whereas the smaller C is, the opposite happens [92].

Keerthi and Lin [59] noticed that there is a way in which an RBF kernel could behave linearly, with the use of $gamma$ and C. Let us rewrite equation 2.19 as [21]:

$$k(\mathbf{X}_u, \mathbf{X}'_u) = \exp(-\frac{\|\mathbf{X}_u - \mathbf{X}'_u\|^2}{2\sigma^2}) \tag{2.20}$$

They explained that, as $\sigma^2 \to \infty$ and with a penalty parameter value of $\frac{C}{2\sigma^2}$, then the RBF kernel has similar testing results as a linear kernel [59].

Something similar can be done to approximate the RBF kernel into a polynomial kernel. Lippert and Rifkin [75] found out that the penalty parameter $(\frac{1}{2\sigma^2})^{-2d}C$ would make the RBF kernel to behave degree-d polynomial, when $\sigma^2 \to \infty$.

These relationships represent the connections between kernels and how they tend to behave in certain conditions.

2.7.2 Cox Proportional Hazards (CoxPH)

Cox Proportional Hazards (CoxPH) models are a type of survival models; statistical models that analyze the expected time in which a given event occurs. In particular, CoxPH tracks these events with a function called the *hazard function*, which can be defined as, in a CoxPH model:

$$h(t) = \lambda(t)e^{\mathbf{x}_i^T \beta} \tag{2.21}$$

$\lambda(t)$ and β are common to all observations in the model: the baseline hazard function and coefficient vector, respectively [5], because of this combination of functions, CoxPH is considered a semi-parametric model.

The purpose of this model came to be on the bases of describing a multivariate process as a proportionality of other co-variate processes allowing for a statistical regression analysis considering patterns and time dependent co-variates, ultimately, to measure time [5].

There are different considerations when two or more events do not occur in the same instance of time, then the partial likelihood for these observations should be calculated, which is defined as:

$$PL(\beta) = \prod_{m=1}^{M} \frac{e^{w_m \mathbf{x}_m^T \beta}}{\sum_{j \in R_m} w_j e^{\mathbf{x}_j^T \beta}} \tag{2.22}$$

M is the set of all event observations and R_m is the set of risk observations in a given period of time [107].

The same can be applied to the case where the events do occur at the same time, and they are observed:

$$PL(\beta) = \prod_{m=1}^{M} \frac{e^{\sum_{j \in D_m} w_j \mathbf{x}_j^{\beta}}}{(\sum_{R^*:\|R^*\|=d_m [\sum_{j \in R^*} w_j exp(\mathbf{x}_j^T \beta)])^{\sum_{j \in D_m} w_j}} \tag{2.23}$$

The main difference between equation 2.22 and 2.23 is the set D_m of observations. Each of which has a size of d_m for a specific event in a time t_m [107]. Nonetheless, these equations

are considerably expensive to compute. The best approach is to use an approximation. The best ones are: Efron's and Breslow's Approximations. The former resembles more to the exact calculations [49].

2.7.3 Deep Learning

Deep learning is a type of Machine Learning technique that is based upon neural networks [11]. There are different variants and architectures used in Computer Vision, Speech, Image, and Music recognition, biosecurity, bioinformatics, Autonomous transportation, Natural Language Processing, those of which the most common architectures are: Recurring Neural Networks (RNN), Convolutional Neural Networks (CNN), Feedforward Artificial Neural Networks (ANN)/Multi-Layer Perceptron (MLP) [26][54]; these last ones are the most common used on Table-data-like problems, such as the one presented in this work, and is the one natively used on H2O implementation.

The Multi-Layer Perceptron architectures could be constructed with three layers or more (i.e., input, hidden(s), and output layers) and have a linear activation function in every neuron. There are several functions but the more popular and used are [42]:

Sigmoid function:

$$f(x) = \frac{1}{1 + e^{-x}} \tag{2.24}$$

Tanh function:

$$f(x) = tanh(x) = \frac{2}{1 + e^{-2x}} - 1 \tag{2.25}$$

Rectifier linear unit (ReLU):

$$f(x) = \begin{cases} 0 & \text{for } x < 0 \\ x & \text{for } x \geq 0 \end{cases} \tag{2.26}$$

The network learns by means of *back-propagation*, which is a supervised learning technique, in which at each output of node j of each hidden layer compares the expected result with the one calculated for each training sample n. It is described mathematically as [65]:

$$e_j(n) = d_j(n) - y_j(n) \tag{2.27}$$

Where, $d_j(n)$ is the expected value, and $y_j(n)$ is the one calculated.

Furthermore, all the weights have to be updated accordingly, using *gradient descent*:

$$\Delta w_{ij}(n) = -\eta \frac{\partial \varepsilon(n)}{\partial v_j(n)} y_i(n) \tag{2.28}$$

Where, $\varepsilon(n)$ is the cost function (eq. 2.29), $y_i(n)$ is the output of the previous neuron and η is the *learning rate*, which the smaller it is, the better the network will learn (but will be computationally costlier), and v_j is the induced local field, which is the output of each neuron before applying the activation function to them [65] [68].

$$\varepsilon(n) = \frac{1}{2}\sum_{j} e_j^2(n) \tag{2.29}$$

Neural Networks in general are considerably difficult to properly tune. Hyperparameter and parameter tuning in Neural Network is one of the biggest challenges the field has [70]; many approaches have been proposed in the literature: Transfer learning, genetic algorithms, etc. It depends on the number of hidden layers and neurons in each layer that the parameters can grow considerably. The H2O deep learning models have tanh, rectifier, and max out activation functions, different layers of regularization (L1/L2), dropout, and check pointing; other advanced features are: adaptive learning rate, rate annealing, and momentum training [69].

2.7.4 Distributed Random Forest and Extremely Randomized Trees

Random Forest, in general, is a popular Machine Learning model used across many applications, because it is an easy concept to understand and explain to others not familiar with the field [17]. It is a type of ensemble model, because it uses *ensemble learning*, particularly building multiple decision trees that will output the given class predicted (for classification problems) or average the results of all end-decision trees (for regression problems) [52].

The particularity of Random Forests is that they use all the data available to construct the trees to provide a solution (respecting the training/testing split), but with *Distributed Random Forest* more subsets of data (specially attributes) are created to ensemble more trees to compensate for variation among data [69]. Instead of averaging results only for regression, it requires also to do so in classification.

Furthermore, The splitting rule in *Extremely Randomized Trees* is obtained by randomizing the thresholds of candidate features while constructing each tree level. This comes with a cost, it compensates for variance but sacrifices overall bias [69]. The *H2O* library provides a wide number of parameters to tune a Distributed model, which does it efficiently and automatically by testing with different numbers and ranks them by performance.

This split process is controlled with the parameter K, which is the number of random splits at each level of Tree-creation. It has a limited range of possible values: from 1 to N, where N is the total number of attributes of a data set [43]. When K is set to 1 then the forests are built fully randomized and independent of the target variable, regardless if it is a regression or classification problem. In other words, the smaller K is, the more randomized and independent the trees are built [43].

2.7.5 Generalized Linear Model

Linear models describe an independent variable (usually called *response* variable) in terms of the dependent variables (or *predictor* variables) linearly. They present an intrinsic error of prediction that usually is normally distributed [20]. Nonetheless, this distribution is a limiting to other, more complex, applications. For this reason, there have been other linear models developed to try to address this issue (e.g., Poisson, binomial, gamma regression and so on) [79].

This causes a lot of regression models and causes confusion on which to use when; so, *Generalized Linear Models* (GLM) unifies all of them into a single procedure and also extends their use to even be applied to classification problems; all this being is possible with the help of a *link function* and an iterating process to reweigh the system's error for maximum likelihood [84].

The link function is a relationship between the predictor and the distribution error, specifically it's mean. In other words, it maps a non-linear relationship into a linear one. This function varies depending on the data set, attributes, type of problem (i.e., regression or classification), among others [84]. There is a *canonical* link function which comes from the density function, but it could also be a particular function depending on the aforementioned reasons [79].

It can be defined as a function of μ, say $g(\mu)$, that relates the input vector X with a factor β so that $X^\top Y$ is statistical sufficient. $X\beta$ changes according to the distribution the user wants to obtain, depending on the problem they are working with [79].

The following table (2.1) summarizes how $g(\mu)$ changes depending on the distribution and purpose of the GLM.

Table 2.1: Link functions according to their respective distribution [79].

Name	**Function**	**Distribution**
Inverse	$X\beta = \mu$	Normal
Negative Inverse	$X\beta = -\mu^{-1}$	Exponential Gamma
Log	$X\beta = \ln(\mu)$	Poisson
Logit	$X\beta = \ln(\frac{\mu}{1-\mu})$	Binomial* Multinomial Categorical

*For binomial distribution, 1 should be replaced by N, the number of positive classifications on the target variable.

2.7.6 Generalized Additive Model

Generalized Additive Models (GAMs) are a particular case of GLMs that replaces the link function $X\beta$ for Xs, where s is the so-called *smooth* function, which is found through a process called *local scoring* algorithm, using *scatter plot smoother* iteratively to find s [50]. This means that the smooth function changes and adapts depending on the representation of the given problem in the local scoring algorithm.

To understand this concept, let us define a simple linear model with n observations, x_i with corresponding response variable y_i, where the latter is an observation in Y_i. Let u_i be the linear relationship between predictor and response variables, where is defined as:

$$u_i = \beta_i x_i + \beta_0 \tag{2.30}$$

Furthermore, x_i and Y_i share the following relationship:

$$Y_i = u_i + \epsilon_i \tag{2.31}$$

Where, ϵ_i are the zero mean variables and β_i and β_0 are unknown variables that can be obtained with GLM process [69].

To establish a simple GAM Model, we can proceed from equation 2.31 as follows:

$$Y_i = f(x_i) + \epsilon_i,\ where\ f(x_i) = \sum_{j=1}^{k} s_j(x_i)\beta_j + \beta_0 \tag{2.32}$$

Equation 2.32 is an extension of a regular linear model, β_j and β_0 are still unknown variables obtainable through GLM and s_j are the called *smooth function*. These functions are also to be obtained through the aforementioned *scatter plot smoother* process [50]. Both GAM and GLM are similar in the obtaining of their "link" functions, and their conditions are similar as well: find non-linear covariate effects in exponential and/or likelihood-based models [50].

2.7.7 Maximum R Square Improvements

The *Maximum* R^2 technique is focus on finding the best one-variable, two-variable, three-variable, ..., n-variable model with the highest R^2 score. This process only holds when the best one-variable R^2 has been found, then by each iteration it concatenates the second R^2 highest score to the previous model, afterwards it calculates a new score and actively changes the second variable with the rest and recalculates R^2, until the best overall score is obtained. It continues until it has found the best models for each number of variables up to the n-variable model [69].

The aforementioned process is conceptually. In actuality, the H2O library does the following [69]:

1. Generate all combinations of predictors possible.
2. Generate the training sub sets among said combinations.
3. Calculate R^2 from each model created.
4. Choose each best.
5. Store the results and the variables' names in an array.

Is a simple procedure that takes advantage of GLM (see 2.7.5) for its parameters.

2.7.8 Naïve Bayes Classifier

Naïve Bayes (NBC) is a classification technique that takes advantages of the *Bayes Theorem* by assuming independence between predictor and response variables. Furthermore, a Gaussian distribution is also expected from the predictor variables; their mean and standard deviation are obtained from the training set [94].

In this type of model, missing data is omitted completely from the probability calculation (testing phase) and also from training phase of the technique [69]. The data is not required to be sorted nor shuffled for training.

The NBC model can be explained with a binomial case. Let $\mathcal{X}$ be a subset of a discrete set of features, of the form $\{X_i, y_i\}$ for every i-th observation. So, the joint likelihood probability can be defined as [94]:

$$\mathcal{L}(\phi(y), \phi_{i|y=1}, \phi_{i|y=0}) = \prod_{i=1}^{m} p(X^{(i)}, y^{(i)}) \tag{2.33}$$

The ϕ function is the maximum likelihood of the given events to happen: $\phi(y)$ is the likelihood of the observed target variable to be predicted from $\phi_{i|y=1}$ and $\phi_{i|y=0}$ [94].

The main objective is to maximize $\phi(y)$, $\phi_{i|y=1}$, and $\phi_{i|y=0}$, which are:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m}(x_j^{(i)} = 1 \cap y^i = 1)}{\sum_{i=1}^{m}(y^{(i)} = 1)} \tag{2.34}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m}(x_j^{(i)} = 1 \cap y^i = 0)}{\sum_{i=1}^{m}(y^{(i)} = 0)} \tag{2.35}$$

$$\phi(y) = \frac{(y^i = 1)}{m} \tag{2.36}$$

The next step to create a Bayes model is to calculate probabilities from features $X_{(i)}$ with the help of the Bayes Theorem [74]:

$$p(y = 1 \mid x) = \frac{\prod p(x_i \mid y = 1)p(y = 1)}{\prod p(x_i \mid y = 1)p(y = 1) + \prod p(x_i \mid y = 0)p(y = 0)} \tag{2.37}$$

And

$$p(y = 0 \mid x) = \frac{\prod p(x_i \mid y = 0)p(y = 0)}{\prod p(x_i \mid y = 1)p(y = 1) + \prod p(x_i \mid y = 0)p(y = 0)} \tag{2.38}$$

From equations 2.34 and 2.35 we can smooth them down with Laplace and this allows the model to predict further observations from maximum likelihood estimates (although it should be used in cases where the events are scarce or rare) [69]:

$$\phi_{j|y=1} = \frac{\sum_{i=1}^{m}(x_j^{(i)} = 1 \cap y^i = 1) + 1}{\sum_{i=1}^{m}(y^{(i)} = 1) + 2} \tag{2.39}$$

$$\phi_{j|y=0} = \frac{\sum_{i=1}^{m}(x_j^{(i)} = 1 \cap y^i = 0) + 1}{\sum_{i=1}^{m}(y^{(i)} = 0) + 2} \tag{2.40}$$

Note that, this example was carried out with a binary classification problem, but the process would be exactly the same to any other case or problem with which one wants to apply the *NBC* model to.

2.7.9 RuleFit

A *RuleFit* model takes advantage of trees and linear models, the former excels on accuracy and the former on interpretability [41]. This technique focuses on building trees by fitting them into the data, the rules are generated by traversing each tree and evaluates them on the data itself, then a linear model (LASSO is a popular option) is fitted into the previous model to map the new rule feature set with the original set [41].

It should be clear that this methodology is for linear models fitting into a tree generative process, so non-linear distribution and, for that matter, problems with similar issues, should be assessed separately, and if dimmed appropriately to change the overall methodology to another one, like *GLM* (section 2.7.5) or *GAM* (section 2.7.6).

2.7.10 Stacked Ensemble Models

Stacked Ensemble models are a set of different Machine Learning methods that work together to obtain better performance and results as a group rather than individually, and this process generally does better than building their respective models separately [119]. The classical example of these models is Random Forests, which takes individual decision trees and their results into a much powerful model overall.

The principal element of the construction of this model is the *stacking* process, also called *Super Learning*, where it builds diversified learners that compensate to other models and together perform better than another ensemble methodology like bagging. This is possible by training a so-called *metalearner* which main function is to find the best combination of parameters and models through cross-validation [119].

The full process is [113]:

1. **Put together the ensemble model.**
   (a) Set up all the models to stack upon themselves.
   (b) Specify the metalearner algorithm.
2. **Train the model.**
   (a) Train all individual models L.
   (b) Perform k-fold cross-validation on each model, creating the N-cross-validated models.
   (c) Now, from L and N we can obtain a new matrix of data, called *level-one* data.
   (d) Train the metalearner with the level-one data and stack it with all the other models to predict new values or for testing
3. **predict.**
   (a) First, leave the metalearner out of the test step, and use the validation set on the other models.
   (b) The predicted values from the step above are used on the metalearner for new predictions.

Overall, this type of methodology works by having the individual models work together and solve problems on their own combined with a meta algorithm to create a more robust model (e.g., boosting, bagging, voting, etc.), rather than only leaving them out of the end result or the combination of them, like boosting or bagging, respectively. But this might mean that computational time could be compromised, but that should be weighed after preliminary results are made from the models. This is the classical trade-off between computation time and performance.

2.7.11 XGBoost

One of the most popular methods in recent Machine Learning applications and research project is *XGBoost*. Its increase in popularity is due to the overall performance of the model, although it still presents drawbacks. It is a type of boosting algorithm, meaning it builds sequential models and further compares the result so that the new model trained can fix the deficiencies the previous model might present [23]. But the clear difference is how efficient this process is, because instead of only building one Decision Tree, it builds many in parallel to solve many problems at once.

XGBoost stands for Extreme Gradient Boosting, and as such it takes advantages of GPU usage for better efficient overall runtimes [69], meaning CUDA usage is needed and parallel computing (OpenMP) for better management of multithreads and cores.

Some clear limitations when trying to be implemented in a usable Framework, like H2O is the fact that natively it does not support Windows as an environment, it only supports Linux and OS X. Nonetheless, tools like Google Colab are a fair structure to use Auto Machine Learning libraries like H2O [69].

One thing to remember is that AutoML tools will test with different models at once, rank them and build a new and refined model that will satisfy the needs of the user by doing extensive testing among the models and rank them by their better performance or do a stacking model and work from there. Also, key to everything, is that the intention of H2O-like libraries (Or in general of AutoML, See section 2.7) is the welcome of new users into the Machine Learning and Artificial Intelligence domain by providing easy-to-use tools for them. And for experience professionals and researchers, it provides them a tool to efficiently train a large amount of models at once with an easier notation and without worrying too much of parameter tuning.

Chapter 3

Front-End Design

In section 4.1, chapter 4 there is a discussion about the scarcity of databases of supervised handwriting and drawing-based data sets, specially on emotion recognition. EMOTHAW is a data set that provides a framework to test ideas (i.e., models, feature extraction, selection, etc.) for everyone interested in problems like recognizing emotions through data. In this work, there were important ideas developed in order to create a model that could yield considerable improvements compared to the baseline of this project: *EMOTHAW: A Novel Database for Emotional State Recognition from Handwriting and Drawing* by Likforman-Sulem, *et al.* [73]. Although, for in-depth result comparison, see Chapter 4.

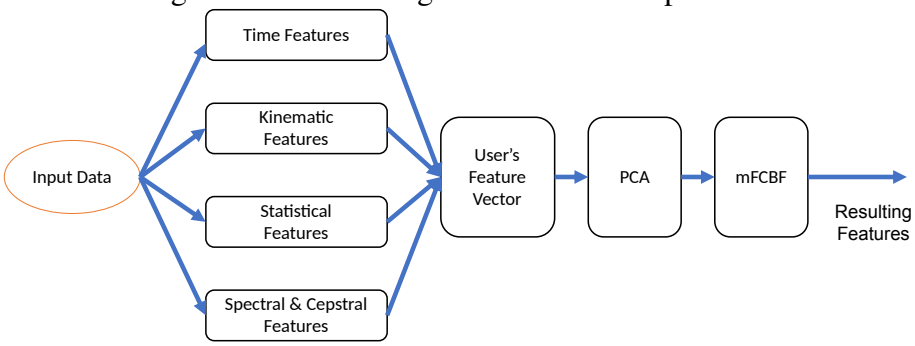
Given the relative uniqueness of this problem, there were several lines of work that one could pursue or develop; Even further, the extension of possible testings and experimentation are of large number, and due to the time frame given by the Master program, most of them had to be left for future work (More in section 5.2, chapter 5). This said, the focus of this methodology is in front-end (i.e., in feature extraction and dimensionality reduction rather than in robust Machine Learning models), creation of attributes, signal processing, parameter tuning and efficient Machine Learning training/validating/testing tools such as H2O.

The main feature extraction methodology is based on the following block diagram (figure 3.6), which primary purpose is to establish a working path towards improvements, mostly by using signal processing techniques to draw features for a problem that resembles that of a traditional time series, focusing on spectral and cepstral realms (See section 2.2.2 and 2.2.3, respectively in chapter 2). The addition of other types of features (i.e., statistical, kinematic, time) helped to create a unique model that had enough information to work with different dimensionality reduction techniques, some of which are novel as of this work.

Dimensionality reduction and feature selection played an important rule in this work, because they provided another layer of abstraction and information for the sub-data sets created thanks to feature extraction. The main feature selection algorithm used was *Fast Correlation-Based Filtering* (FCBF) [122] and its modified version [86] (see sections 2.5.3 and 3.3.2, respectively in chapter 2 and chapter 3) and the go-to dimensionality reduction algorithm was Principal Component Analysis; But, the important step here is, the combination of both steps to further filter unwanted information/attributes on the data sets (see figure 3.1 for a graphical representation).

As mentioned in section 2.2 in chapter 2, signal processing is a special type of feature extraction that requires specific parameters to work as consistently as possible (mitigating

Figure 3.1: Block diagram of Front-End process.



uncontrolled variables' noise). It constitutes an important part of this process and further will be talked about in section 3.2.4.

The H2O framework provides an important work space in which different Machine Learning models can be tested out and compared with each other without sacrificing time and resources, which is particularly convenient for the later stages of this work. In the early stages the first approach was to test a different traditional model with the original time, spectral, and cepstral features; Support Vector Machine proved to be the better performing algorithm (more in section 3.7).

First things first, let's review the resulting feature vector and where it comes from:

3.1 The EMOTHAW Raw Features

In section 4.1 (EMOTHAW), the database contains seven initial attributes to work with, all of which were used in the current work, and they could be also used in future work, but that is left to consideration. The original attributes are : $x(n)$ X position, $y(n)$ Y position, Ts Timestamp, $sq(n)$ On/Off-pen status, $az(n)$ Azimuth, $al(n)$ Altitude, $p(n)$ pressure were used to generate Time, Kinematic, Statistical, Spectral and Cepstral features.

All of them share the same tasks and raw attributes, the main difference is the score measured by the DASS scale. because of this, let's define the task set C such that:

$$C = \{pentagons,\ house,\ words,\ LH-loops,\ RH-loops,\ clock,\ sentence\} \quad (3.1)$$

In the EMOTHAW database, there are also seven different tasks that are considered the raw attributes that can be defined as the set of raw attributes T for each task c in C, and each user u so that (detail information of each task in section 4.1.2 in chapter 4):

$$T^{C,u} = \{x^{C,u}(n),\ y^{C,u}(n),\ Ts^{C,u},\ sq^{C,u}(n),\ az^{C,u}(n),\ al^{C,u}(n),\ p^{C,u}(n)\} \quad (3.2)$$

As we can see from sets 3.1 and 3.2 there would be a considerable amount of new features generated and taken advantaged off different feature extraction techniques. In section 2.3, chapter 2 there is an exposure of several algorithms that are widely used in the literature in different applications, most of which relate in a way to this research work, and they are: *Time*, *Kinematic*, *Statistical*, *Spectral*, and *Cepstral*.

3.2 Feature Extraction

3.2.1 Time Features

As explained in Section 2.3.1 in Chapter 2, the Time Feature vector was proposed by Likforman, *et al.* [73], but it has relative limited information drawn from the original data set. Nonetheless, crucial information can be derived from them when combined in a valid arithmetic way [86]; setting the new TF vector for user u and every task c in C as:

$$TF^{C,u} = \left[(F_1)^{C,u}, (F_2)^{C,u}, (\dot{F}_1)^{C,u}, (\dot{F}_2)^{C,u}, \mathfrak{t}^{C,u}\right] \tag{3.3}$$

The first two features were used as in the original paper of the EMOTHAW database [73] and a complete description of these features is in section 2.3.1, chapter 2. The third feature $(\dot{F}_1)^{C,u}$ represents the normalization of $(F_1)^{C,u}$ to task in-paper duration ($(\dot{F}_1)^{C,u} = \frac{(F_1)^{C,u}}{\tau}$). The same goes for $(\dot{F}_2)^{C,u}$ and $(F_2)^{C,u}$. The last feature, $\mathfrak{t}^{C,u}$ is the ratio between the duration of pen on-air and the duration in-paper over the tablet.

Thus, the complete set of Time Feature vector would be then defined as (for all tasks and users):

$$TFv^U = \left[\bigcup_{c \in C} [TF^{c,u}]^\top\right]^\top, \text{ for every } u \in U \tag{3.4}$$

The TF vector is defined as a row column, so to concatenate all of these features for all users, it has to be transposed beforehand. But not only that, once each row vector is transpose and concatenated accordingly, it has to be further transpose again to keep the original structure intact.

Figure 3.2: Graphical representation of the Time feature generation process

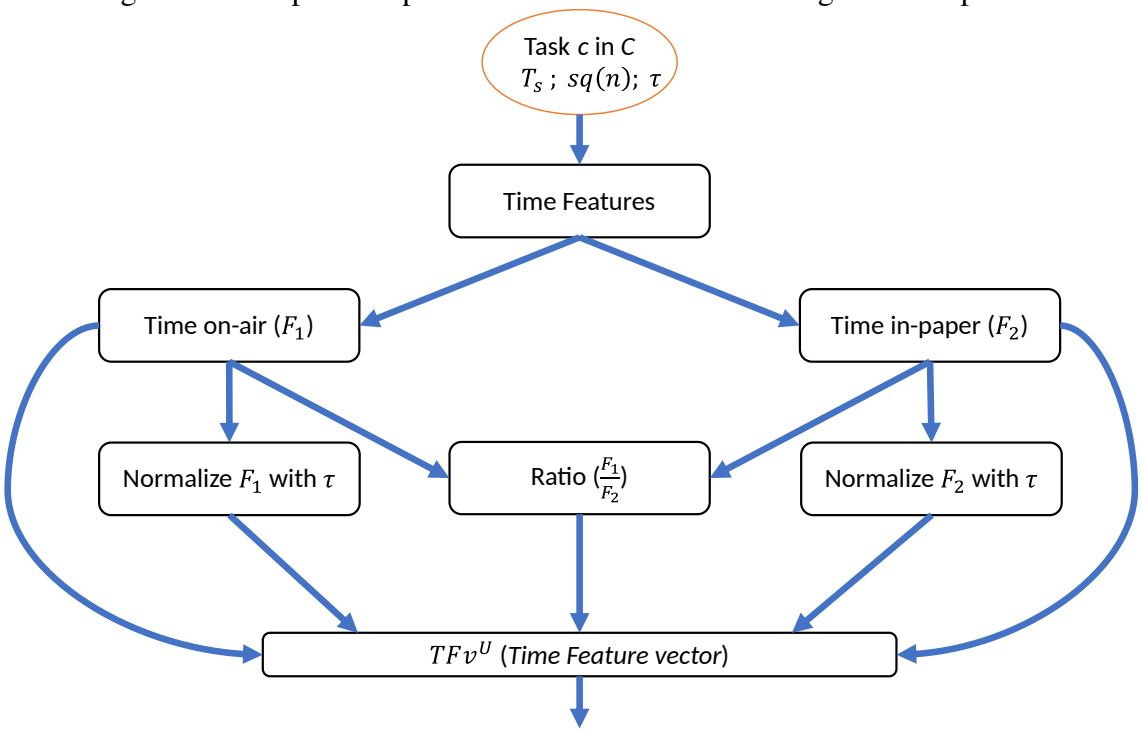


Figure 3.2 shows a brief visualization of how each feature step is interconnected with each other and how they all form the final *Time Feature Vector*, Although it is not a detailed view of how each feature came to be, it provides a better understanding of the vector as a whole.

3.2.2 Kinematic Features

Kinematic features are proved to be of great use in Machine Learning applications when the given conditions are met [56] (see Section 2.3.2 for further information). As previously mentioned, the *Kinematic Feature* vector of user u and task τ (in this work's case task c) used in Parkinson's disease can be also applicable to the EMOTHAW database, because of similarities between data sets [85]; meaning the features explained in Section 2.3.2 are also used for this work.

The main raw attributes used for this analysis are: horizontal and vertical position of pen ($x(n)$ and $y(n)$) of each task. With them, kinematic features can be drawn, and they help in creating a new feature vector as well (full definition of these features can be seen in section 2.3.2, chapter 2).

So, the Kinematic Feature vector for a user u can be written as:

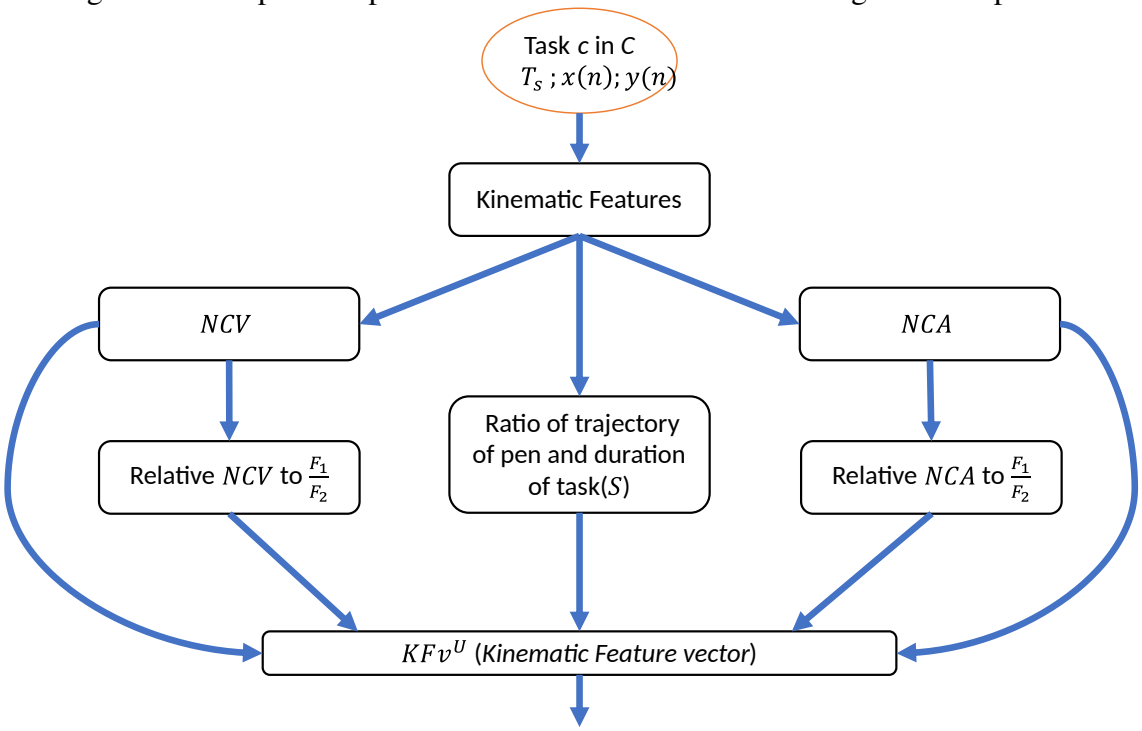
$$KF^{C,u}_{d^{C,u}(n)} = \left[S^{C,u}, NCV^{C,u}, NCA^{C,u}, NCV^{C,u}_{\mathfrak{t}}, NCA^{C,u}_{\mathfrak{t}}\right] \tag{3.5}$$

All of which are fully described in section 2.3.2, chapter 2 and $d^{C,u}(n)$ is the displacement of the pen in the tablet.

Now, the complete set of Kinematic Feature vector for every user u in U and each task c would be defined as:

$$KFv^U = \left[\bigcup_{c \in C} \left[KF^{c,u}_{d^{C,u}(n)} \right]^\top \right]^\top \text{, for every } u \in U \tag{3.6}$$

Figure 3.3: Graphical representation of the Kinematic feature generation process



In figure 3.3 the same approach from figure 3.2 in the *Time Feature Vector* Section (3.2.1) follows where, a brief visualization of the overall process is presented within the diagram. Each step has been explained in detail in this section and should only provide the reader an understanding and a small review of this process.

For now, there are two new sets of features that complement each other and take full advantage of different attributes on the original database. Nonetheless, those abstractions are not enough to create a new characteristic vector to feed a model to outperform considerably the baseline of this work. So three different feature types are going to be generated from the original attributes and the Kinematic and Time features alike.

3.2.3 Statistical Features

In particular, the *Statistical* features are generated from *kinematic* and the so-called stroke signals (also known as Number of strokes, see section 2.3.2 and 2.3.1, respectively in chapter 2). It is the bigger feature vector in this work in terms of number of different techniques to create them. This reason is enough to define individual statistical feature vectors according

to the method used to draw them. In this case, following the division made in section 2.3.3, chapter 2.

The first step is to define the set in which the different statistical methods are going to be applied to, in this case for every user u and task c:

$$\mathfrak{g}^{C,u}(n) = \{k_w^{C,u}(n), \mathbb{S}^{C,u}(n)\} \tag{3.7}$$

Where n is the length of the signal, the first term, $k_w^{C,u}(n)$ is a given set that contains the three main kinematic signals $v_w^{C,u}(n)$ velocity, $a_w^{C,u}(n)$ acceleration, and $j_w^{C,u}(n)$ jerk, and as mentioned in section 2.3.2 $w^{C,u}$ is the set of displacements where distance, horizontal, and vertical positions are. Lastly, $\mathbb{S}^{C,u}(n)$ is the stroke signal. Each one of the signals here used are fully defined in section 2.3.2.

As mentioned before, the set 3.7 is where all the statistical methods are going to be applied. There are three main groups of methods previously defined in section 2.3.3: *Descriptive* ($\mathcal{D}$), *Momentum* ($\mathcal{M}$), and *Mean*($\mathfrak{M}$) statistic groups. Each with their own methods. Said this, let's define the Statistical Feature vector for user u and task C:

$$SF_{\mathfrak{g}}^{C,u} = \left[\mathcal{D}_{\mathfrak{g}}^{C,u}, \mathfrak{M}_{\mathfrak{g}}^{C,u}, \mathcal{M}_{\mathfrak{g}}^{C,u}\right] \tag{3.8}$$

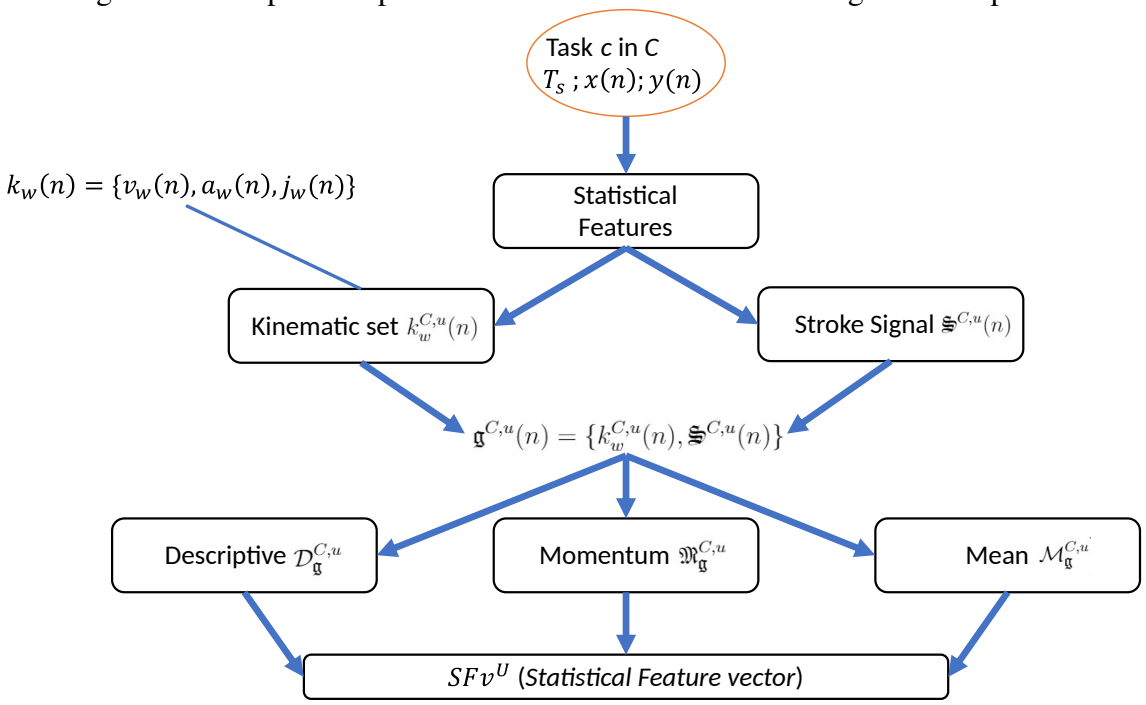
And, subsequentially, the complete Statistical Feature vector for all users is defined as:

$$SFv^U = \left[\bigcup_{c\in C}\bigcup_{\mathbf{g}\in\mathfrak{g}^{C,u}(n)} \left[SF_{\mathbf{g}}^{c,u}\right]^\top\right]^\top, \text{ for every } u \in U \tag{3.9}$$

The jointed set union symbols are part of relational algebra. It helps keep track of the overall process linked to all tasks, all users, and all statistical signals involved in SFv^U.

Figure 3.4 has a graphic representation of the overall statistical process and how each input is transforming to the end-result of the statistical feature vector.

Figure 3.4: Graphical representation of the statistical feature generation process



3.2.4 Signal Processing

Spectral-domain Features

The idea behind the relational algebra that is applied to the different types of feature vectors and sets have been proved self-explanatory, and for Spectral and Cepstral feature vectors alike, its applicability is likewise.

This particular definition of feature vector takes advantage of the usage of signal processing as a feature extraction technique instead of its traditional use. Meaning that, the generation of filters help out in creating a wide range of new numerical features that intrinsically have crucial information about a time series signal. So, the definition of the spectral-domain feature row vector is:

$$SDF_s^{C,u} = \left[LEFB_s^{C,u}(1), ..., LEFB_s^{C,u}(M)\right], \text{For } m = 1, 2, ..., M \tag{3.10}$$

Where, $s^{C,u}$ is a subset of $T^{C,u}$ that only contains $\{x^{C,u}(n), y^{C,u}(n), p^{C,u}(n)\}$ (horizontal, vertical positions, and pressure). $LEFB$ stands for *Logarithmic Energy FilterBank* and is defined as:

$$LEFB_s^{C,u}(m) = \underset{\theta=1,2,...,M}{\text{filterbank}}(E_s^{C,u}(k)), \text{For } m = 1, 2, ..., M \tag{3.11}$$

Where, $E_s^{C,u}(k)$ is the logarithmic energy spectrum obtained from the Discrete Fourier Transform of the signal $s^{C,u}$ (detail explanation in section 2.2.2).

Furthermore, the complete feature set of the spectral-domain attributes on every user u, task C, and signal $s^{C,u}$ is:

$$SDFv^U = \left[\bigcup_{c\in C}\bigcup_{\mathbf{s}\in s}[SDF_{\mathbf{s}}^{c,u}]^\top\right]^\top, \text{ for every } u \in U \tag{3.12}$$

Cepstral-domain Features

Similarly, the same case applies to the Cepstral feature vector and complete set of attributes, so the definition of the former (CDF) is:

$$CDF_s^{C,u} = \left[FBCC_s^{C,u}(1), ..., FBCC_s^{C,u}(Q)\right], \text{For } q = 1, 2, ..., Q \tag{3.13}$$

Where, $Q = \frac{M}{2}$ and $FBCC$ stands for ***FilterBank Cepstral Coefficients*** and it's defined as:

$$FBCC_s^{C,u}(q) = \sum_{m=0}^{M-1} LEFB_s^{C,u}(m)e^{-\frac{2\pi i}{N}qm}, \text{For } q = 1, 2, ..., Q \tag{3.14}$$

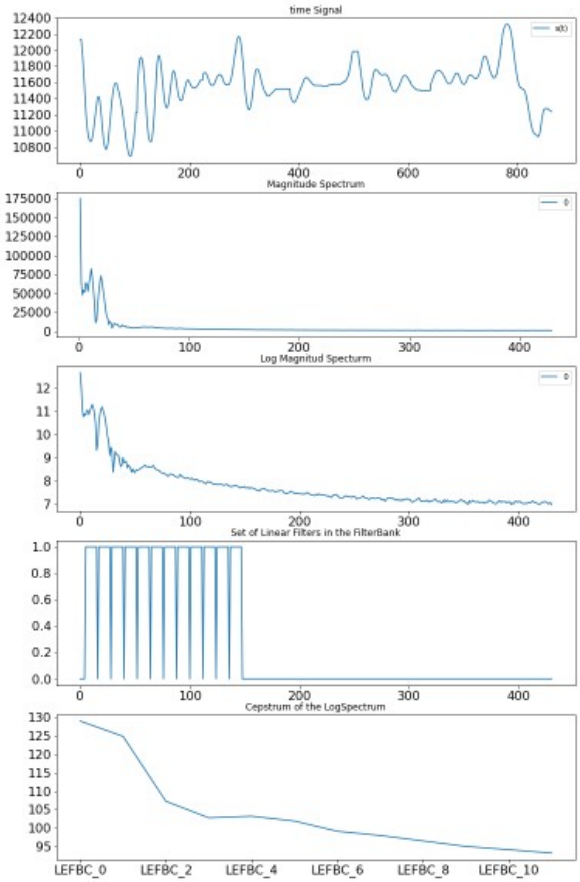
As the definition states (see section 2.2.3), *Cepstral Coefficients* are obtained by applying the Discrete Fourier Transform to the log energy, in this case $LEFB$. Q represents only the half of the spectrum, because when looking at its complete form, the information in it is mirrored.

Lastly, the complete cepstral-domain feature set for all users u is defined in relational algebra as:

$$CDFv^U = \left[\bigcup_{c\in C}\bigcup_{\mathbf{s}\in s}[CDF_{\mathbf{s}}^{c,u}]^\top\right]^\top, \text{ for every } u \in U \tag{3.15}$$

Spectral and Cepstral features can be analyzed separately. In figure 3.5, it can be observed how a time series from the raw data is transformed as each stage of signal processing is applied. At the end of them, the numerical information is transformed into a row vector, where it represents the new-obtained information for each user's task.

Figure 3.5: Transformation from input signal to Filterbank vector row (spectral and cepstral features).

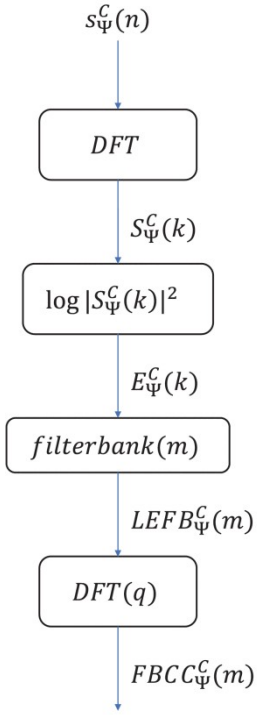


Description: The signal processing steps in a time series from EMOTHAW raw attributes example: a) is the time signal itself. b) is the Magnitude Spectrum from a. c) is the Log Magnitude Spectrum obtained from b. d) Is the set of linear filters that will be applied to c. And lastly, e) is the Cepstrum of the LogSpectrum [86].

The original database contains information for three different emotional states: *Depression*, *Anxiety*, and *Stress*, so let's represent them in set notation: let E be the set for all three emotional states, so that:

$$E = \{Depression,\ Anxiety,\ Stress\} \tag{3.16}$$

Figure 3.6: Block diagram of the Spectral and Cepstral Features.



Description: The block diagram represents the overall steps of the feature extraction methodology followed in this work. The signal $s_{\psi}^{C}(n)$ is the input time series of the EMOTHAW data set, and the blocks represent the technique used for its conditioning to obtain in the end the Filterbank's information [86].

Figure 3.6 summarizes the Spectral- and Cepstral-domain features as a whole. In which Tracks down how signals of each task in the original data set behave and how they are transformed into the final numerical data used and considered as the resulting feature vector.

3.2.5 The complete Users' Feature Vector

Every feature vector has been defined individually based on the type of techniques used to generate them. There is enough variety to consider it a diversified feature vector for the problem at hand. The list of the types of features are:

1. Time Features,
2. Kinematic Features,
3. Statistical Features,
4. Spectral-domain Features,
5. Cepstral-domain Features.

All of them represent the complete output features of this process. The Users' Feature Vector is the concatenation of all row vectors generated, meaning feature rows 3.4, 3.6, 3.9, 3.12, 3.15 will then be set up like the following:

$$CFV^U = \Big[\big[TFv^U\big]^\top \cup \big[KFv^U\big]^\top \cup \big[SFv^U\big]^\top \cup \big[SDFv^U\big]^\top \cup \big[CDFv^U\big]^\top\Big]^\top \tag{3.17}$$

For the complete resulting database, The classification output has to be considered. In this case, there are a binary and a ternary case. For which they can be represented as:

$$\mathfrak{B}^U = \begin{cases} 0, & \text{Normal} \\ 1, & \text{With Disease} \end{cases} \qquad \text{For all } u \in U \tag{3.18}$$

$$\mathfrak{T}^U = \begin{cases} 0, & \text{Normal} \\ 1, & \text{Moderate Disease} \\ 2, & \text{Severe Disease} \end{cases} \qquad \text{For all } u \in U \tag{3.19}$$

Both $\mathfrak{B}$ and $\mathfrak{T}$ are the target column vector for the binary and ternary problems presented in this work, respectively. More information about their split can be seen in section 4.2 and 4.3. With them, we can then form the complete DataFrame for each separate problem:

$$FVB^U = \Big[\big[CFV^U\big]^\top \cup \big[\mathfrak{B}^U\big]^\top\Big]^\top \tag{3.20}$$

Array 3.20 represents the DataFrame for a Single user in the Binary classification problem. One thing to notice before defining the resulting DataFrame is that, only the output or target variable changes in both cases.

This said, we can proceed to define the ternary DataFrame:

$$FVT^U = \Big[\big[CFV^U\big]^\top \cup \big[\mathfrak{T}^U\big]^\top\Big]^\top \tag{3.21}$$

Afterwards, there are two sets to unite in unison with relational algebra the set of emotions E (set 3.16) and consider all users in U. So, the newly generated row feature vector for each user u also is applied to each emotion e in E. This can be accomplished with relational algebra once again:

$$FD^E = \bigcup_{u=1}^{U} FD^{E,u} \quad U = 1, 2, 3, ..., 129 \tag{3.22}$$

With the definition of the DataFrame 3.22 (it can be done accordingly to the ternary case) there is full consideration for all the emotional states and conditions for each user in all the original database, as well as the newly generated feature vectors. This information is crucial in the next step of the process' pipeline: *Feature Selection*.

One thing to point out is that this complete feature vector is generated from a sensor-type of data. This means that the data was gathered in one way or another measuring a human activity, most likely related to their senses [85]

3.3 Feature Selection

In chapter 2, section 2.4, there were several techniques described, including their advantages and their capabilities. Nonetheless, among the methods explained, the $PrincipalComponentAnalysis$ is a reliable tool to work with within the context of numerical attributes as a pre-processing step in the Feature Selection data pipeline proposed in this work. In other words, After generating the FVD^E feature vector, the next step is to map it into a dimensionality reduction technique before selecting the relevant features (or in this case, components). This methodology proved to be a robust Dimensionality Reduction-Feature Selection procedure that enhanced the performance of the output model (more in chapter 4).

3.3.1 Principal Component Analysis (PCA)

PCA is a dimensionality reduction technique that renders a new feature vector space where each component is form from the original database, preserving covariance from it. Depending on the PCA kernel or methodology, the output shape of the resulting vector is different. In this case, it is bounded to $min(len(observations), len(attributes))$ from the input vector in the PCA method. In other words, is bounded to the number of users in this particular problem.

We can define the PCA method in this problem as:

$$O^E = \mathfrak{P}_{\%}(PCA(FVD^E)) \tag{3.23}$$

Where O^E is the most representative components vector after applying PCA and $\mathfrak{P}_{\%}$ is the representative percentage of 99% of information in the orthogonal PC dimensions. The percentage route guarantees that the vector space was maximized information without sacrificing the size of the overall new dimensions. This is important for the next step to be successful and accountable.

3.3.2 modified Fast Correlation-Based Filtering

The methodology presented in section 2.5.3, chapter 2 is similar to the one presented in this one; but, there is an important difference to take in consideration: the way redundant features are removed. FCBF methodology removes them by applying three different Heuristics, whereas the modified Fast Correlation-Based Filtering (mFCBF) works only with the correlation that each relevant feature has with the output [86].

In algorithm 1, it can be seen the overall procedure of the process. The pseudocode provides the general idea of what is different in comparison to the algorithm in section 2.5.3. The main difference is that instead of having three different heuristics, this one has an extra correlation threshold that filters redundancy and only keeps the attributes that have the highest correlation to the output and the lowest to other attributes [86].

Algorithm 1 The *mFCBF* algorithm receives the User's Feature Matrix (O^E), minimum correlation threshold (*oTh*) and the maximum correlation threshold (*iTh*) and returns the selected set of features. [86]

function *mFCBF*(O^E, oTh, iTh)
    Calculate corr(O^E)
    $O_{tmp} \leftarrow$ Select columns whose correlation with the output is $> oTh$
    Calculate corr(O_{tmp})
    $\hat{O}^E_{oTh,iTh} \leftarrow$ Select columns whose correlation with the input is $< iTh$ and with the highest correlation with the output.
    return ($\hat{O}^E_{oTh,iTh}$)
end function

3.4 Balance method - Gaussian White Noise Data Augmentation on Training Data

The data augmentation techniques discussed in 2.1 have advantages and disadvantages (Imaging data augmentation is not applicable in itself but the idea behind it is), and the one that fits better the problem at hand is to generate synthetic data with White Noise.

Time series processes specialized in forecasting have similarities and differences, but one clear characteristic they all share is the building block in which they are constructed [32].

Let y be an observed series of interest, so that:

$$y_i = \varepsilon_i \tag{3.24}$$

where, $\varepsilon_i \sim (0, \sigma^2)$ and ε_i is uncorrelated over time or *serially uncorrelated* (therefore, y_i as well). If this behavior also presents a zero mean, constant variance, no serial correlation and $\sigma^2 < \infty$, then the process can be called *zero-mean white noise* or *white noise* only [32]. The *white* term comes from the idea of light, where it is composed of all other colors of the spectrum. Mathematically, it can be expressed as:

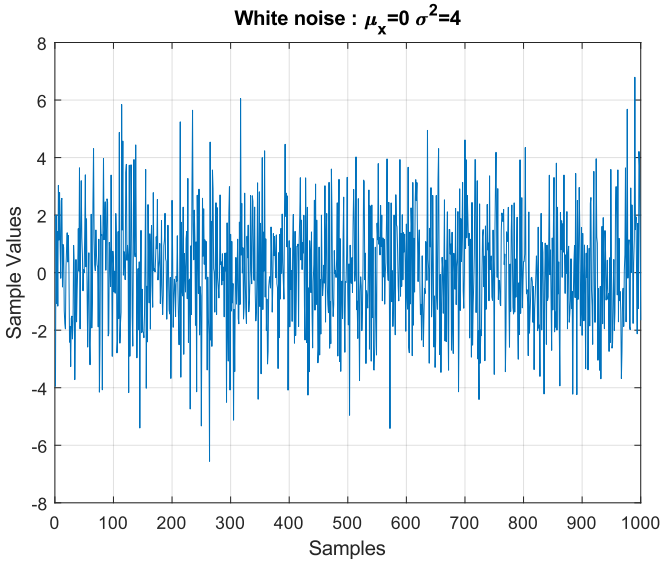
$$y_i = \varepsilon_i \sim WN(0, \sigma^2) \tag{3.25}$$

Independence between y_i and ε_i is not necessarily true, unless they were normally distributed. Given the case, then it would be called *independent white noise*; if it follows that it is also normally distributed, then it's called *Gaussian White Noise* (see equation 3.26) [32].

$$y_i \overset{\text{ind}}{\sim} N(0, \sigma^2) \tag{3.26}$$

What can be seen in figure 3.7 is the representation of what could be added to the instances of our original database to over sample the classes with less representation in the binary and specially in the ternary problem.

Figure 3.7: Gaussian White Noise synthetic example



In particular, the *White Noise* algorithm followed was:

1. Identify the imbalance class of the data, called C_m.

2. Calculate the ideal synthetic observations to create to compensate, called N_S

3. Randomly select number of samples N_S for C_m, and

4. Add the newly generated observations with a Gaussian random noise, such that:

$$FD^E = FD^E + \alpha * GV \tag{3.27}$$

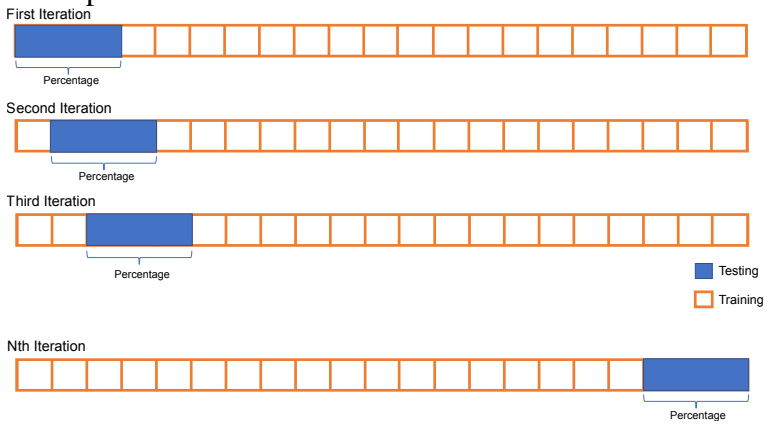
Where FD^E is the complete feature DataFrame for every emotional state E, α is the Gaussian Noise, ranging from 0 to 0.2, and GV is the *Gaussian Vector*.

3.5 Validation: Leave-Percentage-Out

As the name suggests, *Leave-Percentage-Out* (LPO) is similar to LOO. In fact, LPO is the generalization of LOO to a given percentage (i.e. a parameter to tune). In figures 2.4 and 3.8, the similarities are clearer: the only difference is the size of the testing set; instead of having only one observation for it, it is a higher observation set, depending on the percentage established [82].

The percentage parameter in LPO could be set so that LPO could work as either LOO or K-fold cross-validation.

Figure 3.8: A brief representation of how LPO works within the observations of a data set,



3.6 Hyper-parameter Tuning

In the Feature Extraction section (3.2), and Feature Selection (3.3), there are several parameters to take into consideration prior to building the Machine Learning models. There is a direct correlation on tuning hyperparameters and the accuracy performance of a model in most techniques today. In this work, there are several important parameters to consider. Spectral-domain feature vector depends on four different terms that affect also cepstral-domain features, and they are: *Filterbank Bandwidth* ($fbbw$), *Filter bandwidth in the Filterbank* (fbw), *Filterbank initial Frequency* (i_f), and *Overlap filters in Filterbank* (ov). Their range of values is as follows:

1. $fbbw = [0 - 75]Hz$
2. $fbw = [0.5 - 3]Hz$
3. $i_f = 0.5Hz$
4. $ov = 0\%$

There are other parameters that could be taken into account, but the fact is that even by only considering these four parameters, time is not enough to properly test all possible combinations. It is not feasible to even plan for the complete testing, say similar to a GridSearch of sort.

In the *modified Fast Correlation-Based Filtering* algorithm, there are two extra parameters to tune, which are oTh and iTh, which are the output and input threshold, respectively. Their range is as follows:

1. $oTh = [0, 0.2, 0.4, ..., 0.18, 0.2]$
2. $iTh = [0, 0.1, 0.2, ..., 0.8, 0.9, 1]$

In total, six different parameters are taken into account for the tuning of the modeling.

3.7 Machine Learning modeling

In traditional Machine Learning Techniques, Support Vector Machine, Random Forest, XGBoost, and others are good enough models for certain applications already, but they need time for testing and tuning, which is the resource that is of less availability. Nonetheless, a tool like H2O, an autoMachine Learning alternative, helps to tune and test on a diversified list of models that help each other and compete at the same time for better performance. There is a detail explanation of how it works and the types of models it has available in section 2.7, chapter 2.

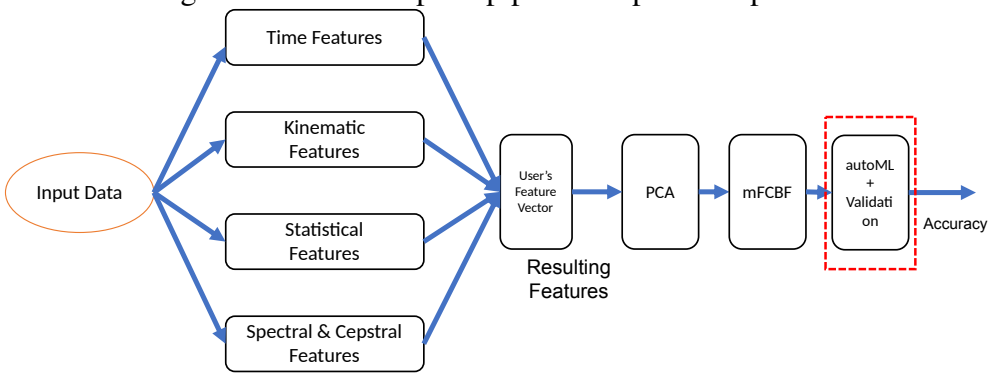
The following list describes the different models used for this work, and the reader can notice how different they are in their own complexity, simplicity, parameter tuning, and other specs that make these models unique or share aspects with one another.

- Deep Learning
- Distributed Random Forests
- Extremely Randomized Trees
- Generalized Linear Model
- Generalized Additive Model
- Gradient Boosting Machine
- Naïve Bayes Classifier
- RuleFit
- Stacked Ensembles
- Support Vector Machine
- XGBoost

The advantage of using autoML is that, all the parameter tuning the models require is done automatically and efficiently due to the Bayesian optimization done by H2O. There are several modalities for the tool, buy the one used for this work is the leaderboard approach, which will rank the performance (Area Under the ROC Curve) of all models and the one on top is the resulting metric for the modeling. It also has the possibility to control cross-validation by the means of a parameter in the H2O functions, which will be set to 2-fold cross-validation.

In figure 3.9 we can observe a general view of how the process works out and how it is chronologically carried out. First, each different type of feature is generated and then concatenated into the Complete Users' Feature vector. Afterwards, the *PCA* and *mFCBF* steps follow up to, at the end, finish with the *Modeling* and *Validation* process. One thing to notice is that in the last step, each different model works to find the best accuracy result out of all the models, just as described in section 2.7 and 3.7.

Figure 3.9: The complete pipeline steps of the process



Chapter 4

Results

The methodology proposed in chapter 3 enable for noticeable improvements compared to baseline *EMOTHAW* paper by Likforman-Sulem, *et al.*, but lets first introduce formally, the EMOTHAW database and how was it build and what describe the different parts it contains.

4.1 EMOTHAW Database

EMOTHAW stands for EMOTion recognition from HAndWriting and draWing, and is a database made by Likforman-Sulem, *et al.* [73], consisting of two-multitask components: The Depression-Anxiety-Stress Scale (DASS) and a seven-task battery of tests, both written and drawn. The former consists of a 42-item questionnaire that assesses three emotional states: Stress, Anxiety, and Depression (14-item per each) [76], which maximizes discrimination among all three states. Each of them correspond to the gathering and implementation of different psychological tests. Psychological tests that measure these types of conditions have been created and tested for over 60 years. They provide viable and reliable inside information of a given patient [73]. The latter consists of five drawn and two written tasks that serve a specific purpose to identify each emotion state. They figured that the emotional states could be identified better with particular tasks of the tests, using Random Forest for verification, they found: Drawing-like features work better on Depression and Anxiety, whereas Stress had better recognition with Drawing and Writing features.

This database is classified as a biometric type. This means that it contains information related to human characteristics, usually to: face recognition, iris detection, fingerprints, etc. (i.e., *Morphological biometrics*) [72]. The classification for handwriting and drawing is called *Behavioral biometrics*, and this field is relatively in development in that there are not many open-source databases to work with [31] and one of the main reasons for that is the fact that it is not easy to construct a supervised database in the first place, like EMOTHAW. Its applications are in e-security and e-health, for the most part [37], specially in detection of diseases and mental disorders [86].

The authors built this database to provide a reliable open-access alternative for testing in the Sensor Data community. Emotion detection through handwritten activity is in early stages. There are not a lot of public databases or resources in which to work on for the time being, meaning EMOTHAW is one of the few available online [86].

A further explanation can be found in subsection 4.1.1 and 4.1.2 for emotion states and tasks, respectively.

4.1.1 Emotions

Depression

Depression or major depressive disorder is a common mental disorder that generates a lack of interest and profound sadness, according to the World Health Organization (WHO). It is a complicated state due to the frequently missed diagnosis and its confusion with other illnesses [8]. From Beck and Beamesderfer [8] there is an estimation conducted by Lehmann (1971) that 3-4% of the depressed American population only 1 person of 5 is treated, 1 in 50 hospitalized, and 1 in 200 commits suicide. This means that for the American population in the '70s (210 million), 8,4 million were depressed, and 42,000 committed suicide. The importance of detecting depression on time is important to reduce these numbers.

Depression causes loss of interest in activities that were enjoyed by the individual; energy, willingness and sadness take over the brain, causing the body to feel tired. Furthermore, severe cases end in suicide [8].

Beck[8] recognized the difficulties of diagnosing depression. It led to the creation of the Beck Depression Inventory (BDI), a 21 question self-managed questionnaire that assesses the severity of depression.

Although, theoretically speaking, stress should be more complicated to identify as a mental disorder, depression was the most difficult when analyzed on the EMOTHAW paper [73].

The DAS Scale used in the EMOTHAW paper was used to identify depression with low motivation and self-esteem items [73].

Anxiety

Schlenker and Leary definition of anxiety is: "A cognitive-affective response characterized by physiological arousal (nervous system activation) and apprehension regarding a potentially negative outcome that the individual perceives as impending" [97]. Key symptoms of identifications are, according to [8] and [104] are: sweating, palpitations, tachycardia, rapid breathing, among others, that help our medics to identify it on patients. Spielberg [104] cites Freud on these observations in anxiety being prevalent of psychiatric disorders.

Anxiety is a condition that makes body and mind to be alert all the time, which consumes energy and causes the individual to be tired and irritated [104].

Costello and Comrey [28] and Beck, *et al.* [9] designed a scale in which Anxiety and Depression could be identified properly. The former scales correlate a 0.50 in comparison to the already-existing scales the authors based their work on, and the latter (the Beck Anxiety Inventory (BAI)) had the same correlation with the BDI. The BAI consists of a 21-item scale, just like its depression scale counterpart.

For the Anxiety state, the DAS Scale items used were fear and perceived panic [73].

Stress

Hans Selye defines stress as: *the nonspecific response of the body to any demand* [100]. Another definition is: *emotional experience accompanied by predictable biochemical, physiological, and behavioral changes* [7].

Stress has aroused in the medical conversations, being the cause of many diseases nowadays. Compared to the other two states, this one is a natural process which every individual feels when faced with a complicated situation in his day-to-day life, according to [100], even animals, bacteria, and plants experience stress. Although the frequency in which it has been felt has changed, it has increased up to a point in which the hormones secreted by the body overflow it, causing diseases such as Diabetes, and cardiovascular diseases [25], this means that it becomes a mental disorder that expands to the rest of the body.

For this reason, stress is considered the most difficult emotional state out of all three of them.

Stress also has items for which it can be identified in the EMOTHAW paper, which are: tension and irritability related items [73].

The DAS Scale

The DAS Scale is a psychometric test to be an efficient measure for these three states [45], designed by Syd Lovibond and Peter Lovibond at the University of New South Wales in 1995 [76]. Consists of 42 items of a 1-4 scale that described the frequency of the symptoms in the individual. All the results can be analyzed per state. Table 4.1 shows all five possible ranges for each emotional state measured in the test. Nonetheless, it is out of the scope of this work to consider all of them.

In the EMOTHAW methodology, The DAS Scale is used to label each task according to the questionnaire results. It was performed to 129 users. The Lovibond tool used in this work was assessed by Henry and Crawford in 2003 for the English language [29].

Table 4.1: Complete DASS Score Range (taken from Likforman-Sulem [73])

	Depression	**Anxiety**	**Stress**
Normal	0-9	0-7	0-14
Mild	10-13	8-9	15-18
Moderate	14-20	10-14	19-25
Severe	21-27	15-19	26-33
Extremely Severe	28+	20+	34+

The complete range in table 4.1 is out of the scope of this work. It provides complexity that would not be able to be handled in the duration of this program. Nonetheless, Analysis and results are based on a binary and ternary classification (both of which were obtained from table 4.1. For the binary case, could be also called the simplest case, there was a new set of ranges for the states (shown in table 4.2), where all ranges, except *Normal*, were grouped; This redistribution diminished the imbalance problem within the database. This methodology was proposed by Nolazco, *et al.*, where they worked out different ideas presented in this work [86].

Table 4.2: Independent Binary Classification ranges. [86]

	Depression	Anxiety	Stress
Normal (0)	0-9	0-7	0-14
With Disease(1)	10-28+	8-20+	15-34+

In table 4.2 we can observe the new distribution of ranges for a binary case, as used in the *Emotional State Recognition Performance Improvements on a Handwriting and Drawing Task* paper [86].

The distribution in table 4.2 is:

- **Depression**: Someone who took the test and got a score higher than 9 on the depression questions is considered depressed.
- **Anxiety**: Someone who took the test and got a score higher than 7 on the anxiety questions is considered anxious.
- **Stress**: Someone who took the test and got a score higher than 14 on the stress questions is considered stressed.

Table 4.3: Ternary Classification ranges.

	Depression	Anxiety	Stress
Normal (0)	0-9	0-7	0-14
Mild (1)	10-13	8-9	15-18
Severe (2)	14-28+	10-20+	19-34+

4.1.2 Tasks

Emotion recognition from handwritten activities is relatively new in the field of machine learning [86], but there are validated and proven psychological written/drawn tests that identify underlying causes for certain behavior, specially for depression and anxiety [73]. They are also used for medical reference: In cases of brain disorders (e.g., Parkinson's, Alzheimer, Dementia, Schizophrenia) there is proof that patients tend to deteriorate motor functions and can be tracked in handwriting [38].

The seven tasks used in this work were chosen from three different tests that are used to identify cognitive problems, they are: The Mini Mental State Examination (MMSE) [40], the House-Tree-Person (HTP) [64], and the Clock Drawing Test (CDT). The last one was then reformulated as the ClockMe System, a computational version of the test [62].

The original MMSE test is a 30-point test measures cognitive functions like: registration, attention, language, ability to follow instructions, among others. The relevant tasks that this test has to this work are: pentagon drawing, writing sentences and words [40].

The HTP is a clinical test that consists of drawing a house, tree, and a person and to answer brief questions to complement the drawing results. The test could be also used to identify brain damage [64].

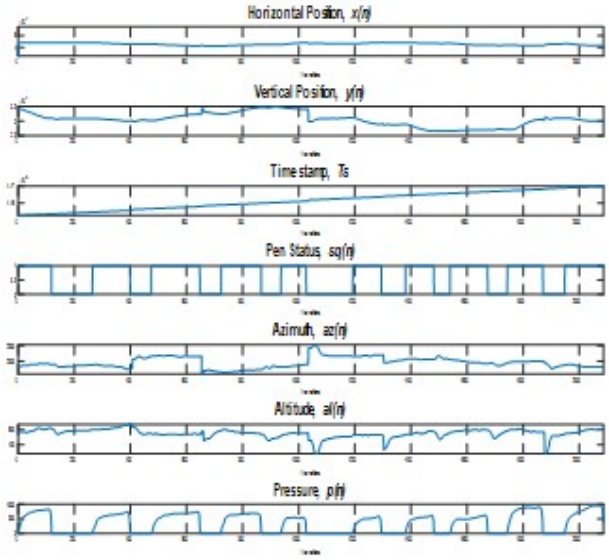
In The third test, the Clock Drawing Tests/ClockMe, the participants have to draw a clock with 12 digits (1 through 12), and also place the arrows reading 11:50. Its application is best on Parkinson's disease, but the same principle is used for the emotional states present in this work.

With these tests the EMOTHAW database was created, in particular: Clock, pentagon, circle, and house drawings, and word and sentence writing. The database creation was possible thanks to the advancements on tablets and the software that comes with them. Their precision and accuracy are top-notch, specially the tablet INTOUS WACOM series 4 and the Intuos Inkpen [73]. Not only that, these improvements on tablets make the case for implementing these and many psychological tests on a digital platform, which benefits are for a faster and better diagnostic [73].

From the recollection of these tasks, the tablet gathered seven different initial attributes, which are: Horizontal Position $x(n)$, Vertical Position $y(n)$, Time stamp, T_s, Pen Status $sq(n)$, Azimuth $az(n)$, Altitude $al(n)$, and Pressure $p(n)$. In figure 4.1 we can observe the behavior of the time series generated from a user doing the pentagon drawing task.

In figure 4.1 there are seven time series plots taken from a user drawing two pentagons (Figure 4.2), from which new attributes/features will be generated (more in section 2.3). These are seven initial parameters for each task; there are several things to notice: Time stamp T_s increases throughout any task, Pressure $p(n)$ and Pen Status $sq(n)$ are dependent in that if the latter is zero, the former as well [73].

Figure 4.1: Example of the seven initial attributes gathered from the tablet of a given task (in this particular case, the pentagon drawings)



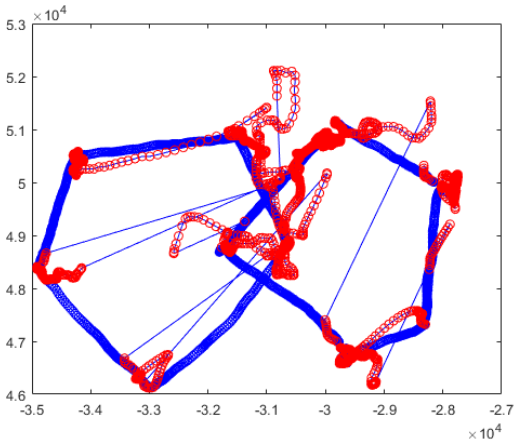
In the next figures (i.e., from 4.2 to 4.8) we can observe red scribbles along each drawing or words, except for the circles, and it is because the tablet also tracked the pen's movement while it was not touching the tablet (specially if it was nearby it, 1 cm of distance). We call this the pen status of tasks, it's an on-off signal that represents whether the user was making contact with the tablet or not. In other words, the red dotted-like scribbles are off-pen status (0), and blue lines are on-status (1) [73]. It is key to notice that the users were not told that the

tablet was tracking also when they lift the pen, this adds for a more natural way of performing the seven tasks at hand.

Task 1: Two Pentagon Drawing

The pentagon drawing task is part of the Mini Mental State Examination test and can be seen in figure 4.2.

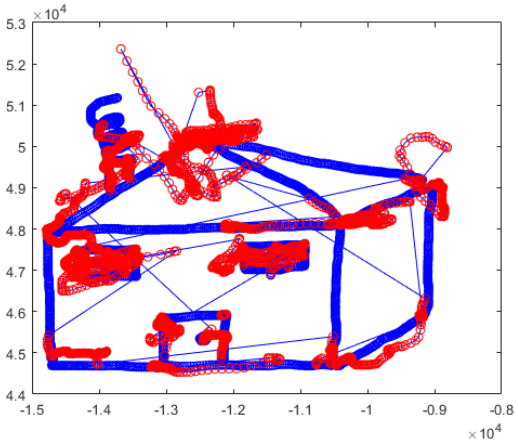
Figure 4.2: Pentagon Drawing Example with pen status



Task 2: House Drawing

The House Drawing task was taken from the House-Tree-Person test and it is shown in figure 4.3.

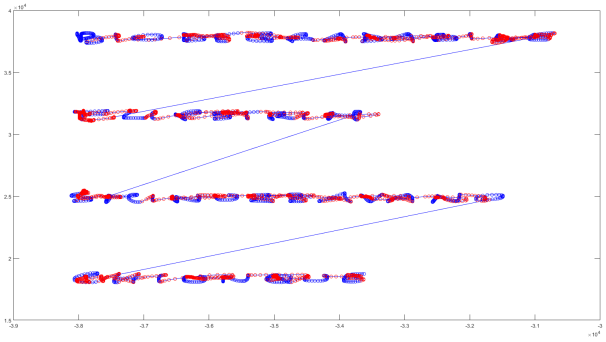
Figure 4.3: House Drawing Example with pen status



Task 3: Four Italian Word Writing

This third task consisted of writing four different Italian words in Upper case. They do not have any positive nor negative connotation, according to PhDs in the Psychology department of the Seconda Università di Napoli. The words are, in order (figure 4.4): BIODEGRADABILE (biodegradable), FLIPSTRIM (flipstrim), SMINUZZAVANO (to crumble), CHIUNQUE (anyone) [73].

Figure 4.4: Italian Words Example with pen status



Task 4 and 5: Loops with Left and Right Hand

The loops drawings are the simplest of all the tasks (figures 4.5 and 4.6), but still provide good information, specially how steady the user can draw them and analyze that for motor activity in their muscles. Their dominant hand can also be determined by looking in the traces, by the looks of it, the user taken for illustration seems to be a right-handed person.

Figure 4.5: Loops with Left Hand Drawing Example with pen status

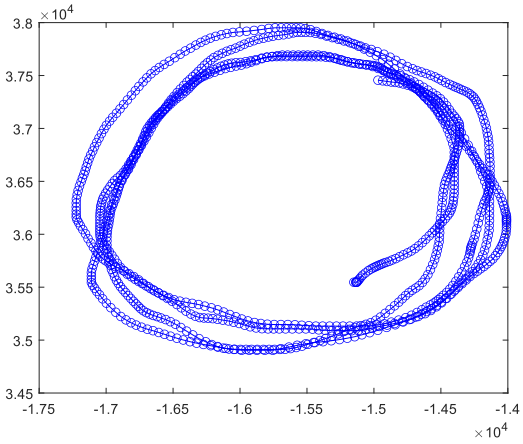
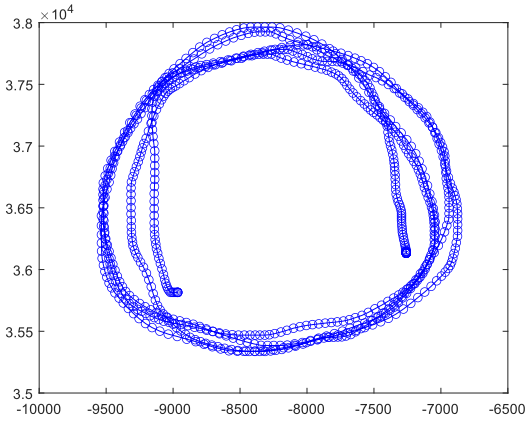


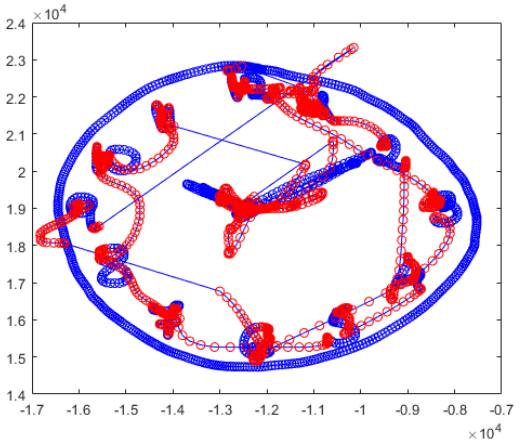
Figure 4.6: Loops with Right Hand Drawing Example with pen status



Task 6: Clock Drawing

The Clock Drawing task is another that was taken from a psychological test: Clock Drawing Test. There is an example in figure 4.7.

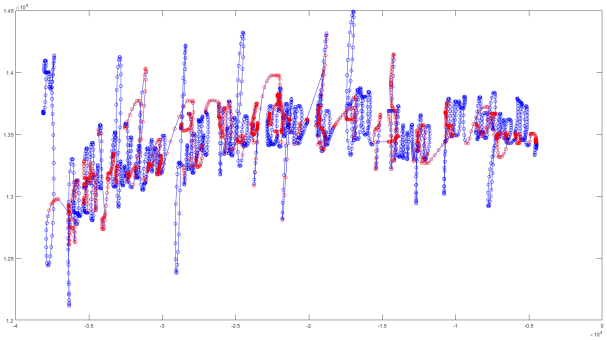
Figure 4.7: Clock Drawing Example with pen status



Task 7: Sentence Writing

This task follows the same methodology as the *words* task, but this one was written in cursive style. There is an example of it in figure 4.8.

Figure 4.8: Sentence Writing Example with pen status



4.1.3 Users

The population considered for building this database was between the ages of 21 and 32 years old. They were enrolled either in the Masters or B.S. program in Psychology in the Università della Campania "Luigi Vanvitelli" in Italy [86]. In total, they were 129 people (71 female and 58 male). This type of database is an accomplishment on its own, because it is not common to have one available to anyone with access to the internet, and the construction in itself is not an

easy task to do [73]. The age parameter allows controlling the degree in which each subject has or not a given emotional state. In other words, there are variability attributed to age that might have added more complexity and bias to the database.

They first took the Italian DASS test, and then did the seven task test design for the purpose of building the EMOTHAW database [86].

The overall database turned out to be an Imbalanced case for all three emotional states. In figure 4.9, for Anxiety and Stress the distribution is the same, and it resembles more a balance problem, whereas Depression has a clear imbalance, there are significant more instances in class 0 than in 1. Nonetheless, in figure 4.10 it is more pronounce, and this variation of the problem will be, then, more difficult to generalize in the Machine Learning problems at hand, because now all three Emotional States are imbalanced (class 1 from table 4.2) got split up into two new classes in table 4.3).

Figure 4.9: Class distribution of the binary classification (see table 4.2)

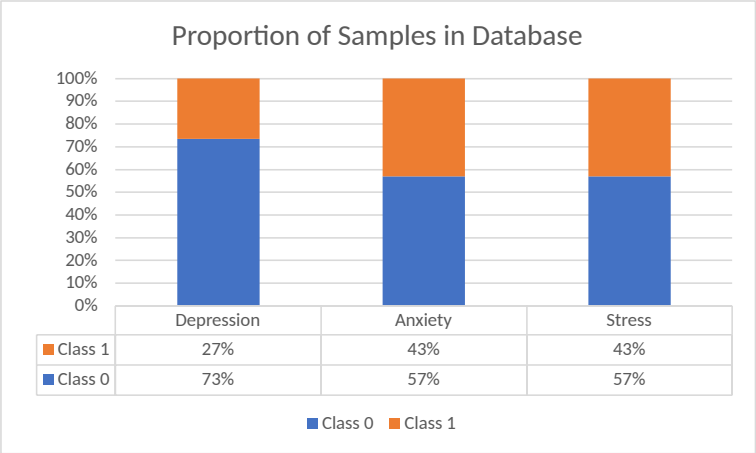
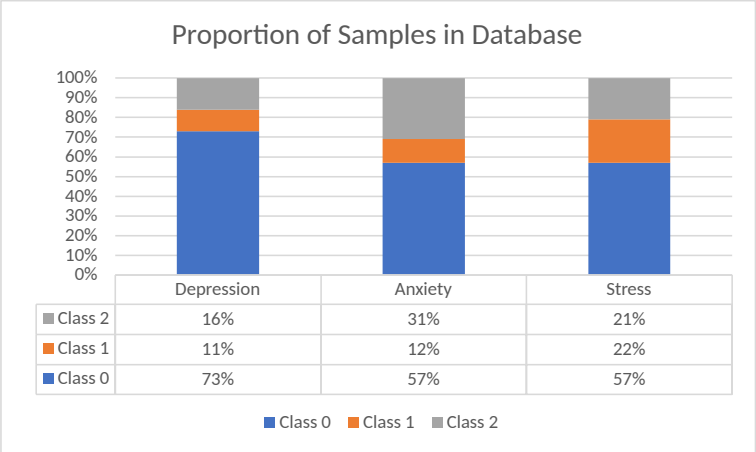


Figure 4.10: Class distribution of the ternary classification (see table 4.3)



4.1.4 Baseline Results

In table 4.4 is a brief summary of the results obtained by Likforman, *et al.* in their publication of the EMOTHAW database [73]. Their accuracy performance provides room for improvement, as well as, an understanding of the difficulty of the task at hand. They decided to only focus on a binary problem, since their main objective was the actual building of the database. Their Machine Learning analysis is deep enough to visualize the dimension of the problem, though.

Table 4.4: Model Results separated by original Feature Type (Writing, Drawing, and combined). [73]

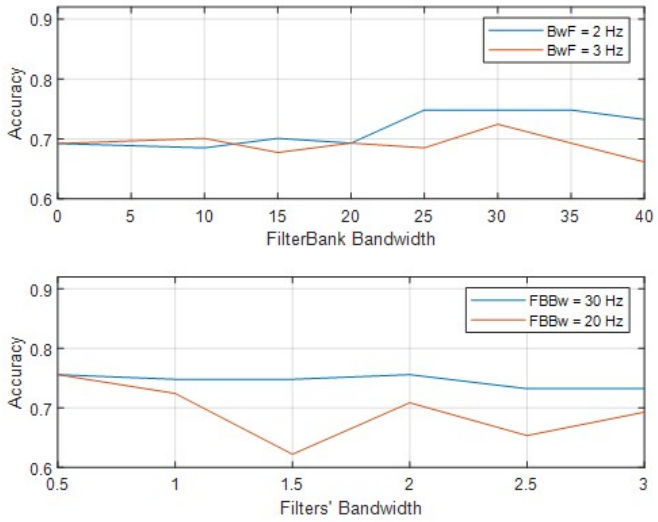
Model	Feature Type	Accuracy (%)	Sensitivity (%)	Specificity (%)
Depression	Writing	67.8	6.7	89.7
Depression	Drawing	71.6	19.1	90.4
Depression	Drawing & Writing	71.2	12.6	92.2
Anxiety	Writing	56.3	38.2	69.9
Anxiety	Drawing	60.5	46.2	71.2
Anxiety	Drawing & Writing	60	45	71
Stress	Writing	51.2	32.3	65.3
Stress	Drawing	60.1	44.5	71.7
Stress	Drawing & Writing	60.2	41	74.5

4.2 First Phase

In this phase, the main objective is to improve results based primarily on spectral and cepstral features, as well as the understanding of the hyperparameters oTh and iTh of the *modified Fast Correlation-Based Filtering* method, and the others in the Filter section ($fbbw$, fbw, i_f, ov). The latter parameters are more complicated to understand their impact among the myriad of possibilities, that is why they were limited to the values presented in section 3.6.

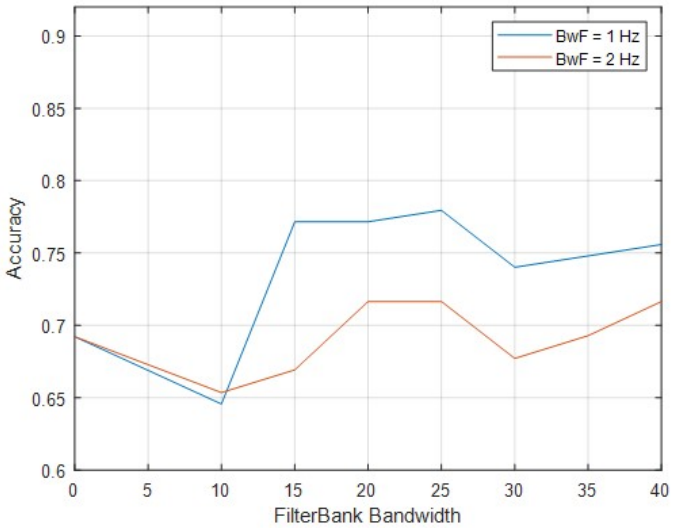
In figures 4.11 and 4.12 there are the results of three different experiments conducted for the purpose of understanding the hyperparameters, and how accuracy behaved when modifying them. For the sole purpose of the aforementioned reasons, H2O was not used, only Support Vector Machine, rather, and only depression for written tasks were considered, just to diminish uncontrolled variables. These experiments prove to be of importance because of, 1) the performance behavior is not dependent on the more magnitude the parameters have, there is a range of values that the model performs on, rather. 2) Computational Run-time is considerable, and performance results do not justify it. Lastly, 3) The myriad of possibilities is extensive and should not be a priority for subsequent experiments. Though, H2O provides for better and efficient ways of testing, hyperparameters of this sort are independent [86].

Figure 4.11: Accuracy Results when changing $fbbw$ and fbw, respectively.



Description: a. Represents the results for depression considering only written tasks when changing $fbbw$ from 0 to 40 Hz and fixing fbw or BwF to 2 and 3 Hz. b. The same follows in this graph, but the other way around, varying fbw from 0.5 to 3 Hz and fixing $fbbw$ to 20 and 30 Hz. [86].

Figure 4.12: Accuracy Results when changing $fbbw$ and fixing fbw, with a Filter in-between separation of 2 and 3 Hz.



Description: Blue and Red graph have a variation in $fbbw$ from 0 to 40 Hz, and have fix values for fbw/BwF of 1 and 2 Hz, respectively. The main difference between figure 4.11 is the separation between filters in the Filterbank, in figure 4.11 it was fixed to 1 Hz, whereas here it goes for 2 and 3 Hz, respectively [86].

Figure 4.13: Change in Number of features and varying oTh, fixed iTh.

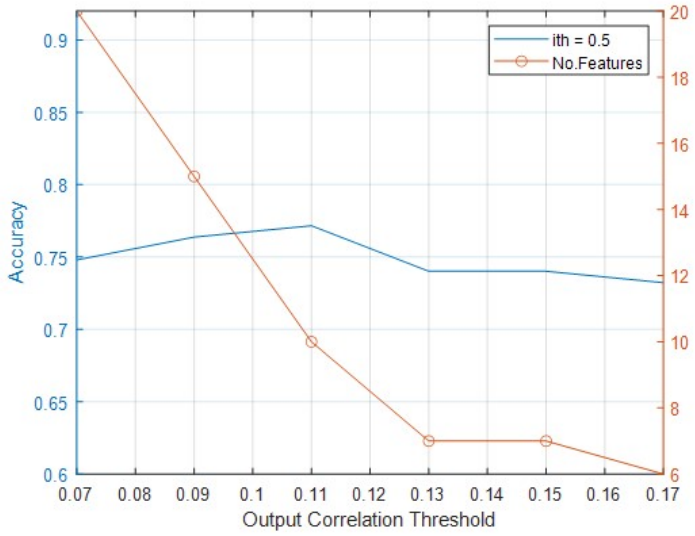
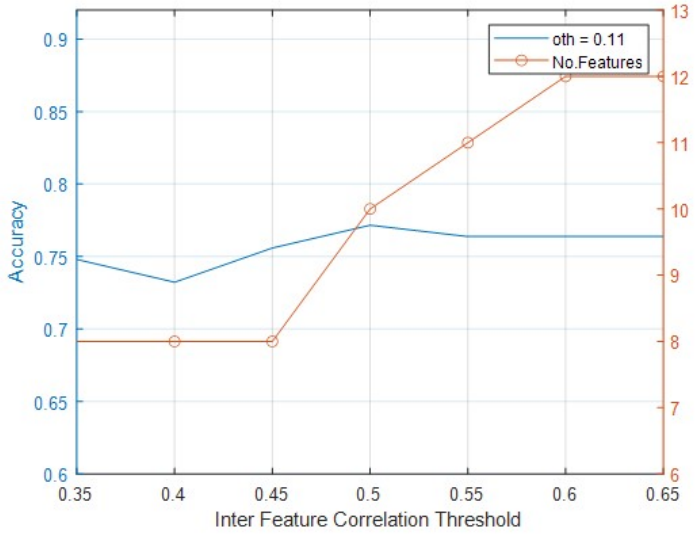


Figure 4.14: Change in Number of features and varying iTh, fixed oTh.



Figures 4.13 and 4.14 are fairly similar in construction, but not in output, what they show is the impact of each separate threshold value in both accuracy and the number of resulting attributes after applying *mFCBF* algorithm [86]. In particular, figure 4.13 shows the behavior of the number of attributes when oTh increases. The range goes from 0.07 to 0.17 with an increase of 0.01, and as it is bigger, the number of attributes decreases, having an inverse relationship between both parameters. For figure 4.14, the relationship is a direct-type, meaning when iTh increases, the number of attributes does as well.

In Table 4.5 there is the complete information regarding the performance of all the task-type data sets, as well as a brief comparison to the original results in Likforman, *et al.* [73]. Anxiety and Stress were the two emotional states with the most improvement out of the three,

with 20%. Depression was not that behind, but 8% is not big enough, although of the three it is still the highest accuracy performance. Overall, there was a 15% improvement, averaging all three emotional states' results.

Table 4.5: Overall results of the Experiments conducted in the hyperparameter scheme in *mFCBF* and signal processing [86].

Emotional State	Task Type	Baseline [73] (%)	Max (%)	Max Performance (%)
Depression	*Writing*	69.22	80.31	16
	Drawing	72.48	75.59	4
	Both	71.47	74.01	4
Average		**71.06**	**76.64**	**8**
Anxiety	*Writing*	56.59	68.5	21
	Drawing	58.68	67.71	14
	Both	58.53	72.44	24
Average		**57.93**	**69.29**	**20**
Stress	*Writing*	50.54	67.71	34
	Drawing	59	67.71	15
	Both	61.24	70.07	14
Average		**56.93**	**68.5**	**20**
Overall Average		61.97	71.47	15

This methodology followed a Leave-One-Out validation technique, followed with White Noise Data Augmentation. The Machine Learning of choice was a Support Vector Machine with a radial basis kernel in a Binary classification problem [86].

Values for each hyperparameter varies and most likely depend on the type of emotional state one is working with, but overall, from the experiments conducted in the aforementioned results, one can say:

1. $fbbw$ best values are around 25 and 30 Hz.
2. fbw best values are 1 and 2 Hz.
3. oTh best performing values are around 0.11.
4. iTh best performing values go from 0.45, 0.5, 0.55.

Lets remember that these values are based on the experiments performed with the framework previously mentioned. They serve as a guide and not as the absolute truth for further testing.

4.3 Second Phase

The *exploratory* phase was done in first Phase, section 4.2. In there, several ideas were tested and most of them were carried on to the core of experiments conducted in a so-called *Second Phase*. From now one, the Machine Learning modeling used for test/training will be a

H2O framework, which consists of eleven different models that will work together for improving performance in the upcoming experiments, improving both time and computational efficiency. Said this, there is a continuation in testing with the Binary classification problem, since there was important room for improvement. From there, once the binary models were shaped to important results, finally the Ternary Classification problem arises as the new challenge for testing new and old ideas, because as it will be seen, some ideas do not carry the same performances as once did.

The main differences between *Phase Two* and the *First* one are:

1. Attribute generation techniques were widened in the *Second Phase*, adding, *Kinematic*, *Statistics*, and refinement into *Time* features.

2. Filterbank hyperparameters were fixed to the better performing ones from *Phase One*:

   (a) $fbbw = 30Hz$

   (b) $fbw = 1Hz$

   (c) $i_f = 0.5Hz$

   (d) $ov = 0\%$

3. Threshold values were variables, but with different ranges:

   (a) $oTh = [0, 0.02, 0.04, ..., 0.18, 0.2]$

   (b) $iTh = [0.2, 0.3, ..., 0.9, 1]$

4. Both Binary and Ternary classification problems were considered.

5. For *dimensionality reduction* and *Feature selection,* a novel procedure was performed, combining PCA and *mFCBF* in the model.

6. Leave-Percentage-Out instead of Leave-One-Out.

7. H2O framework used instead of individual models.

For this phase, several models are compared with each other and the *PCA+mFCBF* model in particular to visualize the importance of this novel idea. In total, there are six different models for the binary problem (including baseline [73] and the *Phase One* model [86]) and seven for the ternary classification problem. Notice that values are not exactly the same as those presented previously, but they are within confidence intervals.

Table 4.6: A brief comparison among models based-off accuracy results for the Binary classification problem for emotional state recognition.

Emotional State	*Model 1* (%)	*Model 2* (%)	*Model 3* (%)	*Model 4* (%)	*Model 5* (%)	*Model 6* (%)
Depression	71.47	74.01	80.7	81.57	92.98	100
Anxiety	58.53	72.44	71.93	75.43	88.6	100
Stress	61.24	70.07	66.67	71.92	89.47	100

Table 4.7: A brief comparison among models based-off accuracy results for the Ternary classification problem for emotional state recognition.

Emotional State	*Model 2* (%)	*Model 3* (%)	*Model 4* (%)	*Model 5* (%)	*Model 6* (%)
Depression	73.68	74.56	77.19	81.57	82.45
Anxiety	57.89	50.87	57.89	71.92	72.8
Stress	57.89	47.36	54.38	65.78	74.56

The difference in models in tables 4.6 and 4.7 are because the ternary problem did not have a "Baseline" perse, this work can be considered it, after all. The rest of the models represent various settings that exemplify how each model contributes to performance, all the way to *Model 6*, which is the *PCA+mFCBF* model. Furthermore, the following list discloses the type of features and feature selection techniques which were used for each model:

1. *Model 1*: TF (baseline) [73].
2. *Model 2*: $Spectral \cup Cepstral$ Features + *mFCBF* [86]
3. *Model 3*: Full range of Features + no Feature Selection technique.
4. *Model 4*: Full range of Features + *PCA* or O^E (see section 3.3.1)
5. *Model 5*: Full range of Features + *mFCBF*
6. *Model 6*: Full range of Features + *PCA+mFCBF*

For both the Binary and Ternary classification problems, all comparisons were done equally, excepting the seventh model in the latter problem. As we can see from the results sheets, the Binary problem reach a 100% accuracy in *Model 6*, which suffice for the halt of furthering brainstorming new ideas for the improvement of results. Nonetheless, in the Ternary problem, the story goes different. All models were not performing as well as in the Binary problem. Although, the difficulty of three classes is understood, how it came to be in the end, was not expected. But all things consider, *Model 6* holds for their impressive results considering that, in ternary problems, the average result for a class is 33% taking into account all others (instead of the 50% from a binary problem).

Chapter 5

Shared Thoughts and Conclusions

5.1 Discussion & Further Thoughts

Emotional State recognition as is presented is a young field, where the possibilities and ideas to try are uncountable. From this work we can say that the Binary Classification problem is solved, although one can propose to find a more efficient way to perform testings on each model, but that goes beyond the scope of the research objectives from the author's perspective. Firstly because, the opportunities given by ternary and higher classification problems are of a bigger weight than those of efficiency.

The first important challenge was that of the distribution of classes, an imbalance data set problem. In the beginning, modeling was a big concern and performance was below 50%! But, different imbalance-solve methods, whether implemented in libraries e.g., *SMOTE* or with signal processing-like techniques, e.g., *White Noise*, help to synthetically create reliable more observations to compensate this issue, and to provide the reader and fellow researcher with viable implementations and solutions to the emotional state recognition problem. Once this issue was dealt with, the next one was the creation of diversified features.

Emotional State Recognition from a database like *EMOTHAW* (i.e., open-source and with seven time series as input attributes) is quite a trial, but the resemblance with traditional signals was the key to the success of this work. Nonetheless, the short duration of the raw characteristics could be a problem at the beginning, but as testing and trying new ideas went on, it was not a problem after all. From here, and with smaller sampling rates than other applications, *spectral* and *cepstral* features could be drawn. Then, from literature reviewing similar applications, *statistical* [33], *kinematic* [56], and different *Time* [86] features could be added to the final users' vector.

From the first testing phase, there was a different approach than that of the second phase. It follows and resembles most to the work done in the original *EMOTHAW* paper [73], but it was due for comparison purposes. So, in the result table 4.5 shows performance for all three emotional states, as well as a Task differentiation (Column **Task Type**). From this table, the reader can observe that the best task type is *writing* with an 80.31% accuracy performance. For the other two emotional states, their best resulting model came from considering both types of tasks. Throughout this work, there is a similarity in behavior from *Stress* and *Anxiety*, they both underperform in comparison to *Depression*.

The Binary classification problem can be said to be *solved* because, as table 4.6 shows,

in particular *Model 6*, 100 % accuracy is reach for all three emotional states, which is an astonishing accomplishment in its own. Although, it took six different models for it to be as is. As mentioned before, *Model 1* is the table results from the baseline paper by Likforman, *et al.* [73], *Model 2* are the results from the first phase testing [86], which resulted in a published article in the *IEEE ACCESS* journal, Models 3 through 6 share the same feature vector, but differentiate by the feature selection method. The combination of *PCA* and *mFCBF* open-up important ideas, such as: testing with different combinations of dimensionality reduction and feature selection techniques, or even be creative and create dataframes that can excel on maximizing information gain and minimizing attribute number.

Table 4.7 shows a different story than that in table 4.6. For starters, *Model 1* is not found in this table, because the *EMOTHAW* research paper did not contemplate ternary classification, and one can say this research thesis are the baseline results for it. Models 2 through 6 remain the same as its binary counterpart, nevertheless, the results are not as high as expected (at some point, results in the binary classification problem were impressive enough to think that they will transition just as well to higher classes, but it was not the case). This does not mean that the results were disappointing, however, it only means that there is still room for improvement. Which leads to a discussion about further ideas that could be beneficial to try them on this research work.

One of the research questions presented in the introduction of this work posed the viability of signal processing techniques to be viable for feature generation, and as seen in *Phase One*, they hold for expectations, and even further, in *Phase Two* they hold their own as principal feature generation from the five different ones proposed. Following up with research questions, the *EMOTHAW* database is interesting in its own way, firstly because of how it was created, secondly, because of the type of information it provides; even though it presents only seven initial raw attributes, one can draw multiple and different types of features from them; so, to answer the question: *Does the current database have enough information to work with? If not, how to successfully broaden it?* It does, but not on its own. It has to undergo a necessary feature extraction process to reorganize all the files into one single Dataframe.

5.2 Future Work - Subsequent Phases

The transition made from a binary classification problem to a ternary one proved that the ideas that worked on the former were not necessarily transferable to the latter. Nonetheless, it does not mean either that they should be discarded for the rest of multi-class problems, it only means that they should not be priority on testing; Here is where the H2O library comes in handy, it allows for multiple model testings, including parameter tuning, without sacrificing computational efficiency (compared to building those models individually), so those testing can be done with relative ease.

Said this, there are several ideas that are worth mentioning for further development of this research line of work, one of which is the exploration of segmenting each original time series into N-pieces and consider them as separate unique input signals, i.e., let s^t be the complete input signal of a task t, and s^t_n be the n-th section of s^t, where $s^t_n = \frac{s^t}{n}$, Its justification lies in the different stroke patterns a person could have when suffering an emotional state condition (i.e., muscle tension, Parkinson's-like behavior, etc. [77]).

Deep Learning and Neural Networks have proved to be important learning model techniques that have wide applications across many research lines and private businesses. One of the advantages of neural networks is the flexibility of architecture creation. There are plenty of possibilities when building them, and their integration to a complex problem might be the way to go, specially because of how they handle input and outputs.

Anomaly or Outlier detection is also a feasible research line to explore, because there have been important advancements on the algorithms and their performance with a wide number of different databases, specially *Bagging-Random Miner* [18]. In this research work, all modeling done was supervised, meaning there is a defined target value for each observation in the dataframe, with Outlier Detection one can test how semi-supervised and non-supervised modeling will behave in this research work.

Use all the original attributes in the generation of spectral and cepstral feature row vectors as well as expanding the range of testing for the four main filter parameters ($fbbw$, fbw, i_f, and ov), specially the last two. One limitation for these proceedings is that it takes important and considerable run-time to produce the output data sets, which might mean that the effort is not worth the extra percentage-increase on performance, although it is a strong belief that, once the optimized parameters are found, the performance will rise considerably. For this reason, the idea is still considered for future work.

All of these ideas are thought to be implemented in ternary and higher classification problems, even in compound problems (multiple target variables).

Bibliography

[1] ABANDAH, G., AND ANSSARI, N. Novel moment features extraction for recognizing handwritten arabic letters. *Journal of Computer Science 5*, 3 (2009), 226.

[2] ABDI, H., AND WILLIAMS, L. J. Principal component analysis. *Wiley interdisciplinary reviews: computational statistics 2*, 4 (2010), 433–459.

[3] ALLEN, D. M. The relationship between variable selection and data agumentation and a method for prediction. *technometrics 16*, 1 (1974), 125–127.

[4] ANARTE, M. T., LÓPEZ, A. E., MAESTRE, C. R., AND ZARAZAGA, R. E. Evaluación del patrón de conducta tipo c en pacientes crónicos. *Anales de Psicología/Annals of Psychology 16*, 2 (2000), 133–141.

[5] ANDERSEN, P. K., AND GILL, R. D. Cox's regression model for counting processes: a large sample study. *The annals of statistics* (1982), 1100–1120.

[6] BARANDELA, R., SÁNCHEZ, J. S., GARCIA, V., AND RANGEL, E. Strategies for learning in class imbalance problems. *Pattern Recognition 36*, 3 (2003), 849–851.

[7] BAUM, A. Stress, intrusive imagery, and chronic distress. *Health psychology 9*, 6 (1990), 653.

[8] BECK, A. T., AND BEAMESDERFER, A. *Assessment of depression: the depression inventory.* S. Karger, 1974.

[9] BECK, A. T., AND STEER, R. Beck anxiety inventory (bai). *Überblick über Reliabilitäts-und Validitätsbefunde von klinischen und außerklinischen Selbst-und Fremdbeurteilungsverfahren 7* (1988).

[10] BENDAT, J. S., AND PIERSOL, A. G. *Engineering applications of correlation and spectral analysis*. Wiley, 2013.

[11] BENGIO, Y., COURVILLE, A., AND VINCENT, P. Representation learning: A review and new perspectives. *IEEE transactions on pattern analysis and machine intelligence 35*, 8 (2013), 1798–1828.

[12] BIRD, J. J., FARIA, D. R., EKÁRT, A., AND AYROSA, P. P. From simulation to reality: Cnn transfer learning for scene classification. In *2020 IEEE 10th International Conference on Intelligent Systems (IS)* (2020), IEEE, pp. 619–625.

[13] BISHOP, C. M. Pattern recognition. *Machine learning 128*, 9 (2006).

[14] BLEHA, S. A., AND OBAIDAT, M. S. Dimensionality reduction and feature extraction applications in identifying computer users. *IEEE Transactions on Systems, Man, and Cybernetics 21*, 2 (1991), 452–456.

[15] BOTTEMA, O., AND ROTH, B. *Theoretical kinematics*, vol. 24. Courier Corporation, 1990.

[16] BOUDRAA, A., CEXUS, J., AND SAIDI, Z. Emd-based signal noise reduction. *International Journal of Signal Processing 1*, 1 (2004), 33–37.

[17] BREIMAN, L. Random forests. *Machine learning 45*, 1 (2001), 5–32.

[18] CAMIÑA, J. B., MEDINA-PÉREZ, M. A., MONROY, R., LOYOLA-GONZÁLEZ, O., VILLANUEVA, L. A. P., AND GURROLA, L. C. G. Bagging-randomminer: A one-class classifier for file access-based masquerade detection. *Machine Vision and Applications 30*, 5 (2019), 959–974.

[19] CAO, L., CHUA, K. S., CHONG, W., LEE, H., AND GU, Q. A comparison of pca, kpca and ica for dimensionality reduction in support vector machine. *Neurocomputing 55*, 1-2 (2003), 321–336.

[20] CHAMBERS, J. M. Linear models. In *Statistical models in S*. Routledge, 2017, pp. 95–144.

[21] CHANG, Y.-W., HSIEH, C.-J., CHANG, K.-W., RINGGAARD, M., AND LIN, C.-J. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research 11*, 4 (2010).

[22] CHAWLA, N. V., BOWYER, K. W., HALL, L. O., AND KEGELMEYER, W. P. Smote: synthetic minority over-sampling technique. *Journal of artificial intelligence research 16* (2002), 321–357.

[23] CHEN, T., AND GUESTRIN, C. Xgboost: A scalable tree boosting system. In *Proceedings of the 22nd acm sigkdd international conference on knowledge discovery and data mining* (2016), pp. 785–794.

[24] CHOU, Y.-M., MASON, R. L., AND YOUNG, J. C. The control chart for individual observations from a multivariate non-normal distribution. *Communications in statistics-Theory and methods 30*, 8-9 (2001), 1937–1949.

[25] CHRISTENSEN, A. V., DIXON, J. K., JUEL, K., EKHOLM, O., RASMUSSEN, T. B., BORREGAARD, B., MOLS, R. E., THRYSØE, L., THORUP, C. B., AND BERG, S. K. Psychometric properties of the danish hospital anxiety and depression scale in patients with cardiac disease: results from the denheart survey. *Health and quality of life outcomes 18*, 1 (2020), 1–13.

[26] CIREGAN, D., MEIER, U., AND SCHMIDHUBER, J. Multi-column deep neural networks for image classification. In *2012 IEEE conference on computer vision and pattern recognition* (2012), IEEE, pp. 3642–3649.

[27] CORTES, C., AND VAPNIK, V. Support-vector networks. *Machine learning 20*, 3 (1995), 273–297.

[28] COSTELLO, C., AND COMREY, A. L. Scales for measuring depression and anxiety. *The Journal of psychology 66*, 2 (1967), 303–313.

[29] CRAWFORD, J. R., AND HENRY, J. D. The depression anxiety stress scales (dass): Normative data and latent structure in a large non-clinical sample. *British journal of clinical psychology 42*, 2 (2003), 111–131.

[30] DASH, M., AND LIU, H. Feature selection for classification. *Intelligent data analysis 1*, 1-4 (1997), 131–156.

[31] DE STEFANO, C., FONTANELLA, F., IMPEDOVO, D., PIRLO, G., AND DI FRECA, A. S. Handwriting analysis to support neurodegenerative diseases diagnosis: A review. *Pattern Recognition Letters 121* (2019), 37–45.

[32] DIEBOLD, F. X. *Elements of forecasting*. Citeseer, 1998.

[33] DROTÁR, P., MEKYSKA, J., REKTOROVÁ, I., MASAROVÁ, L., SMÉKAL, Z., AND FAUNDEZ-ZANUY, M. Analysis of in-air movement in handwriting: A novel marker for parkinson's disease. *Computer methods and programs in biomedicine 117*, 3 (2014), 405–411.

[34] DUROVIC, Z. M., AND KOVACEVIC, B. D. Robust estimation with unknown noise statistics. *IEEE Transactions on Automatic Control 44*, 6 (1999), 1292–1296.

[35] DWORK, C., FELDMAN, V., HARDT, M., PITASSI, T., REINGOLD, O., AND ROTH, A. The reusable holdout: Preserving validity in adaptive data analysis. *Science 349*, 6248 (2015), 636–638.

[36] ERICKSON, N., MUELLER, J., SHIRKOV, A., ZHANG, H., LARROY, P., LI, M., AND SMOLA, A. Autogluon-tabular: Robust and accurate automl for structured data. *arXiv preprint arXiv:2003.06505* (2020).

[37] FAUNDEZ-ZANUY, M., FIERREZ, J., FERRER, M. A., DIAZ, M., TOLOSANA, R., AND PLAMONDON, R. Handwriting biometrics: Applications and future trends in e-security and e-health. *Cognitive Computation 12*, 5 (2020), 940–953.

[38] FAUNDEZ-ZANUY, M., HUSSAIN, A., MEKYSKA, J., SESA-NOGUERAS, E., MONTE-MORENO, E., ESPOSITO, A., CHETOUANI, M., GARRE-OLMO, J., ABEL, A., SMEKAL, Z., ET AL. Biometric applications related to human beings: there is life beyond security. *Cognitive Computation 5*, 1 (2013), 136–151.

[39] FEURER, M., KLEIN, A., EGGENSPERGER, K., SPRINGENBERG, J. T., BLUM, M., AND HUTTER, F. Auto-sklearn: efficient and robust automated machine learning. In *Automated Machine Learning*. Springer, Cham, 2019, pp. 113–134.

[40] FOLSTEIN, M. F., FOLSTEIN, S. E., AND MCHUGH, P. R. "mini-mental state": a practical method for grading the cognitive state of patients for the clinician. *Journal of psychiatric research 12*, 3 (1975), 189–198.

[41] FRIEDMAN, J. H., AND POPESCU, B. E. Predictive learning via rule ensembles. *The Annals of Applied Statistics 2*, 3 (2008), 916–954.

[42] GARDNER, M. W., AND DORLING, S. Artificial neural networks (the multilayer perceptron)—a review of applications in the atmospheric sciences. *Atmospheric environment 32*, 14-15 (1998), 2627–2636.

[43] GEURTS, P., ERNST, D., AND WEHENKEL, L. Extremely randomized trees. *Machine learning 63*, 1 (2006), 3–42.

[44] GIJSBERS, P., LEDELL, E., THOMAS, J., POIRIER, S., BISCHL, B., AND VANSCHOREN, J. An open source automl benchmark. *arXiv preprint arXiv:1907.00909* (2019).

[45] GLOSTER, A. T., RHOADES, H. M., NOVY, D., KLOTSCHE, J., SENIOR, A., KUNIK, M., WILSON, N., AND STANLEY, M. A. Psychometric properties of the depression anxiety and stress scale-21 in older primary care patients. *Journal of affective disorders 110*, 3 (2008), 248–259.

[46] GUO, C.-Y., AND CHOU, Y.-C. A novel machine learning strategy for model selections-stepwise support vector machine (stepsvm). *Plos one 15*, 8 (2020), e0238384.

[47] GUYON, I., AND ELISSEEFF, A. An introduction to variable and feature selection. *Journal of machine learning research 3*, Mar (2003), 1157–1182.

[48] GUYON, I., GUNN, S., NIKRAVESH, M., AND ZADEH, L. A. *Feature extraction: foundations and applications*, vol. 207. Springer, 2008.

[49] HARRELL, F. E., ET AL. *Regression modeling strategies: with applications to linear models, logistic and ordinal regression, and survival analysis*, vol. 3. Springer, 2015.

[50] HASTIE, T., AND TIBSHIRANI, R. Generalized additive models: some applications. *Journal of the American Statistical Association 82*, 398 (1987), 371–386.

[51] HE, H., AND MA, Y. *Imbalanced learning: Foundations, algorithms, and applications*. Wiley-IEEE Press, 2013.

[52] HO, T. K. Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (1995), vol. 1, IEEE, pp. 278–282.

[53] HOSSAN, M. A., MEMON, S., AND GREGORY, M. A. A novel approach for mfcc feature extraction. In *2010 4th International Conference on Signal Processing and Communication Systems* (2010), IEEE, pp. 1–5.

[54] HU, J., NIU, H., CARRASCO, J., LENNOX, B., AND ARVIN, F. Voronoi-based multi-robot autonomous exploration in unknown environments via deep reinforcement learning. *IEEE Transactions on Vehicular Technology 69*, 12 (2020), 14413–14423.

[55] HYVARINEN, A. Fast and robust fixed-point algorithms for independent component analysis. *IEEE transactions on Neural Networks 10*, 3 (1999), 626–634.

[56] IMPEDOVO, D. Velocity-based signal features for the assessment of parkinsonian handwriting. *IEEE Signal Processing Letters 26*, 4 (2019), 632–636.

[57] INDYK, P., AND MOTWANI, R. Approximate nearest neighbors: towards removing the curse of dimensionality. In *Proceedings of the thirtieth annual ACM symposium on Theory of computing* (1998), pp. 604–613.

[58] KAISER, J. F. On a simple algorithm to calculate the'energy'of a signal. In *International conference on acoustics, speech, and signal processing* (1990), IEEE, pp. 381–384.

[59] KEERTHI, S. S., AND LIN, C.-J. Asymptotic behaviors of support vector machines with gaussian kernel. *Neural computation 15*, 7 (2003), 1667–1689.

[60] KESSY, A., LEWIN, A., AND STRIMMER, K. Optimal whitening and decorrelation. *The American Statistician 72*, 4 (2018), 309–314.

[61] KHALID, S., KHALIL, T., AND NASREEN, S. A survey of feature selection and feature extraction techniques in machine learning. In *2014 science and information conference* (2014), IEEE, pp. 372–378.

[62] KIM, H. *The ClockMe system: computer-assisted screening tool for dementia*. PhD thesis, Georgia Institute of Technology, 2013.

[63] KIRA, K., AND RENDELL, L. A. A practical approach to feature selection. In *Machine learning proceedings 1992*. Elsevier, 1992, pp. 249–256.

[64] KLINE, P. *Handbook of psychological testing*. Routledge, 2013.

[65] KUBAT, M. Neural networks: a comprehensive foundation by simon haykin, macmillan, 1994, isbn 0-02-352781-7. *The Knowledge Engineering Review 13*, 4 (1999), 409–412.

[66] KWON, O.-W., CHAN, K., HAO, J., AND LEE, T.-W. Emotion recognition by speech signals. In *Eighth European conference on speech communication and technology* (01 2003), pp. 1–4.

[67] LAHDELMA, S., AND JUUSO, E. Signal processing and feature extraction by using real order derivatives and generalised norms. part 1: Methodology. *International Journal of Condition Monitoring 1*, 2 (2011), 46–53.

[68] LAUTERBUR, P. C. Image formation by induced local interactions: examples employing nuclear magnetic resonance. *nature 242*, 5394 (1973), 190–191.

[69] LEDELL, E., AND POIRIER, S. H2O AutoML: Scalable automatic machine learning. *7th ICML Workshop on Automated Machine Learning (AutoML)* (July 2020).

[70] LEUNG, F. H.-F., LAM, H.-K., LING, S.-H., AND TAM, P. K.-S. Tuning of the structure and parameters of a neural network using an improved genetic algorithm. *IEEE Transactions on Neural networks 14*, 1 (2003), 79–88.

[71] LI, C., HU, M., LI, Y., JIANG, H., GE, N., MONTGOMERY, E., ZHANG, J., SONG, W., DÁVILA, N., GRAVES, C. E., ET AL. Analogue signal and image processing with large memristor crossbars. *Nature electronics 1*, 1 (2018), 52–59.

[72] LI, S. Z. *Encyclopedia of Biometrics: I-Z.*, vol. 2. Springer Science & Business Media, 2009.

[73] LIKFORMAN-SULEM, L., ESPOSITO, A., FAUNDEZ-ZANUY, M., CLÉMENÇON, S., AND CORDASCO, G. Emothaw: A novel database for emotional state recognition from handwriting and drawing. *IEEE Transactions on Human-Machine Systems 47*, 2 (2017), 273–284.

[74] LINDLEY, D. V. Fiducial distributions and bayes' theorem. *Journal of the Royal Statistical Society. Series B (Methodological)* (1958), 102–107.

[75] LIPPERT, R. A., RIFKIN, R. M., AND LUGOSI, G. Infinite-σ limits for tikhonov regularization. *Journal of Machine Learning Research 7*, 5 (2006).

[76] LOVIBOND, P. F., AND LOVIBOND, S. H. The structure of negative emotional states: Comparison of the depression anxiety stress scales (dass) with the beck depression and anxiety inventories. *Behaviour research and therapy 33*, 3 (1995), 335–343.

[77] LUNDBERG, U., KADEFORS, R., MELIN, B., PALMERUD, G., HASSMÉN, P., ENGSTRÖM, M., AND DOHNS, I. E. Psychophysiological stress and emg activity of the trapezius muscle. *International journal of behavioral medicine 1*, 4 (1994), 354–370.

[78] MARTIN, R. Noise power spectral density estimation based on optimal smoothing and minimum statistics. *IEEE Transactions on speech and audio processing 9*, 5 (2001), 504–512.

[79] MCCULLAGH, P., AND NELDER, J. A. *Generalized linear models*. Routledge, 2019.

[80] MCKEOWN, M. J., HANSEN, L. K., AND SEJNOWSK, T. J. Independent component analysis of functional mri: what is signal and what is noise? *Current opinion in neurobiology 13*, 5 (2003), 620–629.

[81] MOHAMMED, R., RAWASHDEH, J., AND ABDULLAH, M. Machine learning with oversampling and undersampling techniques: overview study and experimental results. In *2020 11th International Conference on Information and Communication Systems (ICICS)* (2020), IEEE, pp. 243–248.

[82] MOLINARO, A. M., SIMON, R., AND PFEIFFER, R. M. Prediction error estimation: a comparison of resampling methods. *Bioinformatics 21*, 15 (2005), 3301–3307.

[83] MOUSAVI NEZHAD, M., GIRONACCI, E., REZANIA, M., AND KHALILI, N. Stochastic modelling of crack propagation in materials with random properties using isometric mapping for dimensionality reduction of nonlinear data sets. *International Journal for Numerical Methods in Engineering 113*, 4 (2018), 656–680.

[84] NELDER, J. A., AND WEDDERBURN, R. W. Generalized linear models. *Journal of the Royal Statistical Society: Series A (General) 135*, 3 (1972), 370–384.

[85] NOLAZCO-FLORES, J. A., FAUNDEZ-ZANUY, M., DE LA CUEVA, V. M., AND MEKYSKA, J. Exploiting spectral and cepstral handwriting features on diagnosing parkinson's disease. *IEEE Access* (2021), 1–1.

[86] NOLAZCO-FLORES, J. A., FAUNDEZ-ZANUY, M., VELÁZQUEZ-FLORES, O. A., CORDASCO, G., AND ESPOSITO, A. Emotional state recognition performance improvement on a handwriting and drawing task. *IEEE Access 9* (2021), 28496–28504.

[87] NUSSBAUMER, H. J. The fast fourier transform. In *Fast Fourier Transform and Convolution Algorithms*. Springer, 1981, pp. 80–111.

[88] OLSON, R. S., AND MOORE, J. H. Tpot: A tree-based pipeline optimization tool for automating machine learning. In *Workshop on automatic machine learning* (2016), PMLR, pp. 66–74.

[89] OPPENHEIM, A., AND SCHAFER, R. From frequency to quefrency: a history of the cepstrum. *IEEE Signal Processing Magazine 21*, 5 (2004), 95–106.

[90] OPPENHEIM, A. V., AND JOHNSON, D. H. Discrete representation of signals. *Proceedings of the IEEE 60*, 6 (1972), 681–691.

[91] PARTRIDGE, M., AND CALVO, R. Fast dimensionality reduction and simple pca. *Intelligent data analysis 2*, 3 (1997), 292–298.

[92] PEDREGOSA, F., VAROQUAUX, G., GRAMFORT, A., MICHEL, V., THIRION, B., GRISEL, O., BLONDEL, M., PRETTENHOFER, P., WEISS, R., DUBOURG, V., VANDERPLAS, J., PASSOS, A., COURNAPEAU, D., BRUCHER, M., PERROT, M., AND DUCHESNAY, E. Scikit-learn: Machine learning in Python. *Journal of Machine Learning Research 12* (2011), 2825–2830.

[93] RABINER, L. R., AND GOLD, B. Theory and application of digital signal processing. *Englewood Cliffs: Prentice-Hall* (1975).

[94] RISH, I., ET AL. An empirical study of the naive bayes classifier. In *IJCAI 2001 workshop on empirical methods in artificial intelligence* (2001), vol. 3, pp. 41–46.

[95] ROWEIS, S. T., AND SAUL, L. K. Nonlinear dimensionality reduction by locally linear embedding. *science 290*, 5500 (2000), 2323–2326.

[96] SAMMUT, C., AND WEBB, G. I. *Encyclopedia of machine learning*. Springer Science & Business Media, 2011.

[97] SCHLENKER, B. R., AND LEARY, M. R. Social anxiety and self-presentation: A conceptualization model. *Psychological bulletin 92*, 3 (1982), 641.

[98] SCHULLER, B., RIGOLL, G., AND LANG, M. Hidden markov model-based speech emotion recognition. In *2003 IEEE International Conference on Acoustics, Speech, and Signal Processing, 2003. Proceedings.(ICASSP'03).* (2003), vol. 2, Ieee, pp. II–1.

[99] SEERA, M., AND LIM, C. P. A hybrid intelligent system for medical data classification. *Expert systems with applications 41*, 5 (2014), 2239–2249.

[100] SELYE, H. *The stress of life*. McGraw-Hill, 1956.

[101] SHAHROKH ESFAHANI, M., AND DOUGHERTY, E. R. Effect of separate sampling on classification accuracy. *Bioinformatics 30*, 2 (2014), 242–250.

[102] SHAW, B., AND JEBARA, T. Structure preserving embedding. In *Proceedings of the 26th Annual International Conference on Machine Learning* (2009), pp. 937–944.

[103] SHORTEN, C., AND KHOSHGOFTAAR, T. M. A survey on image data augmentation for deep learning. *Journal of Big Data 6*, 1 (2019), 1–48.

[104] SPIELBERGER, C. D. State-trait anxiety inventory. *The Corsini encyclopedia of psychology* (2010), 1–1.

[105] STEIN, M. L. *Interpolation of spatial data: some theory for kriging*. Springer Science & Business Media, 2012.

[106] STONE, M. Cross-validatory choice and assessment of statistical predictions. *Journal of the royal statistical society: Series B (Methodological) 36*, 2 (1974), 111–133.

[107] THERNEAU, T. M., AND GRAMBSCH, P. M. The cox model. In *Modeling survival data: extending the Cox model*. Springer, 2000, pp. 39–77.

[108] THORNTON, C., HUTTER, F., HOOS, H. H., AND LEYTON-BROWN, K. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining* (2013), pp. 847–855.

[109] TIBSHIRANI, R. Regression shrinkage and selection via the lasso. *Journal of the Royal Statistical Society: Series B (Methodological) 58*, 1 (1996), 267–288.

[110] TRIGUERO, I., GALAR, M., VLUYMANS, S., CORNELIS, C., BUSTINCE, H., HERRERA, F., AND SAEYS, Y. Evolutionary undersampling for imbalanced big data classification. In *2015 IEEE Congress on Evolutionary Computation (CEC)* (2015), IEEE, pp. 715–722.

[111] TURNER, A. I., SMYTH, N., HALL, S. J., TORRES, S. J., HUSSEIN, M., JAYASINGHE, S. U., BALL, K., AND CLOW, A. J. Psychological stress reactivity and future health and disease outcomes: A systematic review of prospective evidence. *Psychoneuroendocrinology 114* (2020), 104599.

[112] TURNER, C. R., FUGGETTA, A., LAVAZZA, L., AND WOLF, A. L. A conceptual basis for feature engineering. *Journal of Systems and Software 49*, 1 (1999), 3–15.

[113] VAN DER LAAN, M. J., POLLEY, E. C., AND HUBBARD, A. E. Super learner. *Statistical applications in genetics and molecular biology 6*, 1 (2007).

[114] VAN DER MAATEN, L., POSTMA, E., VAN DEN HERIK, J., ET AL. Dimensionality reduction: a comparative. *J Mach Learn Res 10*, 66-71 (2009), 13.

[115] VAPNIK, V. *The nature of statistical learning theory*. Springer science & business media, 2013.

[116] VEHTARI, A., GELMAN, A., AND GABRY, J. Practical bayesian model evaluation using leave-one-out cross-validation and waic. *Statistics and computing 27*, 5 (2017), 1413–1432.

[117] WEIK, M. *Communications standard dictionary*. Springer Science & Business Media, 2012.

[118] WIDROW, B., AND STEARNS, S. *Adaptive Signal Processing*. Prentice-Hall, Englewood Cliffs, NJ, 1985.

[119] WOLPERT, D. H. Stacked generalization. *Neural networks 5*, 2 (1992), 241–259.

[120] YADAV, S., AND SHUKLA, S. Analysis of k-fold cross-validation over hold-out validation on colossal datasets for quality classification. In *2016 IEEE 6th International conference on advanced computing (IACC)* (2016), IEEE, pp. 78–83.

[121] YAP, B. W., ABD RANI, K., ABD RAHMAN, H. A., FONG, S., KHAIRUDIN, Z., AND ABDULLAH, N. N. An application of oversampling, undersampling, bagging and boosting in handling imbalanced datasets. In *Proceedings of the first international conference on advanced data and information engineering (DaEng-2013)* (2014), Springer, pp. 13–22.

[122] YU, L., AND LIU, H. Feature selection for high-dimensional data: A fast correlation-based filter solution. In *Proceedings of the 20th international conference on machine learning (ICML-03)* (2003), pp. 856–863.

[123] ZHANG, S., ZHANG, C., AND YANG, Q. Data preparation for data mining. *Applied artificial intelligence 17*, 5-6 (2003), 375–381.