eman ta zabal zazu

Universidad del País Vasco
Euskal Herriko Unibertsitatea

INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

Bachelor's Thesis

Grado en Ingeniería Informática

Computación

One-Class models for the prognosis of COVID-19 infection outcome

Unai Carbajo Escajadillo

Advisors
Borja Calvo Molinos
Iñaki Inza Cano
Rubén Armañanzas Arnedillo (BCAM)

June 20th of 2021

Abstract

This project aims to address the prognosis prediction problem for COVID-19 patients making use of One-Class Classification techniques. Data retrieved from Spanish hospitals has been used for the development of models in the attempt to predict whether a prior COVID-19 positive inpatient will decease or not. This data collection includes clinical information (age, sex, first hearth rate check, etc.), diagnosis and procedural information, and laboratory findings (complete blood count variables, D-Dimer count, etc.) of 1,798 patients.

This project presents a machine learning workflow composed by a data filtering process, followed by a model hyperparameter optimization step, and eventually, the training, testing and evaluation steps of the final models. The workflow implements 3 relevant One-Class Classifiers: One-Class Support Vector Machine, Local Outlier Factor and Autoencoder. These models follow the One-Class Classification paradigm, which is a branch of unsupervised machine learning and it is based on making classifications with models entirely trained with data belonging to a single class.

The 3 experiments showed an overall ROC-AUC of 0.558 ± 0.101 and sensitivity of 0.567 ± 0.123. The analysis made after the classifications turned out to highlight the weak representation of deceased samples and strong similarity between deceased and discharged patients, a key issue in COVID-19 prognosis prediction problems.

Keywords: One-Class Classification, COVID-19, prognosis, unsupervised learning

Contents

List of Figures

List of Tables

List of algorithms

CHAPTER 1

Introduction

Coronavirus disease (COVID-19) has been hitting the whole world since the start of the first outbreak in Wuhan, China, in late 2019. This disease is caused by a new type of coronavirus called SARS-CoV-2. The World Health Organization [2] registered by May 11th 2021 a total of 158,551,526 confirmed cases of COVID-19 and 3,296,855 deaths (reported by national authorities). But this disease did not discriminate at the time of hitting countries over all continents. An example of this is the situation in Spain, which has been one of the most damaged countries in Europe, leaving a total of 3,559,222 detected cases and 78,726 deaths for the record of May 11th 2020.

A large set of countries, in the try of slowing down the virus' spread, have joined their forces, resulting in changes in many fields. This could be seen in the facemask manufacturing process improvements, or even in the novel vaccines' development.

However, the effect made by this disease is tangible over many other fields. One example is the improvement on science research made covering COVID-19 related issues. This extends from a medical field as epidemiology to AI related fields, even combinations of them. The impact of the COVID-19-related research is so big that by the end of 2020 it became the topic with more papers in history, registering more than 200,000 papers. However, this overwhelming science growth was not worthless. All these improvements helped the resource management of hospitals, lightening decision making processes of health security entities, and enhancing the security countermeasures for COVID-19, among many others.

The people who suffer from COVID-19 share many of the most common symptoms, such as fever, dry cough or fatigue. However, there are many more indicators, which are less common in COVID-19 positive patients: headache, loss of taste or smell, etc. The most severe patients show signs such as shortness of breath, loss of appetite, confusion, persistent pain or pressure in the chest and high temperature (above 38 °C). All the symptoms are collected in The World Health Organization's COVID-19 section [2]. These symptoms, along with other biochemical, laboratory-related and medication-related notes, were widely used in the decision making process in ER of hospitals. These features were crucial at the time of deciding whether a person would need urgent care or not.

Another application of the symptoms was the prediction of early diagnosis in persons with unknown COVID-19 status. There is another important field that was tackled, which

is the propagation analysis and prediction of COVID-19. This problem was about obtaining early epidemiological results based on demographic data, including the reproduction number of the virus, daily infection number in an area, etc. And, the one which has been recently in an all-time-high situation is the vaccine/drug discovery, crucial at the time of protecting the population against virus' effects.

Even though science developments in COVID-19 related topics grow as days go on, the availability of data still plays an important role in the scientific community. This implies that most of the work was heavily biased by the dataset they were developed with. Every database differed notably in many features, like the place the dataset was obtained from, or the date it was obtained. These facts were clear in machine learning approaches for prognosis/diagnosis prediction, where the reusability of models in hospitals between countries, or even hospitals of the same region, was an impossible task. This was not a viable problem, since the data used to tune the models belonged to different regions and timestamps. Therefore, the best scenario for this type of works should be the definition of a unique standardization of COVID-19 data, and thus joining the efforts of researching over a retrievable and centralized data. However, this could imply the loss of the localization aspect, which is the backbone of problems related to epidemiological analysis.

A large set of machine learning approaches for COVID-19 patient prognosis prediction rely on the use of similar features that are proved to have a heavy impact at the time of establishing the evolution of the disease in an infected person [3], [4], [5]. Clinical features like the age, O^2 saturation at Emergency Room (ER) or the origin (place before hospital) of the patient or other laboratory findings as the lactate dehydrogenase (LDH), D-dimer, etc. represent the set of features that are usually related to severe COVID-19 patients. Many of the previously mentioned problems used this knowledge in order to develop their models in an optimized way. The experimentation of this project also applies these ideas.

This project has the following structure: Chapter 2 analyzes the main objective of the project; Chapter 3 reviews advanced techniques applied to COVID-19 prognosis prediction and the novelties of One-Class Classification (OCC); Chapter 4 describes the dataset used for the experimentation; Chapter 5 explains the main concepts of OCC, covering deeply the techniques applied in the experimentation, and the evaluation made over the 3 main techniques is analyzed; Chapter 6 contains the interpretation of the results obtained in the main experimentation, defining a post-classification analysis of data; Chapter 7 describes the conclusions obtained over the analysis of the results.

CHAPTER 2

Objective

This project proposes an experimentation procedure for the prediction of the outcome of hospitalized COVID-19 patients using a One-Class Classification (OCC) approach [6]. The OCC paradigm is an alternative proposal to the classical multi-class classifiers, which is based on modeling a classifier around a single class, instead of estimating a prediction based on the data of multiple classes.

The experimentation aims to show the performance of 3 well-known One-Class classifiers, following an unsupervised approach and identifying the weaknesses and strengths of these kinds of techniques. This project tackles the main concepts of OCC in chapter 5, analyzing step by step the ideas of OCC, specially applied to health-related problems, showcasing the ideas which could make this approach a reliable option in contrast to the "by default" supervised multi-class classification.

The prognosis prediction in COVID-19 patients revolves around different objectives. This might include the prediction of mortality of a COVID-19 patient, which is usually referred to as risk of mortality. This type of task can be addressed in 2 main ways. The first one related to predict whether a COVID-19 patient will pass away or not [7, 4]; the second one is related to the classification of patients into risk fields [1], [8], which represent the severeness of the disease on infected patients. Even though the later method is not carried out in this project, it is worth noting that the use of several scores for the evaluation of the severeness of COVID-19 is quite popular, e.g. 4C Mortality Score by the ISARIC Coronavirus Clinical Characterisation Consortium [8]. Another analysis developed in this project is the prediction of hospitalization length of a COVID-19 patient [9], a method that depends heavily on the location where the analysis was developed on, and how the management of patients is made.

This project works with the first idea, estimating whether a patient will decease or, on the other hand, will be discharged/translated. The data used for the experimentation and analysis of this project is the database released from HM Hospitales on April 25th 2020, thanks to the project COVID DATA SAVES LIVES (CDSL) [10]. This database is fully anonymized and it contains clinical, biochemical and epidemiological features, widely explained in the chapter 4.

Another issue to point in this project is the post-classification analysis, made after the

predictions obtained by One-Class classifiers. The experimentation covers the comparison between OCC and classical classification approaches, analyzing the real scope of OCC through the representation of the data, and summarizing with final conclusions about the use of OCC in COVID-19 prognosis prediction.

CHAPTER 3

State-Of-The-Art Review

Two of the more relevant works on COVID-19 prognosis are the contribution made by Yan *et al.* (2020) [11] and Knight *et al.* (2020) [8]. Both models differ heavily between them.

The work by Yan *et al.* is able to predict the outcome of the patients more than 10 days in advance with an "interpretable" approach. The final model is based on the use of three critical biomarkers highly related to the severeness disease: LDH (lactic dehydrogenase), lymphocyte count and hs-CRP (high-sensitivity C-reactive protein). In order to carry out the study, 485 blood samples were used (375 for training and 110 for test), all of them collected from patients of the Tongji Hospital (Wuhan). Empirically, the idea behind this model is to identify high-risk patients.

The work by Knight *et al.* summarizes the development of an easy-to-use risk stratification score with common clinical and biochemical parameters obtained at hospital stays from patients of 260 hospitals across England, Scotland, and Wales. This project is one the most relevant and exhaustive works of mortality risk prediction, making use of data from 34,692 patients, resulting in a total of 8 features each. The features contained clinical and biochemical values: age, sex, number of comorbidities, respiratory rate, peripheral oxygen saturation, level of consciousness, urea, and C-reactive protein.

In spite of the fact that COVID-19 studies have been covered from different machine learning and deep learning perspectives, the OCC paradigm has not been applied yet to prognosis prediction. There are examples of OCC applied to the extraction of deep features in normal chest CT scans of patients with pneumonia infection (derived from COVID-19) [12], seen in the work by Khan *et al.* (2021). However, there has not been any One-Class implementation to treat the prediction of the outcome of COVID-19 patients. Therefore, this project presents a novel approach to address this problem.

Most of the One-Class Classification techniques are based on classical machine learning methods. An example of this can be seen in the SVM application for OCC, known as One-Class Support Vector Machine (OCSVM or 1SVM) [13], which transforms the original SVM algorithm to model a single class. Even though these kinds of methods work well in many applications, they often fail when working with high dimensional data due to bad scalability, heavy computational weight and curse of dimensionality (further explanation in Section 5.1.3.1). So, this is where new methods have their chance to be developed. Thus, along

with the rise of the deep learning [14, 15] (LeCun *et al.*, 2015; Schmidhuber, 2015) on the last decade, OCC techniques have been developed as well, adapting to more sophisticated methods based on deep learning and therefore solving many of the problems that classical machine learning approaches have. The most relevant impact of these new techniques was seen in feature filtering processes, where shallow machine learning methods require substantial feature filtering, whilst deep learning techniques address this issue internally.

The improvements on Deep One-Class Classification are clear in anomaly detection tasks. An example of this technique is the novel technique of Deep Support Vector Data Description [16] (Deep SVDD) by Ruff *et al.* (2018). A method based on training a neural network while optimizing an hypersphere enclosing samples representatives of the target class. This method improves the classical SVDD [17] by Tax and Duin (2004), by means of the application of deep learning techniques. These improvements are shown on how the minimization of the hypersphere forces the neural network to capture the most salient features since the network has to map the target data points inside the hypersphere as tight as possible. In this way, the classical feature filtering methods, that are applied to the majority of machine learning problems, are left aside.

Even though not all unsupervised approaches are considered as OCC, there are deep learning methods that have demonstrated to work properly in One-Class scenarios. An example of this is the Deep Autoencoder (Hinton and Salakhutdinov, 2006), a type of artificial neural network that tries to extract salient features from a dataset by the use of an intermediate reduced dimension. This approach is carried out following the classical methodology of deep learning, designing a multi-layer neural network. This work makes use of a shallow autoencoder, defining no more than a single hidden layer for the latent space representation.

CHAPTER 4

Data

The data used for this project is a processed version of the database released on the project COVID DATA SAVES LIVES (CDSL) [10], by HM Hospitales. This database contains anonymized records from 2,310 patients, all of them collected from the HM Hospitales Electronic Health Record (EHR) system. The patients registered were diagnosed previously as "COVID positive" or "COVID pending", from the beginning of the epidemic until April 25th 2020.

The collected variables in this database include clinical information of the patients (2,226 records), with features like age, sex, data of ICU stays, first and last constant records at emergency, and many more. The database also includes information about the administered drugs to the patients (more than 60,000 records) during admission. Another information registered is the record of vital signs taken during the admission, gathering up to 54,000 records. One of the most relevant data registered is the laboratory findings of each patient, including requests made during admission and in the Emergency Room (ER); reaching a total of 398,884 records. And finally, data about the diagnosis and procedural information, encoded in ICD-10 standard[1]. This information is taken during 2 main episodes, the first one during hospital admission (more than 1,600 records), and the second one during an emergency, if it happened. Finally, registering more than 1,900 profiles.

The database was characterized as follows:

- Male/Female ratio: 58.78%
- Confirmed COVID/Suspected covid patient percentage: 90.97%
- Deceased/Discharged (other) percentage: 14.96%
- Prior ER room stay patient percentage: 96.33%

However, the data used in this project was, as mentioned earlier, a processed version of this database. This version of the data followed a preprocessing work in the project [1]

[1]The ICD-10 standard is the 10th revision of the International Statistical Classification of Diseases and Related Health Problems [18], a global standard for health data, clinical documentation, and statistical aggregation. It is used for medical classification.

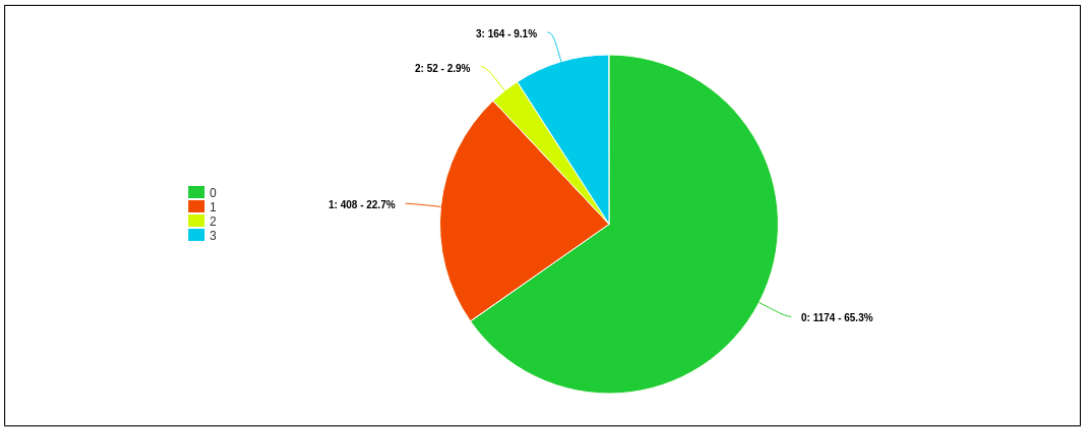


Figure 4.1: Comorbidity number visualization

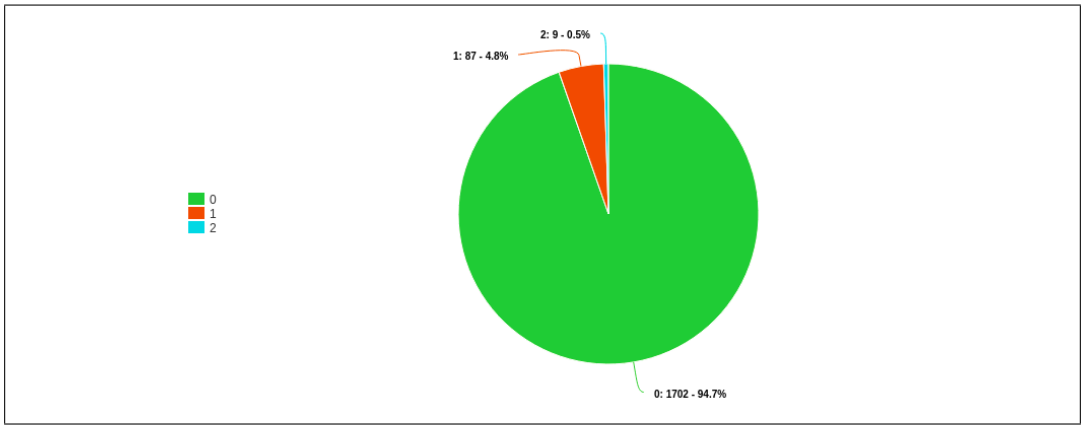


Figure 4.2: Symptom number visualization

developed by Armañanzas *et al.* The pipeline to preprocess the data includes techniques as imputation (filtering missing values) and feature definition. It is summarized in 1,798 selected patients with 44 features each (plus the target variable), composed of 276 (15.35%) deceased patients and 1,522 (84.65%) discharged patients. The percentage of male was around 60.68%. More information about the features is registered in Table 1 of the Appendix A, which shows a brief description of each feature. The mean and standard deviation for every feature is shown in Table 2.

Features like the comorbidities are summarized in a single variable, which is the number of comorbidities the patient has. The same filtering is applied for the symptoms of the patient. These two new variables are defined in a [0,3] range, defining the number of comorbidities/symptoms the patient has, respectively. The value 3 is assigned for those patients that have 3 or more comorbidities/symptoms. The conditions to check are Chronic Cardiac Disease, Chronic Respiratory Disease (incl asthma), Chronic Renal Disease, Mild to Sever Liver Disease, Dementia, Chronic Neurological Conditions, Connective Tissue Disease, HIV or AIDS, Cancer, Obesity. And the symptoms to check are Dyspnea, Fatigue, Lost of Consciousness, Myalgia, Sputum, Anosmia, Fever, Diarrhea, Vomiting and Cough. The distribution of both features is shown in Figures 4.1 and 4.2.

CHAPTER 5

Methods

5.1 One-Class Classification

One-Class Classification (OCC) [19] is a machine learning paradigm whose objective resides in identifying samples from a specific class. In contrast to more classical approaches (binary-class classification, multi-class), where a classifier is trained with samples from all classes that are involved in the problem. In order to predict the true class where a new unseen sample belongs, One-Class classifiers are trained exclusively with samples from one class, finally predicting whether a sample belongs to that class or not. This class is usually referred to as normal or positive class, and, on the other hand, the samples that do not belong to the normal class are considered as anomaly/outlier/novelty/negative samples. Despite the OCC theory does not consider a second class, it is common to label them as anomaly/outlier class, or basically, negative class.

It is usually assumed that the positive class is properly characterized, while the other class is poorly characterized and usually a low number of samples of it is available. The focus of the learning process is uniquely centered in the modelization of the "normal class", i.e. a pattern for the rest of the classes is not learned. This is a common situation in machine failure detection problems, where normal operations of a machine are easily retrievable. However, as the machine is meant to work properly most of the time, the samples of anomaly operations are quite low, so the whole classification depends on the characterization of normal operations and on the performance of the one-class classifier's performance in order to identify normal operations.

Nevertheless, the OCC approaches may differ on the representation of the data. It is possible that both negative and positive samples are properly or fairly well defined. This would be a straightforward approach for multi-class classification techniques. However, this problem could also be solved with OCC techniques, leaving our desired-to-predict class as the positive class, and grouping other classes as the negative class. This kind of approach is carried out in this project.

At first glance, OCC techniques were referred to as *Single-Class Classification* [20] by Minter, who designed a Bayes classifier developing the OCC taxonomy as we know it today, and so, defining those Single-Class classifiers as *classifiers which will classify*

data into the "class of interest"or the "other" classes but will require only labeled training samples from the "class of interest" to design it. This definition perfectly characterizes modern One-Class Classifiers. Many years later, One-Class Classification [19] term was firstly coined by Moya and Hush, in 1996. Over time, the definition of OCC has changed, adapting it to several problems: outlier-detection, novelty-detection, enhancement of multi-class classification results, single-class modeling, concept learning, etc. OCC has shown a particular great prominence in topics related to computer vision, such as abnormal image detection, abnormal event detection and biometric applications, as is covered in the OCC survey [21] by Perera *et al.*

Therefore, OCC techniques had been getting more relevant as they were frequently used in the topics named earlier. Thus, they had suffered many changes, and, in particular, crucial improvements that helped unifying and regularizing standards that made the development and understanding of new OCC methods easier.

One-Class classifiers are considered as an unsupervised learning paradigm, since all the samples used for training are supposed to be integrally composed of samples from one specific class. Therefore, during the training process there is no use of samples apart from those of the desired class, so the classifier does not draw upon on any information of the classes the label of the samples could provide. However, this does not mean that One-Class classifiers are limited for problems whose datasets are not untagged, in fact, as every unsupervised classifier, it has a high reliability in labeled datasets, and could show better results than supervised classifiers, especially in outlier detection problems, where unsupervised classification proves to work better in problems that rely on pattern discovery.

This brings up the discussion about selecting an OCC approach rather than a multi-class approach. Tax *et al.*, 2002, registered in their paper [22] relevant principles of the one-class/two-class classification discussion, where they reach the origin of OCC, and how the core of these problems totally depended on the distribution of the data. Having this in mind, One-Class classifiers only rely on how the positive class is represented. The distribution of the negative class has no relevance, as it is not used at all during the training process. This is directly related with classifiers based on decision boundaries: conventional two-class classifiers set a decision supported from both boundary sides by samples of both classes, whereas, with one-class classifiers the boundary is solely supported by samples of positive data. This fact makes it harder for the classifier, which is based in just one class, to decide how tight the boundary should fit around the data.

5.1.1 Concepts

As it is common in all classification tasks, in OCC even the smallest concepts are in the need of wider and deepest explanation. These concepts could include classical recurrent topics that appear in every Machine Learning task, as the understanding and representation of the data. And other, more specific concepts that include theoretical topics, such as the model's interpretability, which is highly relevant when working with health-related problems. In addition, the representation of the outcome of classifiers has an important role, which represents one of the more important steps when working with One-Class classifiers. These topics (and more) will be discussed along this section.

Let $\mathbb{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in \mathbb{R}^d$ be the set of d-dimensional instances and $\mathbb{Y} =$

$\{y_1, ..., y_n\}$ the corresponding label space (y_i is the corresponding label to $\boldsymbol{x_i}$). Therefore, a dataset D is defined by $(\boldsymbol{x_i}, y_i)$ pairs in the following way: $D = \{(\boldsymbol{x_1}, y_1), ..., (\boldsymbol{x_n}, y_n)\} \in \mathbb{X} \cdot \mathbb{Y}$. Then the $\mathbb{F}$ space of functions to be learned (i.e., a model) is defined, which maps the input $\boldsymbol{x_i}$ to some output label y_i. This mapping is learned by fitting the model to a large data space $D_{train} \in D$, and this highly depends on the chosen classification method. Once the classifier is trained, it is able to label an unseen $\boldsymbol{x'} \in D \backslash D_{train}$ thanks to the function f of the hypothesis space in the following way: $\hat{y} = f(\boldsymbol{x'})$. This is the common definition for supervised classification, which fully depends on how well all the unique values (target classes) of $\mathbb{Y}$ are represented in the feature space $\mathbb{X}$. As mentioned previously, when talking about fraud, abnormal behavior detection, normal operations of a machine are easily retrievable. However, those abnormal operations are really hard to identify, and even more difficult to collect. Thus, One-Class classifiers become a reliable option.

One-Class Classifiers are usually referred to as unsupervised learning. However, their learning process can be approached from a supervised outlook. The approach carried out in this project represents the classification task as an unsupervised learning task, from the OCC view:

(a) **Without any knowledge from labels**. This situation is quite common in scenarios of fraud detection or undesired machine behavior. The notation is defined in the following way: Let $\mathbb{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in \mathbb{R}^d$ be the set of samples, and the labels corresponding to those objects are unknown. Therefore, a dataset D is entirely made up of objects from $D = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$. Then $D_{train} \in D$ is defined, where D_{train} only contains samples from the desired class c^p (positive/normal) class, and occasionally, it might include some not-desired abnormal objects[1]. Then the $\mathbb{F}$ Hypothesis space of functions to be learned (i.e., a model) is defined, which maps the input $\boldsymbol{x'}$ in order to predict an output label $\hat{y} = f(\boldsymbol{x'})$, which may be $inlier$ or $outlier$. This last step is one of the most critical steps of OCC, because it introduces the novel term of "outlierness". The "outlierness" of a sample is defined as the degree of a sample of being outlier in comparison to a defined set of samples.

(b) **With previous knowledge from labels**. This approach has been carried out during this project. The notation is defined in the following way: Let $\mathbb{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in \mathbb{R}^d$ be the set of samples and $\mathbb{Y} = \{y_1, ..., y_n\}$ the corresponding set of labels (y_i is the corresponding label to $\boldsymbol{x_i}$). Taking into account that $y_i \in \mathbb{C} = \{C_1, ..., C_j\}$ where $i = 1..n$. Even though the labeling is known, it does not affect the algorithm, as the only relevant class is the positive one, known as c^p. The dataset D does not include those labels, but it is known whether an object belongs to the positive class or not, in order to fit the model in the training step. Therefore, a dataset D is entirely made up of objects of $D = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}$. Then, $D_{train} \in D$ is defined, where D_{train} only contains samples from the desired class c^p (positive/normal)[2]. Then, the $\mathbb{F}$ Hypothesis space of functions to be learned (i.e., a model) is defined, which maps the input $\boldsymbol{x'}$ in order to predict an output label $\hat{y} = f(\boldsymbol{x'})$, which may be $inlier$ or $outlier$.

[1] It is common to fit the classifier with positive samples and a little amount of poorly distributed negative samples in this particular situation due to the unawareness of true distribution of the positive class in the original data.

[2] It is common to fit the classifier with only positive samples in this particular situation.

As mentioned in these 2 situations, the data to train the model depends on the available knowledge about the data. In the review of OCC [23] made by Khan *et al.*, 2014, they proposed a taxonomy for OCC, which is divided in 3 categories: Availability of Training Data, Algorithm/Methodology and Application Domain. They tried to cover the state-of-the-art by the time the survey was released, where they summarized the key contributions made since the first works in OCC. For this project, the most important category is the first category "Availability of Training Data". This is the one taken into account in this project. In this certain category they list 3 options:

(a) Learning with positive examples only.

(b) Learning with positive examples and some amount of poorly sampled negative examples or artificially generated outliers.

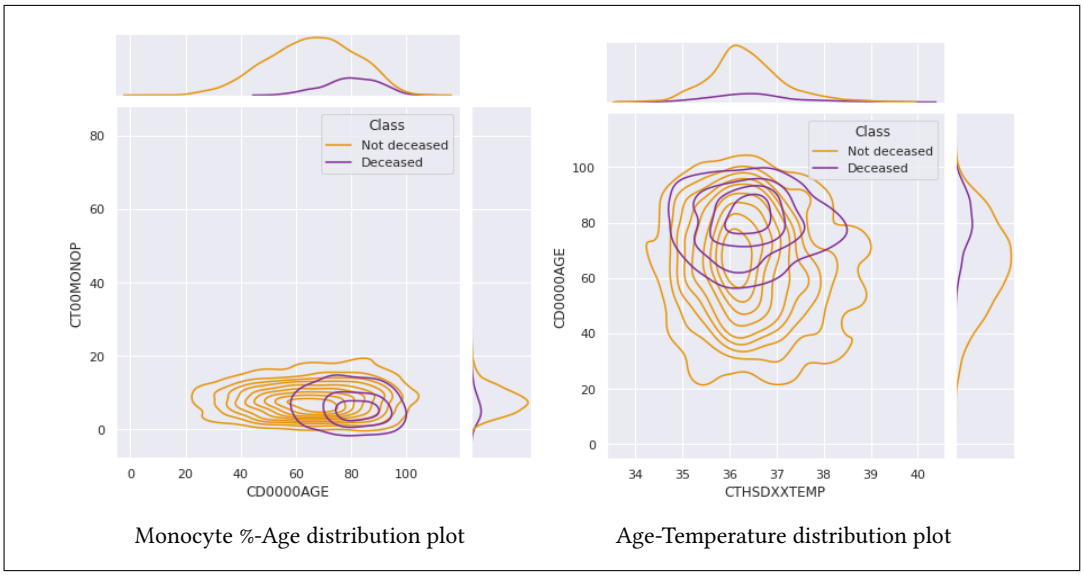
(c) Learning with positive and unlabeled data.

The authors mentioned methods as Support Vector Machine [13] (Scholkopf *et al.*, 1999), where they tackle the OCC problems by means of SVM methodology, using positive examples only. This will be relevant during the development of the project. The main idea is to learn a decision boundary around the positive data in order to discriminate unseen data as outlier. This last issue will be explained widely in Section 5.1.4.

The idea of using only positive samples for the training step is carried out in this project, following the OCC methodology. As it is pointed out in Section 4, the dataset originally contained a reduced set of samples that were not labeled. These unlabeled samples were, in fact, patients whose hospital stay was long enough to not get discharged or be deceased by the end of the *CDSL* research. So they were labeled as "discharged", and as "negatives" by the time of defining the OCC problem.

This brings up the next issue: **selection of the positive class**. Selecting which class should be referred to as positive does not imply a critical issue in the majority of the problems, since the application itself presents particular characteristics that make the selection of the class a straightforward decision (e.g. correct machine behavior examples as positive in abnormal machine behavior). However, there are particular cases , as seen in the dataset obtained from the *CDSL* project, where two classes are defined (discarding unlabeled samples): "discharged" or "deceased". This could represent a situation where the main objective resides on identifying a single class among others, and so, model the classifier around that specific class. This kind of situations requires the establishment of some follow-up steps in order to make a proper selection of the positive class:

- The class should fit the problem's requirements, so the outcome produced by the selection of that class should be useful for solving the problem.
- There should be enough samples in order set-up a proper decision boundary, avoiding the risk of underfitting due to the lack of samples.
- The samples of the desired class should be informative enough. As explained in the previous point, the classifier should be able to define a proper decision boundary thanks to informative features. By the time of evaluating whether a sample is positive or not, the classifier should be able to discriminate that sample based on the retrieved knowledge during the train process.



Monocyte %-Age distribution plot

Age-Temperature distribution plot

Figure 5.1: Feature pair-plot representation for D0000AGE, CTHSDXXTEMP and CT00MONOP. The representation of the data in the class "deceased" is tighter than in the "not deceased" samples.

- The samples should follow a well defined and homogeneous distribution.

This project's objective is about predicting the prognosis outcome of a COVID-19 positively diagnosed patient by means of OCC models. This broughts up two options: selecting discharged patients as the data belonging to the positive class, or selecting deceased patients as the positive class. Firstly, this could be a problem, since both classes were perfectly valid for the positive class position. Both options could be handled properly in order to carry out the desired results. Following with the conditions mentioned, the used data, discussed at Section 4, projected that the number of samples of the "discharged" class was, in fact, considerably higher than the number of samples of the "deceased" class: 16.89% (deceased) to 83.11% (discharged + unlabeled). This second condition was favorable for the "discharged" class, which made the decision even tougher. The third condition (followed by the fourth condition) was the one that made the decision clear: high number of variables turn out to be distributed in a quite similar way for "deceased" and "discharged" classes. However, there were few variables that turned out to be more consistent (less variance) for the class "deceased", in contrast to samples belonging to "discharged". Those variables were, in fact, the more informative features: the age of the patient (CD0000AGE), the first temperature record taken at Emergency (CTHSDXXTEMP), the percentage (x10e3/μL blood) of monocytes (CT00MONOP), etc. These three variables are represented in the Figure 5.1. However, more features were identified as relevant regarding to the deceased patients.

Once the positive class is selected, it is time to get familiar with other concepts around OCC. The idea of *outlier-score*, *anomaly-score* or simply ***outlierness***. This concept represents the degree of a sample of being an outlier. The value of this score may change depending on the chosen method. For the purpose of this project a rule for the visual representation of the *oulierness* is defined: the lower the *outlierness* value, the lower the likelihood of a sample of being outlier; and, the higher the *outlierness* value, the higher likelihood of being outlier. An example can be seen in the Figure 5.2.

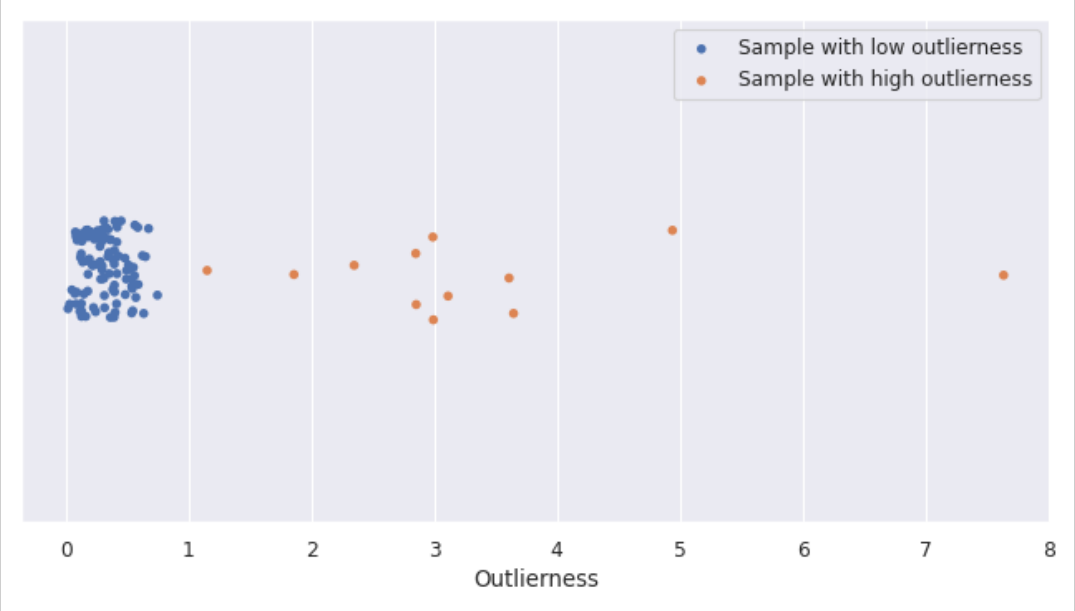


Figure 5.2: *Outlierness* representation example. Toy example for an expected outcome of a One-Class Classification problem: the samples in blue get the lowest *outlierness* and they are represented quite clustered, whilst the orange samples get higher *outlierness* and they are all spread out. Thus, blue samples show to be more similar to the data used for training the classifier, while orange samples don't.

It is common that most of the output of One-Class classifiers are in need of a threshold, so the classifier could predict whether a sample belongs to a certain class or not. This threshold is set over the *outlierness* obtained over the test data for each classifier. This selection is one of the most critical steps when completing the evaluation of OCC models, because it could entail the difference between a good and a bad classification. Thus, this is why the **representation of the outlierness** and the selection of the threshold is so relevant in OCC.

In the previously mentioned work [6] by Tax, the author did not provide a variable similar to outlierness. Instead, the author defined two values to measure how similar is a sample $\boldsymbol{z}$ compared to the target data, previously represented by the training set (made up of positive samples): distance $d(\boldsymbol{z})$, which represents the distance of a sample $\boldsymbol{z}$ to the target class, and the resemblance (or probability) $p(\boldsymbol{z})$, which is defined as the probability of a sample $\boldsymbol{z}$ belonging to the target class.

The author proposed a way to decide whether a new sample $\boldsymbol{z}$ belongs to the target class or not based on setting up a threshold θ_d (for distance) and θ_p (for resemblance):

$$C(\boldsymbol{z}) = \begin{cases} 1 & d(\boldsymbol{z}) < \theta_d \\ -1 & \text{otherwise} \end{cases} \tag{5.1}$$

$$C(\boldsymbol{z}) = \begin{cases} 1 & p(\boldsymbol{z}) > \theta_p \\ -1 & \text{otherwise} \end{cases} \tag{5.2}$$

The approach defined in Equations 5.1 and 5.2 summarizes the way of making the "predic-

tion" step[3] for the most of OCC methods. In this project multiple approaches have been followed for this task, two of them based on the Equation 5.1. More details about each model step are defined in Section 5.2.6.

Once the *outlierness* of each sample is calculated and every label assigned, it is the time of obtaining the **experimental results** and evaluating each model with reliable metrics. This last step does not differ too much from classical multi-class classification evaluation, since the labels of the test samples are known and a probability for each one is learned from the model. The selection of the evaluation metrics depends on many factors, such as the problem's nature (objective, field of work, etc.), availability and unbalancing situations of the data, etc.

As previously discussed, One-Class Classifiers are a common option when working with problems related to anomaly detection, and so, these problems usually suffer from unbalanced data. This is a common situation in machine abnormal behavior detection, where the number abnormal register is critically low. This implies that the evaluation metrics have to adapt to these kinds of situations, giving the minority samples the relevance they lack due to the unbalance. For this reason, the use of metrics which balance the classification of both positive and negative class is critical in OCC.

Another important take in OCC is the selection of the evaluation metric, which depends heavily on the problem's nature. It is not the same situation to evaluate problems about anomaly detection in the machinery and industry field, or evaluate classification errors in health related problems, which is the topic of this project. Every problem type should be evaluated individually, measuring the classification as is convenient for that specific problem. There are some metrics which are widely used along all classification topics. One of those is the AUC (Area Under the Curve) of the ROC (Receiver Operating Characteristic) curve. This metric, apart from summarizing the performance of a classifier in a single metric, it properly describes the performance of a classifier over an unbalanced dataset.

Leaving aside the general evaluation process of OCC, with regard to this project, we have applied more metrics which are descriptive and helpful for explaining the results related to patients' outcome. The metrics used are common when dealing with health related situations, and they're discussed along Sections 5.2.6 and 5.2.7.

5.1.2 Types of OCC methods

When alluding to OCC types, it is proper to mention the work made by Tax (2001) in his PhD Thesis [6] about OCC. In this work, the author proposed a method called Support Vector Data Description (SVDD). These kinds of models are based on decision boundaries, and they will be discussed later. Tax also focused on the establishment of OCC concepts, in particular, the main approaches to solve OCC problems. These approaches had been discussed later in several works [23, 24]. The author proposed (and eventually formalized) 3 approaches: density estimation, boundary methods and reconstruction methods. For the purposes of this project, we will only make one of the methods of the last 2 categories. These methods differ heavily between them, even though each model exploits different characteristics of the data following a specific approach. Their goal is to identify those samples that differ from the data used for model training.

[3] The positive class is usually represented as 1, and on the other hand, the negative class is represented as -1.

Both selected methods are those based on boundaries and the ones based on reconstruction. The methods based on data density were not used as they need certain conditions to be filled in order to show a good outcome. These conditions are the next ones: have a great number of training samples and have a sample size high and flexible. These conditions show that density methods require a clear division between positive and negative samples' distribution, which is not the case in our project. These kinds of methods have a basic but effective working procedure. They are based on selecting a cut-off value on the density of the data, and labeling the samples based on the side of the cut-off of their density.

5.1.3 Boundary based methods

Boundary methods aim to set a decision boundary around a selected group of samples. In order to be correctly set up, the decision boundary usually relies on distances, and as these are defined in an OCC scenario, the decision boundary is built up around the samples belonging to the target (positive) class. These methods ensure to accept all positive samples while minimizing the number of accepted outliers as positive.

Even though there are many method which apply this idea, as seen in the SVDD [17] (Tax and Duin 2004) or One-Class Support Vector Machine (OCSVM) [13] (Scholkopf *et al.*, 1999), there are methods which do not consider a decision boundary in its classical definition: they discriminate samples based only on distance-related metrics. So, this leads to a really common characteristic in the OCC paradigm: label distance based methods as boundary based methods.

So, taking this last point into account, both OCC methods based on decision boundaries were OCSVM and Local Outlier Factor (LOF).

5.1.3.1 One-Class Support Vector Machine (OCSVM)

One-Class SVM [13] (Scholkopf *et al.*, 1999) is a derivation from the classical Vapnik's SVM work made for the good of the progress of statistical learning theory [25] (Vapnik, 1995). The basic idea of the SVM resides on mapping the input data into a high-dimensional feature space, and then, build an hyperplane around the training data that will maximize the distance between patterns, where the hyperplane will serve as a segregation method in that high-dimensional space for the new unseen test samples. SVMs try to minimize the generalization error by the addition of regularization parameters. Later in this section will be discussed deeply the idea of this regularization parameter, but applied to the OCSVM case.

This same concept applies to the OCC version of the algorithm, which basically maximizes the distance of the hyperplane, but in this case the data transformed to the high-dimensional space belongs to a single class, the positive class.

Algorithmic framework The algorithm has as objective to return a function f which has the value +1 in a region which captures the most of the train data points, and -1 outside that region. So, this method implements a solution where it transforms the data into a feature space following a certain kernel, aiming to separate the data from the origin, maximizing the margin. Thus, the function f is computed for a $\boldsymbol{x_i}$ point, so the function $f(\boldsymbol{x_i})$ returns +1 or -1 based on which side of the hyperplane lies the input. It can also be returned the distance to the separating hyperplane.

Let $\mathbb{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\}, \boldsymbol{x_i} \in \mathbb{R}^d$ be the training data. Define a feature map Φ so $\mathbb{X} \to F$. In this way the data can be transformed into a high dimensional feature space: $\mathbb{X} = \{\boldsymbol{x_1}, ..., \boldsymbol{x_n}\} \to \Phi(\mathbb{X}) = \{\Phi_1(\boldsymbol{x}), ..., \Phi_n(\boldsymbol{x})\}$. The idea of representing the feature space resides computing the inner product $\langle \Phi_i(\boldsymbol{x}) \cdot \Phi_j(\boldsymbol{x}) \rangle$ in a function but for the original input points, thanks to a kernel, which can be represented in multiple ways:

$$k(x, y) = (\Phi_i(\boldsymbol{x}) \cdot \Phi_j(\boldsymbol{x})) \tag{5.3}$$

This is where the separation hyperplane appears, which is defined by w, b, w being a weight and b a bias.

Then, in order to separate the data from the origin, a quadratic function with restrictions has to be solved:

$$\min_{w \in F, \xi \in R, p \in R} \frac{1}{2} \|w\|^2 + \frac{1}{\nu l} \sum_i \xi_i - \rho \tag{5.4}$$

$$\text{where } (w \cdot \Phi_i(x)) \geq \rho - \xi_i, \quad \xi_i \geq 0. \tag{5.5}$$

Here, ξ_i represents non zero slack variables to "incorporate" samples located in the incorrect side of the hyperplane, and they are penalized in the objective function. On the other hand, $\nu \in (0, 1)$, is a parameter which defines two concepts: the upper bound (percentage) of incorrectly located samples the margin accepts, and the lower bound (percentage) of Support Vectors (SVs) included in the process. The number SVs of the first condition is calculated with regard to the total number of training samples. The trade-off between the margin and the outliers is controlled by this ν parameter. So having this in mind, the dual problem is derived:

$$f(\boldsymbol{x}) = sgn\left(\sum_i \alpha_i k(x_i, \boldsymbol{x}) - \rho\right) \tag{5.6}$$

In this step the SVs are introduced as the $\boldsymbol{x_i}$ patters with nonzero α_i coefficients, and so, the dual problem is formulated as follows:

$$\min_{\alpha} \frac{1}{2} \sum_{ij} \alpha_i \alpha_j k(x_i, x_j) \tag{5.7}$$

$$\text{where } 0 \leq \alpha_i \leq \frac{1}{\nu l}, \quad \sum_i a_i = 1 \tag{5.8}$$

As seen in Equation 5.7 the kernel is up to be chosen, and it can be represented in several ways (linear, Gaussian RBF, and more).

5.1.3.2 Local Outlier Factor (LOF)

Another distance/boundary based method is the popular Local Outlier Factor (LOF) [26] (Breunig *et al.*, 2000). It is an unsupervised algorithm which consists on finding abnormal data points in a local way, finding points in the data which differs from the local density of their neighbors.

The concept of local outlier represents a sample which differs highly to its neighbors, and so, it is considered an outlier; but neighbor-wise, thus, it is considered a local outlier.

LOF algorithm aims to detect those local outliers based on the density of the neighborhoods that are represented in the data, a degree variable named like the algorithm itself: LOF. This degree increases as the sample is more isolated from the data points of its surrounding neighborhood. This degree is taken in this project as the *outlierness* concept, as it fits in a proper form the definition of "distance" for the segregation Equation 5.1 defined by Tax.

So, this method requires some concepts to be introduced before going after the algorithm of the classifier itself: K-distance (and K-neighbors), Reachability Distance, Local Reachability Density, and finally, the Local Outlier Factor.

K-distance (and K-neighbors) The K-distance is the distance of a certain point to its K^{th} nearest neighbor. The classical approaches are carried out using the Euclidean distance, as it is defined for two arbitrary $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ points:

$$d(\boldsymbol{x_i}, \boldsymbol{x}_j) = \sqrt{\sum_{r=1}^{n}(x_{ri} - x_{rj})^2} \tag{5.9}$$

For a given data point and a certain target point, their distance is computed. Then, for that target sample, the points of the data are sorted based on the distance from smallest to largest, and the first K entries are selected. These K samples are the K closest samples to the target point. And the K-distance is the distance to the K^{th} point of the final list. The K-distance for a certain **a** point is denoted as $N_K(\mathbf{a})$.

For the LOF implementation used in this project, the distance measurement used was the Minkowski distance, which calculates the distance between 2 points in a normed vector space. It is considered as the generalization of Euclidean distance and Manhattan distance. For two given $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$ points and an integer p considered as the order, the Minkowski distance is defined in the following way:

$$d(\boldsymbol{x_i}, \boldsymbol{x_j}) = \left(\sum_{r=1}^{n} |\boldsymbol{x_r i} - \boldsymbol{x_r j}|^p\right)^{\frac{1}{p}} \tag{5.10}$$

Reachability Distance (RD) Once the K-distance is calculated, another concept is defined, the Reachability Distance (RD). The RD is the maximum of the K-distance of a $\boldsymbol{x_j}$ arbitrary point and the distance between $\boldsymbol{x_i}$ and $\boldsymbol{x_j}$, defined in Equation 5.11. These last two points are visualized in Figure 5.3.

$$RD(\boldsymbol{x_i}, \boldsymbol{x_j}) = max(K - d(\boldsymbol{x_j}), d(\boldsymbol{x_i}, \boldsymbol{x_j})) \tag{5.11}$$

Local Reachability Density (LRD) The Local Reachability Density is defined as the inverse of the average of a certain point's RD to its K-neighbors. It is defined as follows:

$$LRD_K(\boldsymbol{a}) = \frac{1}{\sum_{\boldsymbol{x_j} \in N_K(\boldsymbol{a})} \frac{RD(\boldsymbol{a}, \boldsymbol{x_j})}{||N_K(\boldsymbol{a})||}} \tag{5.12}$$

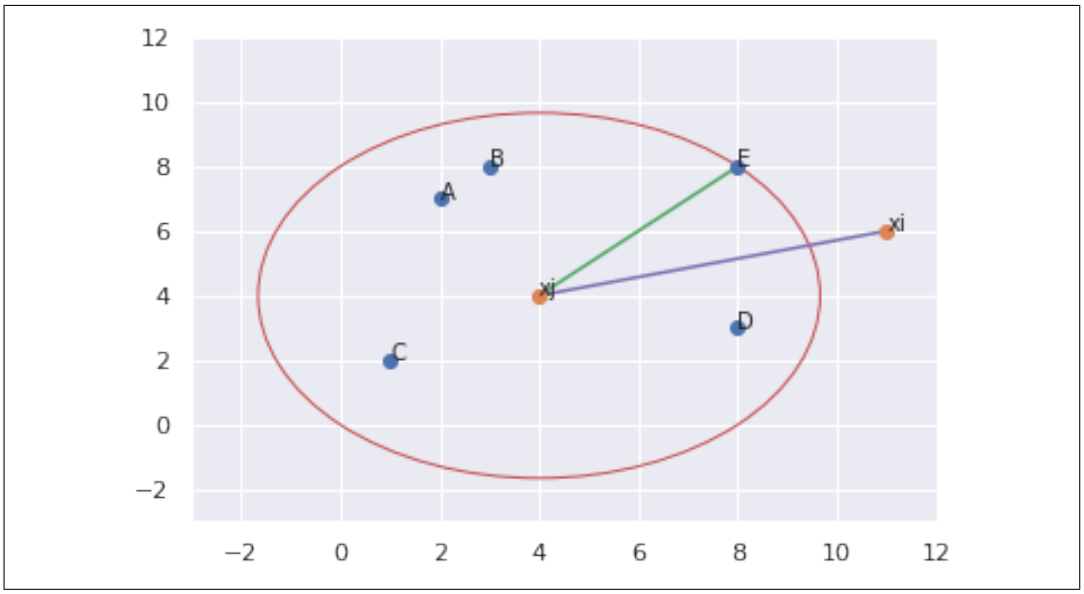


Figure 5.3: An example of K-neighbor representation for a K=5 with the points inside the red circle. The 5th neighbor (E) sets the 5th distance for the $\boldsymbol{x_j}$ data point. It is represented as the reachability distance for the $\boldsymbol{x_i}$ data point, which is the purple line, the maximum between the 5th distance of $\boldsymbol{x_j}$ and the distance between $\boldsymbol{x_j}$ and $\boldsymbol{x_i}$. The distance metric used in this toy example is the euclidean distance.

The interpretation of this factor is made in this way: a low value implies that the closest cluster to the point is far; and a high value represents that the closest cluster is near to the point.

Local Outlier Factor (LOF) Previous concepts sum up to the final equation of the LOF, which is critical at the time of deciding whether a sample is an inlier, or an outlier. LOF value is presented as the ratio of the average LRD of the K-neighbors of a given point **a** to the LRD of that **a** point:

$$LOF_K(\boldsymbol{a}) = \frac{\sum_{\boldsymbol{x_j} \in N_k(\boldsymbol{a})} LRD_K(\boldsymbol{x_j})}{||N_K(\boldsymbol{a})||} \times \frac{1}{LRD_K(\boldsymbol{a})} \tag{5.13}$$

LOF takes value near to 1 for an $\boldsymbol{a}$ point if the ratio of average LRD neighbors is close to its LRD value. This means that the evaluated point $\boldsymbol{a}$ is an inlier. Whilst, higher values of LOF would represent that the LRD of a point is less than the average LRD of the neighbors, implying that the point is an outlier.

In this project, the LOF value is used as *outlierness* value to 'mark' the samples.

Algorithm framework The algorithm for the LOF calculation as follows:

(1) Calculate the K-neighbors and compute the K-distance for each point (5.10).

(2) Compute the Reachability Distance for each point (5.11).

(3) Compute the Local Reachability Density (LRD) for each point (5.12).

(4) Compute the Local Outlier Factor (LOF) for each point (5.13).

(For new data) Compute steps (1), (2) and (3) with the initial data. Compute the step (4) only with the new data.

5.1.4 Reconstruction based methods

Reconstruction methods were not initially designed for OCC. They were originally designed to model the data, and so, obtain better representations of the data. These methods are applied to develop a representation of the data that would, hopefully, mimic the generating process of the underlying distribution of the original data. These methods are developed with the idea of reducing data representation, and so, obtaining a more compact representation of the target data, reducing the impact of noise as the new representation would be more informative about target classes than the original data.

Even though not fitting totally the meaning of reconstruction, the methods of the dimensionality reduction field are commonly considered as reconstruction methods in the classical OCC literature [6]. The main example of these techniques is the Principal Component Analysis (PCA) [27] created by Pearson (1901) but formalized by Bishop (1995) in his work *Neural Networks for pattern recognition* [28]. It is an orthogonal linear transformation that simplifies the complexity of a high dimensional feature space into a smaller but still informative feature space. It is commonly used on problems of pattern recognition. This method is commonly used at the preprocessing step of problems that require an informative representation of the data.

There are methods that apply the concepts of PCA but with a neural network approach, i.e. autoencoders.

5.1.4.1 Autoencoder

The approach of OCC problems with autoencoders fits perfectly the definition of autoencoder, as Japkowicz defined the autoencoder in her work [29] *A Novelty Detection Approach to Classification* about novelty detection by means of autoencoders: "the approach of novelty detection consists of training an autoencoder to reconstruct positive input instances at the output layer and then using this autoencoder to recognize novel instances". The author also proposed that training positive data, is, hopefully, reconstructed in a proper way; whilst, the negative data, is more likely to be reconstructed incorrectly, and thus, recognized as novelty/anomaly/non-positive.

In this project, the input data is made up of positive samples. In this specific case deceased samples, and so, following the description made by Japkowicz, the autoencoder should reconstruct correctly deceased samples. On the other hand, offering an incorrect reconstruction of surviving patients.

The Autoencoder [29] is a type of artificial neural network used for unsupervised learning. The goal of this kind of neural network is to recover the input data by learning an internal representation of that input. It consists of two main parts: one of them is the encoder, which compresses the data into a more compact representation following an encoder function $h = f(x)$. The second part is the encoder, which produces the reconstruction of the data from the latent representation in the hidden layer, which is made thanks to a reconstruction function $r = g(h)$. Even though autoencoders are designed to

not be able to fully copy the input of the data, they're designed in a way that the neural network is restricted to prioritize aspects of the input data that still resemble the initial data, but only the useful properties of it are learned. At their origin they were mainly used for dimensionality reduction or feature learning. However, nowadays they're used for many tasks such as denosising, or, as in this project, for anomaly detection applied in a OCC approach.

Their architecture is based on the input data, the hidden layers, and the output data. The dimensions of the input data and output data are the same. The hidden layers are formed by a minimum of 1 layer, with learned-latent features in the intermediate layer, which include the internal representation of the autoencoder of the initial data. The number of neurons in the encoder side goes from more to less, and the number of neurons in the decoder side goes from less to more. A common type of architecture can be seen in Figure 5.4.

However, there is a large variety of different types of proposed autoencoders, many of them used in OCC. One example of this is the Variational AutoEncoder (VAE), which encodes an input as a distribution over the latent space, instead of encoding it into a single point as happens in classical autoencoder. One example of the VAE is the work made by Khalid (2020) for detection of deepfakes using VAE [30] in a OCC approach, which is a clear example of the use of autoencoders in image processing.

This project implements a OCC approach for the shallow autoencoder.

Algorithm framework The autoencoder measures the quality of a reconstruction with the reconstruction error, a concept that resembles the *outlierness* idea.

This is a classical autoencoder workflow:

(1) The input data is defined as $D_{train} = \{\mathbf{x_1}, ..., \boldsymbol{x_n}\}$.

(2) The input data is encoded by means of a f function: $h = f(D_{train})$. The latent space is defined.

(3) The data in the latent space is decoded by means of a g function: $g(f(D_{train}))$.

(4) The learning process of the neural network is described by minimizing a loss function: $L(D_{train}, g(f(D_{train})))$. Where L is commonly represented by the mean squared error (MSE):

$$MSE = \frac{1}{N}\sum_{i=1}^{n}(\boldsymbol{x_i}, \boldsymbol{y_i})^2 \text{ where } \boldsymbol{x_i} \in D_{train},\ \boldsymbol{y_i} \in g(f(D_{train}))) \tag{5.14}$$

(5) The reconstruction error of a new sample is the MSE of the sample. We also have the reconstructed version of itself.

5.2 Strategy

In this project a total of 3 One-Class classifiers have been implemented, with the objective of showing the performance of a classifier designed for OCC in contrast to classical supervised

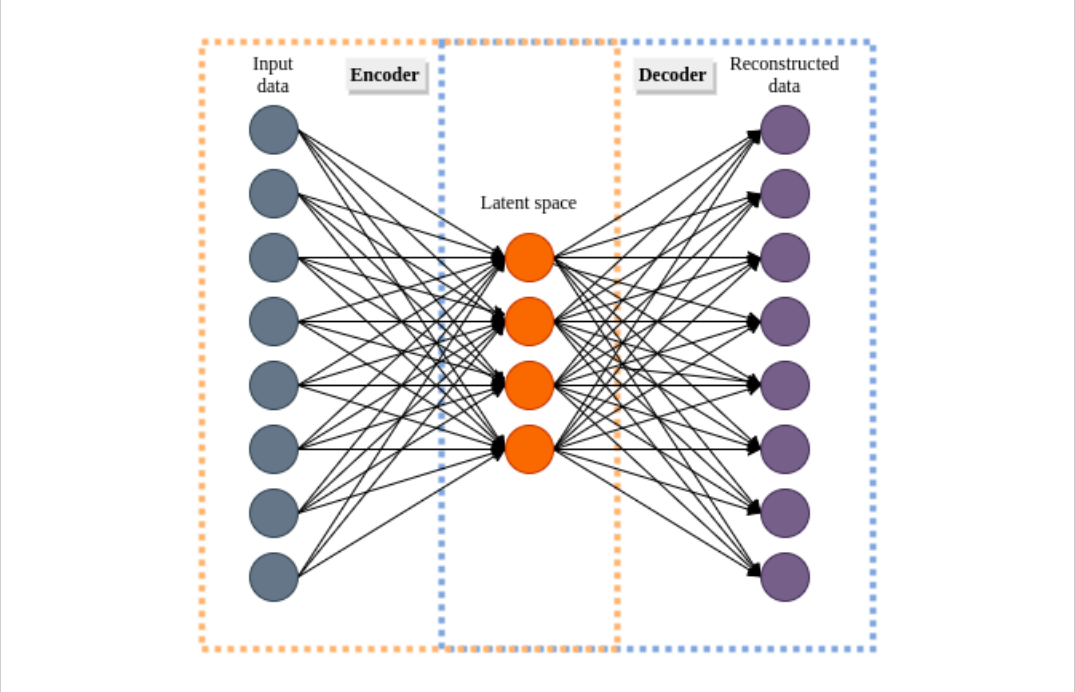


Figure 5.4: Example of a classical autoencoder structure. It is made up of the input nodes, which include the original data, the latent space, which is the internal representation of the data learned by the Autoencoder, and the output nodes, which result in the reconstructed data of the autoencoder. This architecture is an undercomplete autoencoder, a type of autoencoder whose hidden dimension is less than the input dimension.

classifiers. These three classifiers are the previously mentioned ones: One-Class Support Vector Machine (OCSVM), Local Outlier Factor (LOF) and autoencoder (AE). Every of those classifiers need certain requirements to be fulfilled in order to work in a proper way. These include the need for a compact representation of the data, a proper selection of the features, balanced evaluation metrics, and many more.

These topics are all discussed in this section, where the strategy followed for each classifier is explained in a clear and concise way. The steps explained in the following sections are not exclusive for a specific classifier. In fact, all the 3 classifiers explained previously share many of those steps, but the process followed inside every step may change according to the model's requirements, and some steps are ignored for the same reason. The covered steps extend from the structure and configuration of the data, followed by the preprocessing, and eventually ending with the evaluation of the classifiers.

It is good to remind that the OCC is, in his base, an unsupervised method. Some of the steps of the workflow (e.g. preprocessing step) have been constructed following the paradigm of unsupervised learning, with the objective of developing a workflow loyal to OCC methodology.

For a better understanding of the workflow that will be explained step by step during the following sections, a diagram of the whole process is defined in Figure 5.5.

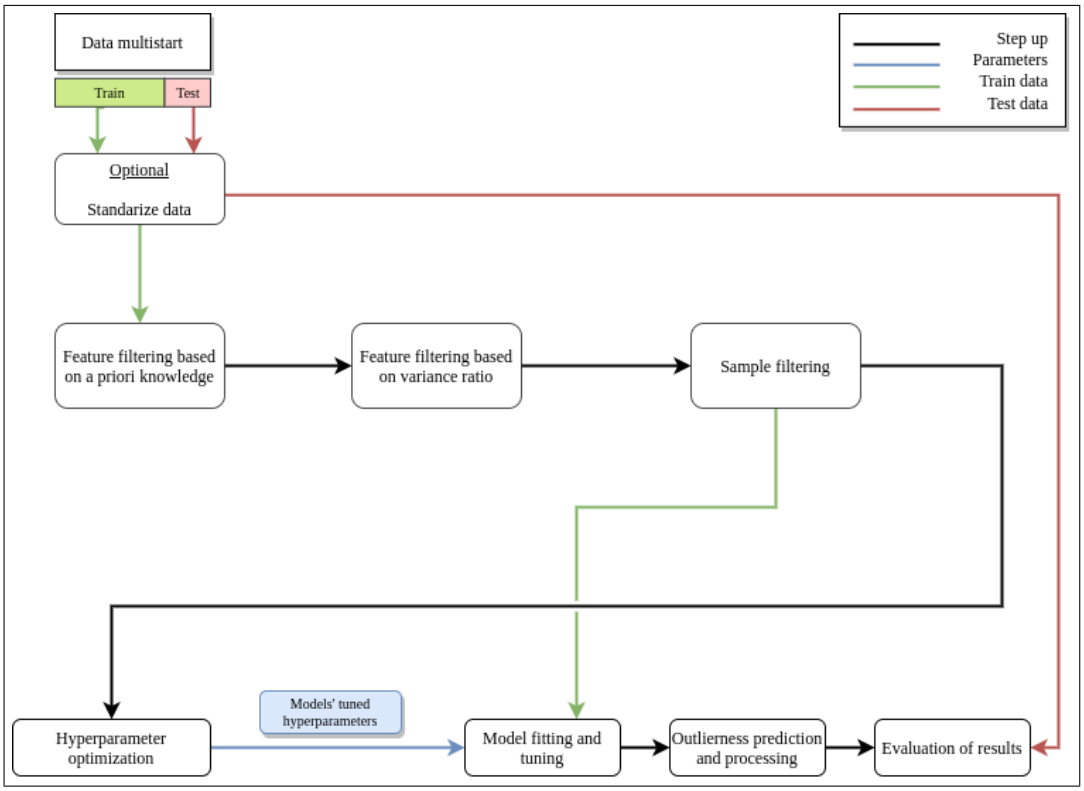


Figure 5.5: The workflow (pipeline) summarizing the whole process of experimentation.

5.2.1 Data structure

The initial data was modified before the start of the pipeline, with the objective of standardizing all the experimentations.

A multi-start procedure was applied to the initial data. The multi-start procedure is based on creating sets from the initial data leaving the same percentage of class distribution for all generated sets. This process can be interpreted as a repeated hold-out over the same data.

There were created a total of 5 multi-starts[4], generated in a random sampling way. These multi-starts were finally splitted into 80% - 20% train/test stratified splits. Leaving a total of 5 train and 5 test subsets, assigning a single pair of them for each multi-start.

In order to avoid any risk of bias, only the train subset of each multi-start was used during the feature filtering, sample filtering and hyperparameter optimization. And the unseen test subset were left aside for the final models' evaluation steps, like the *outlierness* prediction procedure and the evaluation step.

The multi-start configuration used in this project is the same one that was selected in the supervised approach made in the work [1] by Armañanzas *et al.* These multi-starts are shared for all the experiments, with the objective of keeping a high degree of consistency for all the tests.

Previous to any experimentation, this later data modification was carried out. And, in

[4]The multi-starts were referred to as "universes".

the experiments made with the LOF and the OCVSM, the data was standardized avoiding the risk of information lost in scale differences in the data.

5.2.2 Feature Filtering

The preprocessing of the data is a crucial procedure in machine learning, because any classifier's performance relies greatly on how well the data describes the classes.

This step has an important role when facing problems that occur in data unbalancing situations. An example of this is the curse of dimensionality [31] (Bellman, 1957), a common phenomenon advised in problems with high-dimensional data and low sampled classes. Implying that the representation of a high dimensionality space increases exponentially when the number of features of the data increases [6]. This is clear in the OCSVM, specifically in the use of non-linear kernels[5], which implies that the solving of this kind of kernel can take large amount of time, and it is at risk of suffering overfitting, another common problem caused by data representation.

This phenomenon occurs when a classifier builds a decision boundary extremely tight to the training data, being inflexible for new unseen data. In this situation, the classifier performs correctly for the training data, but shows a large error when working with new unseen test data. Implying that the generalization error is high. This problem is common in situations where the data is defined over a large number of features, becoming worse as the number of features increases, and even more in OCC problems.

Anyways, there's no risk of curse of dimensionality in this project, since the data used for this specific problem is defined in a space of 44 features. Avoiding the risk of having the curse of dimensionality, but it can still be potentially subject to overfitting.

In addition, regarding to problems related to anomaly detection, having a smaller (more compact) representation of the data could help obtaining better results at the time of determining anomalies, but on the other hand, this could also imply losing information, thus, losing the data that differ more from the the normal data, classifying these examples as not anomaly. This is totally related to the terms of this project, where an over-compacted representation of the data could result in a diffuse representation of the whole samples, meaning that there wouldn't be a clear separation between deceased and not deceased samples anymore.

Having this in mind, the preprocessing was divided into a two-step process. The first part about identifying the most salient subset of features based on previous knowledge. This information was obtained from the observations made over the distribution of the features (per class), and from similar works [32, 33]. And the second filtering technique is based on analyzing the variance of each sample. This later approach is based on the information the variables provide, taking into account whether the samples are positive or not. And in this way fitting to the OCC paradigm.

Both methods were applied one after another for every classifier, however, the internal procedure of the variance-related statistic differ adapting to each classifier's requirements.

This process was defined as the first stage of the main pipeline, and it executed for all 44 features from the initial data.

[5] Which is the case of this project, where a RBF kernel is used.

Variance ratio analysis There are many methods that provide good results at the time of the selection of subsets of features. Some of those methods are based on supervised paradigms, making the selection of an optimal subset of features based on certain feedback obtained thanks to the knowledge about samples' labels. One example of this is making decisions based on the performance of a classifier (wrapper), or based on the relationship between the features and the class (filter). These methods require the true labels of the data they're working with in order to be applied. However, there are methods which do not need the labels of the samples, many of them based on analyzing how redundant they are, or on how the correlation between features is.

The approach followed in this project is defined over 2 techniques. First of all the features are ranked based on a statistic based on variance, and later the optimal subset of features is obtained following a wrapper criteria.

A quick summary is explained in Algorithm 5.1. Once understood the main idea, is time go ahead with the details about the process:

- For every feature a statistic that summarizes how informative some features are towards the positive samples is computed. This statistic is obtained from the variance a feature has over all the samples and the variance the same feature has only over the positive samples. As the class of the samples is used for the calculation of the statistic, this approach is considered as supervised. However, is good to remind that usually in OCC problems the positive class is known, but the other non-positive classes may not be labeled. So, even though the OCC paradigm is considered unsupervised, this supervised approach would be viable in every OCC problem.

  This statistic, named variance ratio and calculated in Equation 5.15, represents how informative a feature is to positive samples.

$$\sigma^2_{ratio} = \frac{\sigma^2_{total}}{\sigma^2_{positive}} \tag{5.15}$$

  The equation's result can be interpreted in the following way: lower ratio value means that the variance of a certain feature for the positive samples is greater than the variance for all (positive and negative) samples, thus, the feature is less variable for positive samples, and in fact, more informative; whilst, greater ratio values represents a higher variance for all samples, and lower variance for positive samples, so that feature is more variable for positive samples.

- The variance ratio is processed for all features of the data space, keeping aside the features that were previously kept in the *a priori* knowledge step, if it was previously applied. The features are sorted from lower to greater ratio.

- These calculations give raise the final step, the selection of the features. As mentioned, this last procedure aims to select the best features based on the *variance ratio* and how it affects the performance of a model.

  To carry this out, a procedure based on the wrapper technique was designed. First of all, a cross validation with 5 folds is made over the training data. Then a constant p is selected, that will define the maximum number of columns to drop during the process. For each fold a total of $p + 1$ models are trained, removing from *0* to p

features. The $l \in [0, p]$ features removed in each model are the features with higher *variance ratio*.

In summary, there's defined a 5-Cross-Validation (5-CV) where every fold is trained/tested with several models. Each model fit by the initial data, but dropping a fixed number of features, from 0 to p features. Every model[6] is trained with the corresponding train subset of the fold.

The *outlierness* of the test subset is processed for all the models. Then, it is processed and the AUC is calculated. This is used to measure the quality of the selection of the subset of features (for every model).

Every fold selects the configuration with better AUC. And then the configuration of features which had the best AUC of all folds is retrieved. Finally, the initial data is modified following the resultant configuration.

5.2.3 Sample/row filtering

It is common to look up for outliers inside the training set. This is a regular approach when working in anomaly detection problems. This step is carried out in order to remove those samples in the training set that could be represented as abnormal for one or more classes. So, at the time of learning a classifier, the classifier would learn a better representation of the data using good distributed samples, avoiding those abnormal samples that made the representation of the data diffuse. In problems that follow a OCC paradigm is quite effective this kind of approach, since the training set is not used in its whole. Thus, the "real" training set is made up of a single class. This allows the use of OCC approach for this specific task, which works properly for anomaly detection, as mentioned earlier.

There are quite a few algorithm that are useful in this kind of scenario for multi-class classification that also work properly for OCC. One of those methods is the previously mentioned Local Outlier Factor, revised in Section 5.1.3.2, which has as main objective to detect samples of the data with lower density of samples in his surroundings, and thus, can be used to detect outliers inside a certain data set, not caring about the true labeling.

In this specific project it was designed a way to detect and remove positive outliers inside the training in an ad-hoc way[7], using a default configuration for the classifier, except for those which require a certain parameter to work properly.

The approach followed in this project for the train/test process is a Cross-Validation technique. It was used to divide the training set into train/test subsets in order to fit and test models with known data, leaving the unseen test data for later evaluation of the main process.

The train data was divided thanks to a CV approach, which was set up with 5 folds. For every fold a model was trained with positive samples of the training subset. Then, in contrast to classical OCC problems, the data to test the model with was only composed by positive samples. In this way, the *outlierness* of each positive sample in every test subset

[6]The configuration of the model is left as default. There's no custom parameter, except for the LOF and the autoencoder, which need some parameters to be defined previously.

[7]Each experiment used an unique classifier, and it was used for every step: feature filtering, sample filtering, hyperparameter optimization and prediction.

Feature filtering

```
1  input: X, Y
2  output: feature_subset
3  results = []
4  ratio_list = []
5  for feature in X^features
6      σ²_total = variance(X[feature])
7      σ²_positive = variance(X_positive[feature])
8      σ²_ratio = σ²_total / σ²_positive
9      insert(ratio_list, (feature, σ²_ratio))
10 rof
11 sorted_features = sort(ratio_list))[0, :]
12 folds = 5-CV(X)
13 for fold in folds
14     X_train, Y_train, X_test, Y_test = fold
15     for l ← 1 to p
16         feature_configuration = drop_last(sorted_features, l)
17         X̂_train = X_train[column_configuration]
18         X̂_test = X_test[column_configuration]
19         Define classifier
20         fit(classifier, X̂_train,positive)
21         outlierness = predict(classifier, X̂_test)
22         AUC = evaluate_AUC(outlierness)
23         insert(results, (AUC, feature_configuration))
24     rof
25 rof
26 feature_subset = results[argmax(results[0,:])]
```

Algorithm 5.1: Feature filtering process

was computed, making the detection of positive samples with high *outlierness* possible. In order classify the test samples an approach based on analyzing the distribution of the *outlierness* was designed. This technique was about selecting a quantile (by default 0.9), and label all the samples (from each fold) with *outlierness* greater than the value at the specified quantile as outlier. An example of this can be seen at Figure 5.6.

Once the samples with *outlierness* larger than the one defined at the defined quantile are identified, they're removed from the final train set.

5.2.4 Hyperparameter optimization

Every classifier need certain parameters of his structure (hyperparameters) to be set up in a specific way in order to result in a good outcome. The process of selection of the

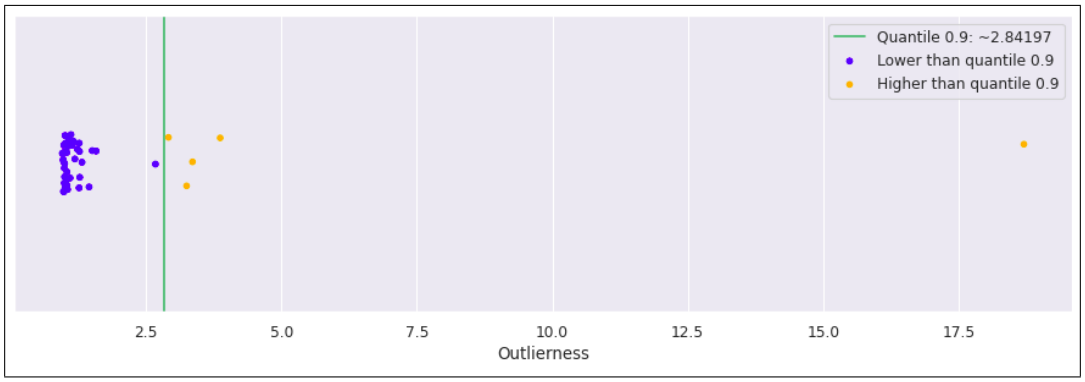


Figure 5.6: Example of *outlierness* analysis for sample filtering step in a certain fold of the 5-CV using LOF. The samples in yellow represent the cases whose *outlierness* is higher the one defined at quantile 0.9. It is clear how a large number of samples is gathered near 0 as expected, since the test subset is entirely made up of positive samples. However, there can be observed samples with higher *outlierness* that will eventually be labeled as outliers, and thus, removed from the train final subset.

hyperparameters differs from one classifier to another, and there is no way to generalize the process for every model. So, the best approach for this problem is to analyze individually each type of classifier the problem requires, and try to adjust them, validating the classifier in an external unseen set, and finally comparing the results, selecting those parameters that provide better results. Clearly there are methods that provide an automatic approach for this problem, as seen in the work NeuroEvolution of Augmenting Topologies (NEAT) [34] (Stanley *et al.*, 2002), which aims to optimize parameters such as the structure (number of hidden layers, number of neurons), batch size, learning rate, and many more hyperparameters of an artificial neural network using genetic algorithms. Another option is the use of exhaustive search methods, which are the most common way of hyperparameter optimization, and the ones that set up the main idea of parameter tuning. One of example of this kind of methods is the GridSearch, which aims for the best configuration of hyperparameters doing an exhaustive search over manually described set of parameters. In order to define a performance metric for this method, it is usually defined a CV over the train data.

In this project it was defined a GridSearch, which was applied together with a nested cross-validation technique.

The nested cross-validation methodology defines a series of train/test splits. It is commonly used in hyperparameter optimization, as it avoid leak of information from train to test, discarding any risk of bias that this problem could lead to. The defined nested-cross validation is presented in the following way:

A stratified 5-CV (80%/20%) is applied to the train set, 5 folds with train and test subsets. And for each fold another 5-CV (80%/20%) is defined. The first CV is usually known as the outer layer, whilst the CV defined at every fold is known as the inner layer. As One-Class models are only trained with positive data, at the time of evaluating models from the outer and inner layer, the samples that are not positive should be discarded, and they could be added to the test subsets.

Once everything is defined, the inner layer classifiers are trained following a GridSearch approach, where the evaluation of every classifier was made using the default prediction

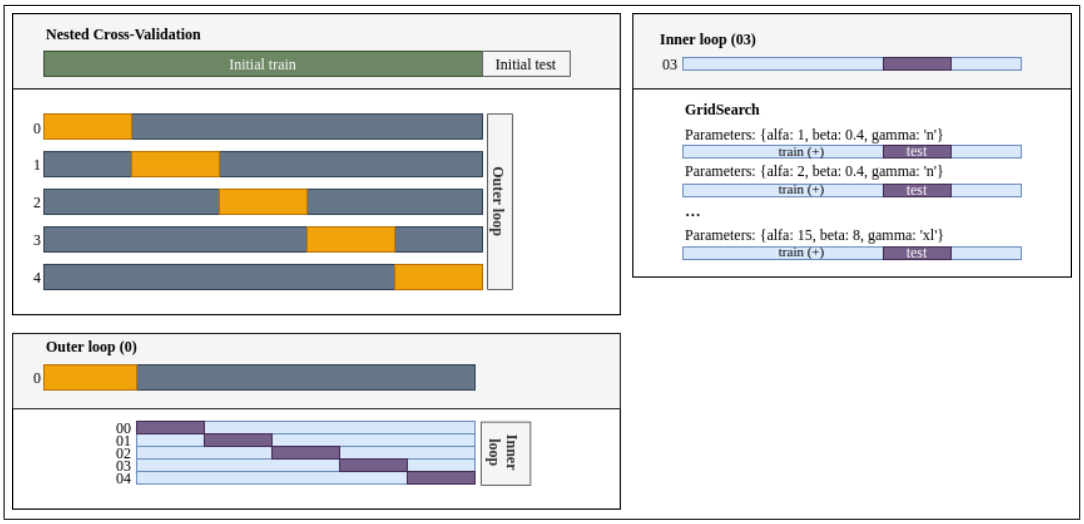


Figure 5.7: Nested Cross-Validation example. GridSearch optimization method works with a set of parameters defined for the hypothetical hyperparameters *alfa*, *beta* and *gamma*·

algorithm[8] of each classifier. For each fold of the inner layer, there is one evaluation for each parameter that were defined for the GridSearch. So, the classifications are evaluated and the F1-score is computed for each classifier. The score of each fold is the maximum F1-score obtained with the GridSearch, saving the configuration of parameters. When the inner layer has been fully processed, the corresponding model from the outer layer it is tuned with the hyperparameters which resulted in the best response (in the inner layers). The outer layers are processed (with every fold tuned with the parameters obtained from their corresponding inner layers), resulting in a F1-score for each model.

Finally, the model with best outcome in the outer layers is selected and its hyperparameters are retrieved. This ends up with an optimized set of hyperparameters that will be set for the final model of the whole process.

For a better understanding of the whole process, a diagram is defined at Figure 5.7.

5.2.5 Model fitting

As mentioned earlier, the training process of One-Class classifiers differs from the training process of classical multi-class classifiers. In contrast to the last ones, classifiers designed for OCC paradigm are trained only with positive samples.

In this process the final classifier of each experiment is modeled. In order to carry this idea out we have to take into account the previous steps, such as the hyperparameter optimization and the data filtering. However, there can be the case that there has not been an optimization of parameters or any filtering to the initial data, thus, in these situations the final model is not getting the parameters tuned or the data to fill the model with is kept as it was originally.

In the case that all previous steps have been followed, the process is quite similar but some differences are clear. First of all, a classifier is defined and it is configured with the parameters optimized previously. Following, the positive samples of the cured data

[8] Set by the chosen implementation package.

(previously filtered) are fitted. To be clear, the fitted data is the training set corresponding to the multi-start with the optimized subset of features in it, and the marked samples as outliers were removed from the training set. In this case, the negative samples that are not used during the training process are not transferred to the test subset of the multi-start, with the objective of keeping the test data as unseen as possible.

This process is equal for all classifier types.

5.2.6 *Outlierness* processing: prediction

The *outlierness* of each model is not meant to be defined in the same way. So, in order to standardize the visual representation of the *outlierness* of the models, the *outlierness* is processed to be positive, meaning that those samples with lower value are more likely inliers, whilst the samples with higher values are more likely outliers, this is previously defined in Section 5.1.1. It is important to note that this transformation is only made for visualization purposes; at the time of processing the *outlierness* for prediction, the values are kept as they were initially calculated by the classifier, and it is during that process where the *outlierness* is transformed.

2 main approaches for the processing of the *outlierness* had been followed on this project:

(a) The prediction process is made by the selection of the cut-off (segregation) value θ for the *outlierness*. θ is selected in two ways:

- θ is selected manually, based on an *a posteriori* observation made on the *outlierness* of the test set. An example of this situation can be seen in Figure 5.8.

- The *outlierness* is processed in order to transform every value as positive[9]. Then, if needed, the *outlierness* is normalized and represented between 0 and 1. The ROC curve is calculated for the test set. Then, the Youden's Index is calculated, which is a criterion for selecting the optimum cut-off value for the ROC analysis, optimizing both the specificity and sensibility. In order to understand the calculation procedure of the Youden's Index, some concepts need to be explained, such as the specificity and sensibility.

  When doing a classification which involves two classes (in this case, positive class and remaining samples, referred to as negative class' samples), at the time of predicting the label of the two classes there can be inferred certain statistical measures for the performance of the classifier. The classification is summarized in a table named confusion matrix, represented in Table 5.1, which is helpful by the time of understanding the previously mentioned measures methods. The classification presents 4 types of results, such as the "True Positive" (TP), which are the positive samples that have been predicted as positive; "False Positive" (FP), which are the negative samples that have been predicted as positive; "True Negative" (TN) which are the negative samples that have been predicted as negatives; and lastly, there are the "False Negatives" (FN), which are the positive

[9]Keeping the next definition: lower the value, lower the likelihood of being outlier, and, larger the value, higher the likelihood of being inlier.

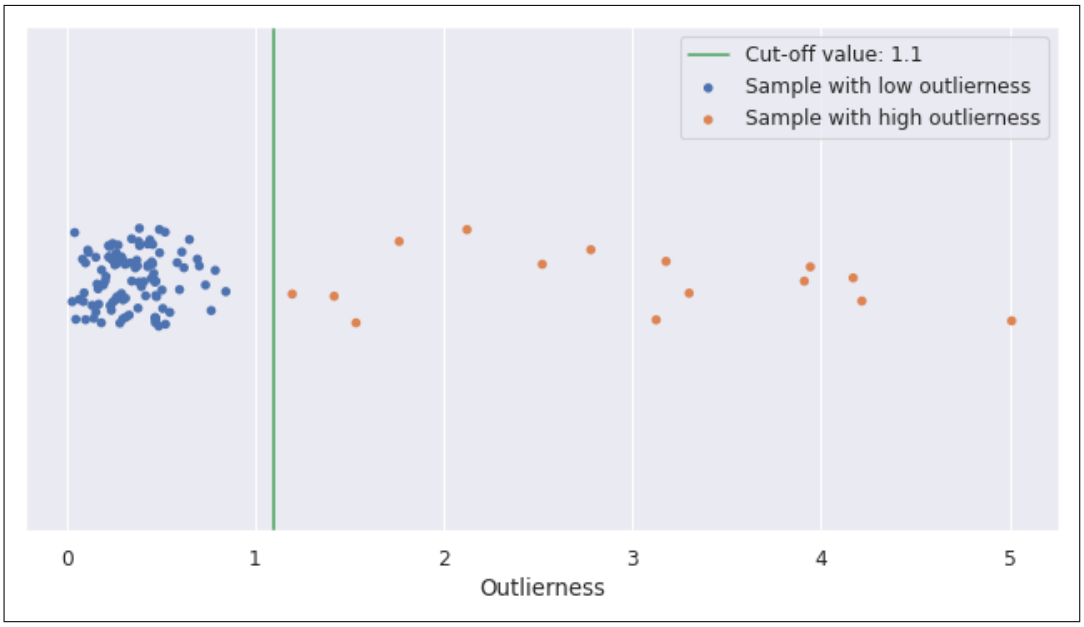


Figure 5.8: Manual cut-off value selection example. Toy example for an expected outcome of an one OCC approach. It is clear that a large number of samples is gathered near 0, forming a small cluster. On the other hand, there are few remaining samples that are all spread out, which are the ones with larger *outlierness*. In this case the selection of the θ cut-off value is in the value 1.1, which is the threshold that separates clearly the samples with small and large *outlierness*.

	$\text{Positive}_{predicted}$	$\text{Negative}_{predicted}$
$\text{Positive}_{g.t.}$	TP	FN
$\text{Negative}_{g.t.}$	FP	TN

Table 5.1: Confusion matrix

samples that have been predicted as negatives. Understanding these 4 values, there can be inferred some measure metrics. For this specific task there are used two lookalike measure metrics: sensitivity and specificity. Sensitivity measures the proportion of positive samples that have been correctly predicted, as seen in Equation 5.16.

$$Sensitivity = \frac{TP}{(TP + FN)} \tag{5.16}$$

Specificity measures the proportion of negative samples that have been correctly predicted, as seen in Equation 5.17.

$$Specificity = \frac{TN}{(TN + FP)} \tag{5.17}$$

Sensitivity is commonly referred to as True Positive Rate (seen in ROC analysis) or recall; and specificity is known as True Negative Rate (seen in ROC analysis). The use of sensitivity and specificity has an important role when working with medical data, this includes the use of more metrics such as *F1-score*, whose meaning and application is widely discussed in Section 5.2.7.

Coming back to the application of the Youden's Index, the ROC curve is calculated from the *outlierness*. Then the Youden's Index is calculated for the

selection of the cut-off θ. This index [35] was suggested by Youden (1950), and was firstly defined as an index for rating diagnostic tests. As explained earlier, this index is used along with the ROC curve, trying to optimize the outcome of certain diagnostic tests (in this case the classification of the test samples). The index, commonly referred to as $\mathbb{J}$, is defined in the following way:

$$\mathbb{J} = sensitivity + specifity - 1 \tag{5.18}$$

Thus, following the previous definition for sensitivity and specificity, the expanded formula is defined:

$$\mathbb{J} = \frac{TP}{(TP + FN)} + \frac{TN}{(TN + FP)} - 1 \tag{5.19}$$

A value 0 for $\mathbb{J}$ indicates that the proportion of positive samples predicted as positives (TP) and negative samples predicted as positives (FP) is the same, and thus, the classification is useless. And, on the other hand, when the index is 1, it indicates that the number of False Positives and False Negatives is 0, therefore, it is a perfect classification.

Finally, θ is assigned with $\mathbb{J}$ for the ROC curve obtained from normalized outlierness. Then, the prediction is made based on the Equation 5.1, taking the normalized *outlierness* as the distance to the positive class.

(b) The *outlierness* is defined in the range $[a, b]_{a<0,b>0}$. a and b could reach infinite and -infinite values theoretically, and they represent symbolic large distances from the decision boundary. The samples whose *outlierness* is defined inside the range $[0, b]$ are predicted as positive. On the other hand, if the *outlierness* is defined inside $[a, 0)$, the sample is predicted as anomaly. This kind of approach is common in decision boundary based methods (e.g. OCSVM). And the explanation of the use of this kind of range is intrinsic to the classifier definition. However, for the approach followed in this project, when a new sample lies inside the decision boundary created during the training process, the new sample is assigned with an *outlierness* with value between 0 and b (both inclusive) when the new sample lies inside the decision boundary made by the classifier with the positive train data; and it gets a value between a and 0 when the sample lies outside of the decision boundary. In this case, as mentioned previously, the *outlierness* is given in a $[a, b]$ range, which represents the distance of a sample z to the separation boundary (hyperplane for OCSVM), $d(z)$ as presented in Equation 5.1. In summary, positive distance is assigned when the sample lies inside the boundary, and negative distance when the sample lies outside. This leads to the selection of θ_d for the project's approach. This approach is used for the OCSVM, which the method itself provides a discrimination method with the distance. So, the θ_d threshold is the value 0, thus, following the next approach, which differs from the ones defined in Equations 5.1 and 5.2:

$$C(z) = \begin{cases} 1 & d(z) \geq \theta_d \\ -1 & \text{otherwise} \end{cases} \tag{5.20}$$

A point to take into account is the fact that the processing of the *outlierness* designed in this project is divided in several categories as previously seen. Thus, the model to work

with should select one of these categories in order to advance this step properly. However, the approach of selecting θ based on the ROC analysis takes advantage of the true target labels for the test set in order to select optimum cut-off value (θ) for the *outlierness*, and so, it couldn't be used in situations where the true labels are not available.

5.2.7 Evaluation

The evaluation metrics used for this project adapt to the intrinsic idea of the problem: medical prognosis. As it is clear in problems like the one treated in this project, where the outcome of a patient is predicted, the main objective resides in correctly identifying those samples with the pessimistic outcome, whereas the idea of misidentifying the samples with the optimistic outcome has less relevance. This implies a direct impact at the time of developing the conclusions of the experimentation.

In this project this is entirely translated to the prediction of deceased/not deceased situations of the inpatients. This idea is represented in multiple related works of outcome prediction for COVID-19 patients, such as the ones mentioned in the chapter 3, which discussed the idea of giving relevance to the correctly classifier deceased patients. Many of these works present some countermeasures for the treatment of this issue, such as post-classification calibration and/or the selection of reliable measure metrics. In this project it was followed an approach which worked with these 2 ideas.

In previous steps there were presented a few ideas revolving around the fact of compensating the good classification of certain samples in front of other ones. An example of this can be seen in Section 5.2.6, in the use of techniques as Youden's Index, which aims to select a cutoff value for a given results in order to optimize the sensitivity and specificity values. And so, choosing metrics which punish the misclassification of positive samples, and the misclassification of negative samples, respectively. Following this main idea, another metric is presented: the F1-score.

This metric is defined as the harmonic mean of precision and recall (sensitivity), another 2 common metrics in binary classification, defined in Equation 5.21. The precision is a metric which is defined as the ratio of number of correctly predicted positive samples to the number of samples predicted as positive (both positive and negative samples), it is defined in Equation 5.22.

$$F_1 = 2 * \frac{precision * recall}{precision + recall} \tag{5.21}$$

$$Precision = \frac{TP}{(TP + FP)} \tag{5.22}$$

Having the precision and sensitivity explained is easier to understand the main idea of F1-score. This metric is sometimes criticized as it doesn't consider True Negatives in its formula, and this can be messy when working with medical data, which gives great relevance to those misclassified negative samples. But, as mentioned earlier in this section, the problem approached in this project focuses mainly on correctly predicting positive samples. So, the use of this metric is justified by the time of comparing the results of the experiments with other works, where the use of this metric is quite common.

In summary, the metrics used for the evaluation task are the next ones: ROC-AUC, Sensitivity, Specificity and F1-score.

5.2.8 Implementation: pipeline

The implementation of the whole process was made in a *Jupyter Notebook* computational environment, all written in *Python* language. The main packages used for the data management were *pandas* (1.2.1), the main packages used for classification tasks were *sklearn* (0.23.2) and *pyod* (0.8.7), the package used for mathematical purposes was *numpy* (1.19.2) and the packages used for visualization purposes were *matplotlib* (3.3.2) and *seaborn* (0.11.1).

For a better understanding, and looking forward to the reusability and modularity of the implementation, the whole process of experimentation was wrapped into a single class, creating a pipeline for further experimentations. The pipeline was made up of the next Sections (independent between them): feature filtering (5.2.2), sample filtering (5.2.3), hyperparameter optimization (5.2.4), model fitting (5.2.5, outlierness-processing (5.2.6) and evaluation (5.2.7). The first 3 modules are independent from each other, and they can be mixed, or even not executed some (or all) of them.

The whole implementation was saved in a *GitHub* repository: https://github.com/unaicar-bajo/one-class-covid19

5.2.9 OCSVM

For the OCSVM it was computed as a single experiment with hyperparameter optimization. The selected OCSVM was the one from the *sklearn.OneClassSVM* package.

In order to avoid scale problems during the computing of the model, the data was initially standardized by removing the mean and scaling to unit variance. The standardization was carried out for each feature $\boldsymbol{x_j}$, transforming the value $x_{i,j} \in \{x_{i,1}, ..., x_{i,n}\}$ of each sample's feature, following the next equation:

$$x_{i,j} = \frac{x_{i,j} - \nu_j}{\sigma_j} \tag{5.23}$$

Where ν is the mean of the training samples for the j^{th} feature, and σ is the standard deviation of the training samples for the j^{th} feature.

Once the experiment is computed with the configuration presented in Table 5.2, the obtained model was set up in the following way:

- The feature filtering process resulted in a total of 19 features to remove of the universe 42, and 16 of the universes 89, 151 and 6 of the universe 189. The sample filtering process turned out to remove a total of 25 samples of the training set of each universe (quantile 0.9)[10].
- The optimized parameters of the OCSVM are presented in Table 5.3.

5.2.10 LOF

The LOF algorithm was applied once, following the same process as the experiment of the OCSVM, which included all the steps defined at Section 5.2.8. Despite not being the

[10] As the *outlierness* of the OCSVM is represented from negative (outlier) to positive (inlier), the removed samples are the ones with *outlierness* lower than the quantile 0.1.

Feature filtering		
Features to keep (a priori knowledge)	Feature name	Description
	CD0000AGE	Age of the inpatient
	CTHSDXXTEMP	First temperature record taken at Emergency
	CTHSDXXSAT	First record of oxygen saturation taken at Emergency
	CTHSDXXRATE	First heart rate record taken at Emergency
Feature filtering method	*Variance Ratio*	
Outcome performance metric	*ROC AUC*	
Number of columns to drop	*20*	

Sample filtering	
Discard quantile	*0.9*

Hyperparameter optimization		
Parameters	*nu*	*{0.01, 0.02, 0.03, 0.04, 0.05, 0.06, 0.07, 0.08, 0.09, 0.1 , 0.11, 0.12, 0.13, 0.14, 0.15, 0.16, 0.17, 0.18, 0.19, 0.2 , 0.21, 0.22, 0.23, 0.24, 0.25, 0.26, 0.27, 0.28, 0.29, 0.3 , 0.31, 0.32, 0.33, 0.34, 0.35, 0.36, 0.37, 0.38, 0.39, 0.4 , 0.41, 0.42, 0.43, 0.44, 0.45, 0.46, 0.47, 0.48, 0.49}*

***Outlierness* processing**	
Cut-off value selection	*OCSVM's internal classification method*

Table 5.2: Configuration summary of OCSVM.

	Universe					
Parameter	*42*	89	101	151	189	Common
nu	0.04	0.01	0.01	0.02	0.01	-
gamma	-	-	-	-	-	*scale*
kernel	-	-	-	-	-	RBF

Table 5.3: Optimized parameters for OCSVM.

Feature filtering		
Features to keep (a priori knowledge)	Feature name	Description
	CD0000AGE	Age of the inpatient
	CTHSDXXTEMP	First temperature record taken at Emergency
	CTHSDXXSAT	First record of oxygen saturation taken at Emergency
	CTHSDXXRATE	First heart rate record taken at Emergency
Feature filtering method	*Variance Ratio*	
Outcome performance metric	*ROC AUC*	
Number of columns to drop	*25*	

Sample filtering	
Discard quantile	*0.9*

Hyperparameter optimization		
Parameters	*number of neighbors*	*{1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24}*

***outlierness* processing**	
Cut-off value selection	*Younden's index of ROC AUC*

Table 5.4: Configuration summary of LOF.

common approach for the LOF technique, the algorithm was applied in 5-CV situations, following a train/test structure instead of applying the technique just to one subset (as train/test). In this way, the LOF is transformed to fit the OCC methodology, following a positive train and negative test example. The implementation of *pyod.models.lof* package of the LOF was the one used for this experiment.

As LOF is a distance-based algorithm, the data was initially standardized following the classical standardization process of the Equation 5.23. In this way the problems of scaling during the computing are avoided.

The pipeline is configured 5.4, and the model is set up with the resultant parameters:

- The feature filtering process resulted in a total of 15, 9, 12, 13 and 13 features to remove from the universe 42, 89, 101, 151 and 189 respectively. The sample filtering process turned out to remove a total of 25 samples of the training set of each universe (quantile 0.9).

- The optimized parameters of the LOF are presented in Table 5.5.

	Universe					
Parameter	*42*	89	101	151	189	Common
number of neighbors	18	19	18	18	19	-
metric	-	-	-	-	-	*minkowski*

Table 5.5: Optimized parameters for LOF.

5.2.11 Autoencoder

Despite of the OCSVM and LOF, the experiment of the autoencoder did not include the step of hyperparameter optimization, since the main hyperparameters of the autoencoder were the architecture of the neural network itself, and, as mentioned in Section 5.2.4, the optimization of the hyperparameters of neural networks could be difficult to carry out with classical approaches, and it should be made with approaches that make use of more advanced techniques. So, the architecture of the autoencoder was defined previous to the experiment, and it was not changed during the process. The architecture of the autoencoder was composed by 32 neurons for the input and output layers (first and last hidden layers), and 16 neurons for the central hidden layer, all of them fully connected. One problem to avoid when designing the architecture of an autoencoder is the excessive compactness of the latent space, which is clear when the number of neurons of the layer defining the latent space is too small, thereby reducing the dimensions of the initial data in such a way that the loss of information is critical. Another point to take into account is that the autoencoder serves as feature filter, and so, the step of feature filtering shouldn't be that relevant, thus, the configuration defined for the feature filtering of this experiment was almost insignificant parameterwise: a small number of features were filtered. The package used for the implementation of the autoencoder was *pyod.models.auto_encoder*.

So, having the initial configuration of the autoencoder defined, and the pipeline defined (Table 5.6), the obtained model was set up in the following way:

- The feature filtering process resulted in a total of 8 features to remove from the universe 89, and 9 features from the universe 42, 101, 151 and 189. The sample filtering process turned out to remove a total of 25 samples of the training set of each universe (quantile 0.9).
- The optimized parameters of the autoencoder are presented in Table 5.7.

Feature filtering		
Features to keep (a priori knowledge)	Feature name	Description
	CD0000AGE	Age of the inpatient
	CTHSDXXTEMP	First temperature record taken at Emergency
	CTHSDXXSAT	First record of oxygen saturation taken at Emergency
	CTHSDXXRATE	First heart rate record taken at Emergency
Feature filtering method	*Variance Ratio*	
Outcome performance metric	*ROC AUC*	
Number of columns to drop	*10*	

Sample filtering	
Discard quantile	*0.9*

Hyperparameter optimization
-

***outlierness* processing**	
Cut-off value selection	*Younden's index of ROC AUC*

Table 5.6: Configuration summary of the autoencoder.

	Universe					
Parameter	*42*	89	101	151	189	Common
Architecture	-	-	-	-	-	32-16-32
Loss function	-	-	-	-	-	*Mean Squared Error*
Optimizer	-	-	-	-	-	Adam optimizer
Batch size	-	-	-	-	-	32
Dropout rate	-	-	-	-	-	0.2

Table 5.7: Optimized parameters for the autoencoder.

CHAPTER 6

Experimental results and discussion

6.1 Evaluation (scoring)

The evaluation of the mortality prediction is a process that has to be executed carefully, as some medical aspects need to be taken into account. The metrics used for the evaluation of the experiments were the ROC-AUC, Sensitivity, Specificity and F1-score, which are widely explained in Section 5.2.7. In addition to these metrics, the confusion matrix is shown, which gives a general perspective for every classification made.

The complete collection of results of the 3 experiments (over 5 multi-starts) is collected in Appendix B, at Table 3, Table 4 and Table 5. From these outcomes, an overall results are computed over the measurements obtained for each universe with every classifier.

After the computing of the OCSVM experiment, the results showed a ROC-AUC of 0.445 ± 0.023, a sensitivity of 0.607 ± 0.045, a specificity of 0.219 ± 0.058 and a F1-score of 0.204 ± 0.010. This test showed that a high number of both positive and negative samples were predicted as positive, manifesting a really low specificity but high sensitivity.

Just like the OCSVM, a test was computed for the LOF. It turned out to reflect an overall ROC-AUC of 0.696 ± 0.022, a sensitivity of 0.400 ± 0.106, a specificity of 0.268 ± 0.093 and a F1-score of 0.144 ± 0.023. In contrast to OCSVM, the LOF balanced the prediction of the samples, instead of predicting the majority as positive.

The autoencoder was also computed, showing that the estimated ROC-AUC of the autoencoder prediction was 0.525 ± 0.021, the sensitivity was 0.694 ± 0.107, the specificity was 0.140 ± 0.109, and lastly, the F1-score was 0.214 ± 0.010.

6.2 Interpretation of results and external study comparison (supervised approach)

Given the results, it is clear that the overall performance of the One-Class Classifiers turned out to underperform. Analyzing each classifier can be observed different behaviors that

happened during the classification:

- The classification made with the OCSVM resulted in a large imbalance on the positive and negative prediction. The high number of samples predicted as positive was reflected in an overall high sensitive but really low specificity over all universes. This results lead to several conclusions about the classifier. One of them is that the hyperplane fit the positive data but it left so much room for negative data to fit inside. These last problem may occur due to several irregularization: the data may not be that informative for the positive samples, since it may be corrupted by abnormal positive samples, implying that the OCSVM could not create an optimized hyperplane covering the positive class. Another explanation for this outcome could be that the selection of the hyperparameters may not be right.

- The LOF showed a more balanced evaluation, which was reflected in a better overall ROC-AUC than the other classifiers. Even though it was more balanced, the classification showed the same problem as the OCSVM and the autoencoder, the number of False Positives was still too high. And, if it was not enough, the number of False Negatives, which is the value to care most about in this specific project, was higher than the number of True Negatives for 4 of the 5 universes. These last two facts can be summarized in the F1-score, which resulted to be really low for all universes. So, the classification of the LOF was bad too.

- Lastly, the autoencoder showed a similar performance to the OCSVM. It shared the fact that the evaluation turned out to classify a large number of samples as positive, reflected in a high number of False Positives. As well as the OCSVM, the classification made by the autoencoder also showed a lower number of False Negatives in contrast to True Positive, which is a good sign, since it represents that the number of inpatients whose outcome is deceased, are correctly classified as deceased. But, as happened in all 3 experiments, the number of False Positives is really high.

None of the experiments showed any consistency through all the attempts, differing the results too much one from another.

All experiments shared the same problems. Many of them can be addressed from several waypoints. A common problem seen through all three experiments, is the large number of negative samples predicted as positive, *i.e.*, the large number of False Positives. As mentioned in the analysis of results of the OCSVM, the explanation of this behavior can be taken from two main sections. One of them is the configuration of the model itself, which is a really important point to take into account when working with highly parameter dependant models, such as the OCSVM, whose computation is entirely based on how well the parameters are tuned. The other two classifiers are not that parameter dependant as the OCSVM. So this leads to the second hypothesis, that is the structure of the positive data, how it is defined, and how well is the segregation between those samples and the negative samples.

The last mentioned issue may be the main problem that caused the bad outcome of the experiments. In previous analysis of the data, it was observed that many inpatients (deceased and discharged) shared a lot of similarities. A clear situation is that many inpatients that ended up passing away, despite being COVID-19 positive, they had an

optimistic diagnosis, and they were eventually registered as deceased for the record. A great number of these samples appeared during the sample filtering made over the three classifiers. Therefore, these two analyses were made in the try of explaining the bad results obtained with the classification. One of them was about the outliers inside the positive data, and the other one was about how well the positive/negative segregation was in the initial data.

It is proper to compare the results obtained following an OCC paradigm (unsupervised) to the supervised approach [1] made by Armañanzas *et al.* The authors of this later work developed a probabilistic model making use of two regularization learners: logistic regressor and lasso regressor. The data and multistart configuration in their project is the one used in our project. So, results can be compared for the 5 multi-starts.

The authors reported the overall results of the classification. The ROC-AUC was 0.906 ± 0.015, the accuracy was 0.768 ± 0.026, the sensitivity was 0.745 ± 0.031, and the F1-score was 0.544 ± 0.028. The complete record of results is defined in the Appendix B at Table 6.

The results obtained in this last work show a really low number of False Negatives, reflected in a high sensitivity over all multistarts. In the same way a great specificity is shown due to the large number of True Negatives in contrast to the lower number of False Positives. This is where the supervised approach truly succeeded, since it could properly identify a large majority of death samples while still properly classifying survival samples. However, the number of survival patients classified as dead is still high. The reason of this behavior is mentioned earlier in this section. Addressing that problem through two main issues: the configuration of the model and the quality of the representation of both dead and survived patients. As the techniques used in the supervised approach differ heavily from the OCC approach, the main explanation for the results may reside in the poor representation of the data, which is analyzed in Sections 6.3 and 6.4.

However, it is clear that the supervised approach outperforms the unsupervised approach. Showing how important is the information gathered from deceased samples in this kind of problem.

6.3 Positive outlier analysis

An analysis was made over the samples identified as outliers inside of the positive training subsets of every universe. This analysis was carried out with an LOF [1] set up with the parameter of *neighbors* as 20. This analysis was executed following this steps:

(1) *A priori* knowledge filtering

(2) Feature filtering with *variance ratio*. Evaluated with ROC-AUC.

(3) LOF model train and tested. Both train and test are made over the same subset of train of each universe, following the classical LOF algorithm.

(4) Samples over quantile 0.9 of outlierness identified as outliers of positive samples.

[1] This post-classification evaluation could be also made with OCSVM and autoencoder. But for methodological reasons, it was only made with LOF, which fits perfectly the objective of this analysis.

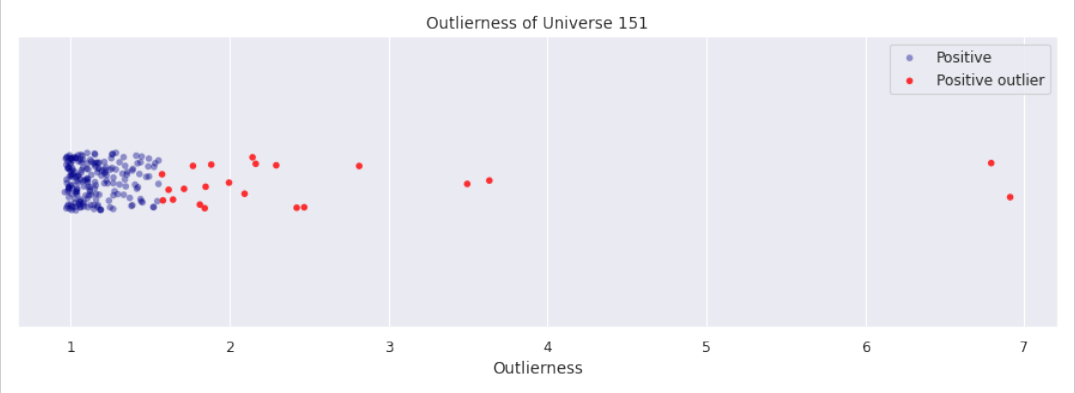


Figure 6.1: Outlierness representation for outlierness analysis with LOF. A lot of samples are gathered near 1 (LOF lowest possible value), which represent inliers, whilst there are less samples with higher outlierness which represent positive outliers in the positive data of the training set.

It is right to highlight that the data was previously standardized, since this kind of methods based on distances suffer from issues when working with data of different scales.

After the process execution, a total of 14, 9, 17, 12 and 7 features were removed in the universes 42, 89, 101, 151 and 189 respectively. So, having the resultant features, as explained previously, the outlierness was processed. The outlierness obtained for the universe 151 can be observed in the figure 6.1, where it is clear how a large number of samples have low outlierness, compared to small numbers of samples which have higher values.

Having this outlierness, the positive outliers were identified[2]. And thus, the data of these samples with the selected features was analyzed looking for patterns that could explain the existence of these abnormal positive samples.

The analysis made over the data turned out to reflect some abnormal values of certain features that could explain this behavior: the feature "CT00000DD", which is the D-dimer value in blood for the inpatient, took the value of mean for the positive samples of 4.083,607, and for the negative samples of 1.829,778. So, the samples from the analysis obtained an overall 1.419,187 value, which is way closer to the negative samples than to the positive ones. This specific feature is really important when predicting the prognosis for COVID-19 patients, because it increases along with elevated inflammation degrees, seen in COVID-19 severe patients, and that's why it is a key parameter for prediction, and why it could be the main reason for these positive outliers. A study for this inflammation related features can be seen in the work of Zhang *et al.* (2020) [32].

Another case would be the feature "CT0000LDH", which is the level of lactate dehydrogenase of the inpatient, that is presented in a really higher overall value (1.353,067) than the overall value for the negative samples (558,364), being closer to the overall value for the positive samples (836,772), but, this does not mean that it is more informative for the positive class. The extreme values for the feature do not fit to the overall representation on the positive data, neither fit for the representation of the negative data. This shares as relevance as the D-dimer value, since a higher value of LDH is directly related to less

[2]For analysis purposes only the samples that were identified more than 3 times along all universes were chosen for analysis.

favorable response to outcome of COVID-19 patients. An exhaustive analysis of this feature can be reviewed in the work by Juan *et al.* (2020) [33], where is collected an analysis made over 94 COVID-19 patients and the correlation between biochemical parameters and COVID-19 patients' outcome.

The "CT0000PCR" was another feature that was heavily impacted at the time of predicting the outcome of COVID-19 patients. This feature represents the level of C-protein in blood, where higher values are present in acute infection or inflammation, and as it increases, it indicates a more severe infection, thus, more severe outcome of COVID-19 patients [36]. This was heavily reflected in the samples detected as positive outliers inside train sets, since the overall value obtained in these cases (97,402) was closer to the negative samples (89,568) than to the positive ones (164,050).

More (not that relevant) features can be analyzed that show a closer overall value to the negative class than to the positive class. For instance, the "CT00000NA" feature, which defines the amount of sodium in blood, had an overall value of 137,902 for the positive class, 136,738 for the negative class and 134,133 for the identified outliers.

All this sum ups to the explanation that many samples which cover the properties explained previously, are those patients that initially had an overall optimistic diagnostic, but they ended up deceasing due to COVID-19, or some other unknown reason. These samples represent the more noisy point of the positive data, and they make the training more diffuse at the time of fitting a decision boundary to the positive data. And, at the time of predicting the outlierness of new unseen values, those samples are more commonly evaluated as False Negatives along all universes.

6.4 Post-classification data analysis

Even though some samples that belong to the positive class differ a lot from the regular representation of the majority of the samples of their own class, there were still a lot of samples of the positive data that were informative enough classwise, but as a whole they were not so informative, resulting in bad results at the time of the evaluation. It is important to highlight that the samples analyzed before were, at the time of the final evaluation, removed from the initial data, and thus, the final set was made up by a subset of positive samples lightly cleansed of noise.

This leads to an exhaustive analysis of each sample, and how well the positive data is related to itself and to the negative samples. To address this problem an unsupervised clustering method was carried out, a K-means [37] clustering. Developed by MacQueen (1967), K-means is a clustering that aims to divide the data into k clusters, assigning each sample to the cluster with the lowest mean. The complete algorithm is defined as follows:

(1) Specify a number k, which is the number of clusters.

(2) Initialize the centroids of the k clusters by selecting k random points of the data, and assign each to one individual cluster as their centroids.

(3) Compute the sum of the squared distance of each point to every centroid.

(4) Assign to each point the cluster with the closest centroid.

(5) Recalculate the centroid of each cluster by taking the average of every point of each cluster.

(6) Iterate from the step (3) until no new assignments are made between clusters or other conditions are filled.

This algorithm can segregate the data into *k* clusters, so the next thing would be the evaluation of the clustering. These evaluations are not like the evaluations of a classifier, which measure how well a classifier performs evaluating with the true labels of the samples. The evaluations made over a clustering, from the unsupervised perspective, are more about measuring the consistency of the clustering. The technique used in this experiment was the silhouette method. An evaluation metric which measures the similarity of a sample to his own cluster (cohesion) compared to other clusters (separation). It is defined in equation 6.1, and it provides a value between -1 and 1, where higher values indicate that the sample is in tone with his own cluster and poorly matched his other clusters. An overall high silhouette over all the samples of a cluster represents that the cluster is well made, and the data is highly related to each other and not that much with samples of other clusters.

Given a sample *x*:

$$s(x) = \frac{b(x) - a(x)}{max(b(x), a(x))} \tag{6.1}$$

a(x) represents the mean distance of the *x* sample to all other data points in the same cluster, and *b(x)* represents the mean distance of the sample *x* to all the samples that not belong to his own cluster. This evaluation metric fit well the objective of this experiment, which was about evaluating the division between positive and negative data.

So, as this was a post-classification analysis, the K-means algorithm was applied for the whole data (previously standardized), including both train and test subsets. This algorithm was applied for several *k*, but the one which returned the best overall silhouette value of all samples was *k=2*, which resulted in an overall 0.096 silhouette. The clustering was visualized using as axis the features of "CT0000LDH" and "CD0000AGE", two really salient variables for the positive class, which make the visualization easier. The final clustering can be observed at the figure 6.3.

The clustering resolution was the next one: the cluster 0 was made up by 65 (23.55% of the total) positive samples and 995 (65.37% of the total) negative samples. Clearly this sample was created around the negative class. The second cluster (1), was formed by 221 (76.45% of the total) positive samples, and 527 (34.63% of the total) negative samples. Even though the percentages of each class are well balanced, the unbalancing of the number of samples per class plays a major role in this classification, mimicking in certain ways the outcome of the OCSVM, LOF and autoencoder for the main experiments. The negative samples are more easily clustered, in part thanks to its great number of samples, and, on the other hand, it is difficult to properly segregate the positive class from the negative one due to the bad representation of itself.

The final results can be seen in the figure 6.3, where the samples belonging to cluster 0 are better matched to their own cluster than the samples of cluster 1 to their own cluster. As mentioned, cluster 1 was the one which had most of the samples of the positive class grouped in it, and it returned a negative silhouette for the majority of the samples. The interpretation of these results are directly related to how the positive and negative samples

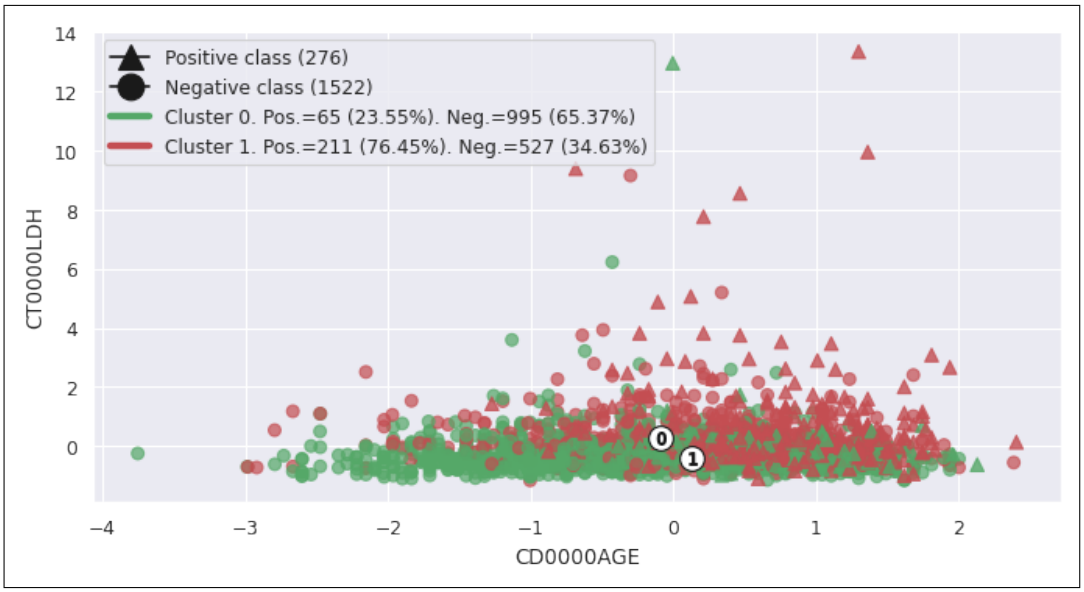


Figure 6.2: Clustering of positive (deceased) and negative (discharged) samples. The red cluster (1) includes a large number of samples of the positive class, whilst the green cluster (0) includes more samples of the negative class. The circles with the number of the cluster inside represent the centroid of each cluster, and as observed they're pretty close, thus, the clusters are not so separable.

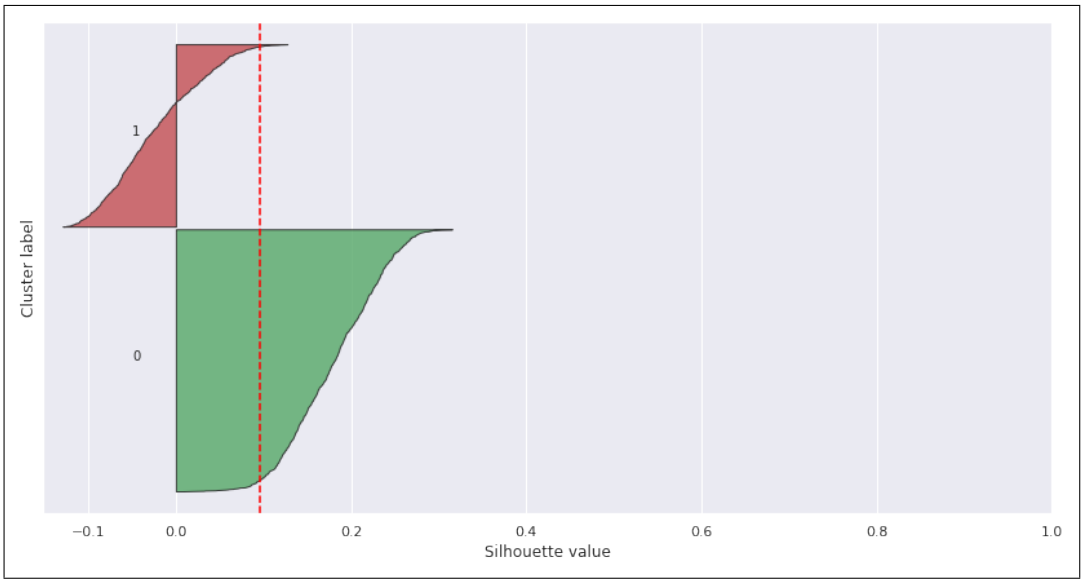


Figure 6.3: Silhouette value representation for each sample, divided in 2 clusters.

are represented, and how well is the differentiation between both of them, and it is slightly related to the method of validation selected.

CHAPTER 7

Conclusions

All the experiments concluded, it is clear that One-Class Classification relies heavily on how well the representation for the positive class is made, underperforming in situations where the difference between the positive and negative samples is not clear. A great number of deceased patients turned out to share a lot of similarities with the discharged patients, showing similar optimistic diagnostics like the discharged patients but eventually passing away, which was reflected as noise in the training data and as a prominent number of false negatives over all classifications, analyzed in chapter 7. The machine learning techniques applied in this work are not reliable when talking about the interpretability of the models, which is a key cornerstone in the machine learning application for medical fields, and, added to the bad results obtained, the interpretability is summarized to the individual analysis made over the positive outlier samples. Any other analysis made over the results and the data obtained worked with, might be explained on a really abstract level, since some more advanced conclusions should be carried out by a medical expert.

However, looking ahead, OCC techniques still have a promising future, and they have enough room to be exploited in fields such as health, industry, or even in day-to-day situations like social networks.

As mentioned, the poor representation of the data was the main problem that punished the classification. But, there is another problem that makes even the simulation of the external models complicated, and it is the lack of COVID-19 data, and the lack of standardization between datasets. Even though this is an issue that is still up, the experimental results of COVID-19 prognosis prediction models have shown a huge evolution since the start of the pandemic, resulting in better results as days go by. Hopefully these methods will eventually be implemented in ER of hospitals all over the world, lightening the decision making process and improving the resource management of hospitals.

Future work The augmentation of natural data (obtained from hospitals) or artificial data (generated by neural networks) of deceased patients in the dataset, may help breaking the imbalance situation, obtaining a more detailed representation of the data, and thus, modeling a classifier powerful enough to discriminate properly between patients with death outcome and other kind of patients, in a One-Class scenario.

Appendix

Appendix A

Feature	*Description*
CT0000ADW	Red blood cell distribution width
CT00000AP	Prothrombin time
CT000APTT	Activated Partial Thromboplastin Time
CT0000BAS	Basophil count
CT000BASP	Basophil Percentage
CT00000BT	Bilirubin count
CT000CHCM	Mean Corpuscular Hemoglobin Concentration
CT000CREA	Creatinite count
CT00000DD	D-Dimer count
CT0000EOS	Eosinophil count
CT000EOSP	Eosinophil Percentage
CT0000GGT	Gamma-Glutamyl Transferase count
CT0000GLU	Glucose count
CT0000GOT	Glutamic Oxaloacetic Transaminase test: aspartate aminotransferase count
CT0000GPT	Glutamic-Pyruvic Transaminase test: alanine transaminase count
CT0000HCM	Mean Corpuscular Hemoglobin
CT000HCTO	Hematocrit level
CT0000HEM	Haematid count
CT0000HGB	Hemoglobin level
CT0000INR	International Normalized Ratio of the prothrombin time
CT000000K	Potassium count
CT0000LDH	Lactate Dehydrogenase count
CT000LEUC	Leukocyte count
CT0000LIN	Lymphocyte count
CT000LINP	Lymphocyte Percentage
CT000MONO	Monocyte count
CT00MONOP	Monocyte Percentage
CT00000NA	Sodium count
CT0000NEU	Neutrophil count
CT000NEUP	Neutrophil Percentage
CT0000PCR	C-Reactive Protein count
CT000PLAQ	Platelet count
CT00000TP	Protombine Time

CT000000U	Urea nitrogen count
CT0000VCM	Mean Corpuscular Volume
CT0000VPM	Mean Platelet Volume
CT0000SYM	Number of Symptoms
CT0000COM	Number of Comorbidities
CD0000AGE	Age
CTHSDXXRATE	First measured Hearth Rate
CTHSDXXSAT	First measured Oxigen Saturation
CTHSDXXTEMP	First measured Temperature
CD000MSEX	Male Sex
CD000FSEX	Female Sex

Table 1: Description of features.

Feature	*Total*	*Discharged*	*Deceased*
CT0000ADW	13.02±2.08	12.88±2.0	13.78±2.33
CT00000AP	74.37±16.74	75.44±15.66	68.43±20.76
CT000APTT	32.94±7.03	32.68±6.05	34.39±10.85
CT0000BAS	0.02±0.02	0.02±0.02	0.02±0.02
CT000BASP	0.31±0.29	0.32±0.3	0.23±0.2
CT00000BT	0.57±0.45	0.55±0.35	0.69±0.79
CT000CHCM	33.62±1.42	33.68±1.39	33.25±1.51
CT000CREA	1.0±0.51	0.93±0.37	1.35±0.87
CT00000DD	2175.75±6823.36	1829.78±5813.5	4083.61±10629.91
CT0000EOS	0.05±0.21	0.06±0.23	0.02±0.04
CT000EOSP	0.7±1.57	0.79±1.67	0.23±0.59
CT0000GGT	74.26±90.8	74.63±92.41	72.24±81.47
CT0000GLU	125.47±45.97	122.24±42.36	143.33±59.25
CT0000GOT	45.02±94.96	41.02±34.85	67.09±227.22
CT0000GPT	39.05±71.5	38.11±39.64	44.25±157.11
CT0000HCM	29.66±2.27	29.62±2.24	29.84±2.47
CT000HCTO	40.53±5.27	40.64±5.13	39.91±5.93
CT0000HEM	4.61±0.66	4.64±0.64	4.49±0.73
CT0000HGB	13.64±1.92	13.7±1.88	13.34±2.14
CT0000INR	1.4±1.33	1.34±1.11	1.72±2.17
CT000000K	4.21±0.55	4.21±0.52	4.24±0.66
CT0000LDH	601.1±367.24	558.36±258.04	836.77±668.69
CT000LEUC	7.62±4.54	7.21±3.67	9.92±7.37
CT0000LIN	1.23±1.41	1.26±1.4	1.07±1.44
CT000LINP	18.19±10.44	19.21±10.31	12.59±9.32
CT000MONO	0.58±1.7	0.54±0.3	0.83±4.28
CT00MONOP	7.77±4.23	7.97±3.76	6.63±6.06
CT00000NA	136.92±4.5	136.74±3.79	137.9±7.17
CT0000NEU	5.75±3.77	5.33±3.24	8.03±5.37
CT000NEUP	73.01±12.99	71.71±12.63	80.17±12.62

CT0000PCR	101.0±100.87	89.57±91.34	164.05±125.03
CT000PLAQ	225.32±96.93	228.22±95.51	209.32±103.15
CT00000TP	15.39±13.89	14.69±10.64	19.25±24.81
CT000000U	43.17±30.72	38.59±23.03	68.41±49.78
CT0000VCM	88.23±5.77	87.95±5.49	89.78±6.94
CT0000VPM	10.35±0.98	10.31±0.96	10.58±1.05
CT0000SYM	0.06±0.26	0.06±0.27	0.03±0.18
CT0000COM	0.5±0.78	0.44±0.73	0.79±0.95
CD0000AGE	67.79±15.67	65.57±15.5	80.02±9.92
CTHSDXXRATE	79.28±14.75	78.45±13.75	83.83±18.76
CTHSDXXSAT	94.67±4.81	95.23±3.73	91.6±7.96
CTHSDXXTEMP	36.39±0.79	36.37±0.77	36.52±0.85
CD000MSEX	0.61±0.49	0.59±0.49	0.71±0.45
CD000FSEX	0.39±0.49	0.41±0.49	0.29±0.45

Table 2: Table of mean and standard deviation for every feature, divided by the target variable.

Appendix B

	Universe									
Metric	42		89		101		151		189	
Prediction	+	-	+	-	+	-	+	-	+	-
Confusion	32	23	36	19	34	21	30	25	35	20
matrix	226	79	244	61	228	77	226	79	267	38
ROC-AUC	0.436		0.470		0.442		0.411		0.465	
Sensitivity	0.581		0.654		0.618		0.545		0.636	
Specificity	0.259		0.200		0.252		0.259		0.124	
F1-score	0.204		0.214		0.214		0.192		0.196	

Table 3: Complete evaluation of the OCSVM

	Universe									
Metric	42		89		101		151		189	
Prediction	+	-	+	-	+	-	+	-	+	-
Confusion	25	30	24	31	17	38	15	40	29	26
matrix	231	74	233	72	193	112	197	108	262	43
ROC-AUC	0.665		0.683		0.716		0.718		0.700	
Sensitivity	0.454		0.436		0.309		0.272		0.527	
Specificity	0.242		0.236		0.367		0.354		0.140	
F1-score	0.160		0.153		0.128		0.112		0.167	

Table 4: Complete evaluation of the LOF.

	Universe									
Metric	42		89		101		151		189	
Prediction	+	-	+	-	+	-	+	-	+	-
Confusion	43	12	30	25	43	12	30	25	34	21
matrix	286	19	214	91	285	77	226	20	240	65

ROC-AUC	0.500	0.526	0.516	0.559	0.525
Sensitivity	0.781	0.545	0.781	0.618	0.745
Specificity	0.062	0.298	0.065	0.213	0.065
F1-score	0.223	0.200	0.224	0.206	0.215

Table 5: Complete evaluation of the autoencoder.

	Universe									
Metric	42		89		101		151		189	
Prediction	Death	Surv.	Death	Surv.	Death	Surv.	Death	Surv.	Death	Surv.
Confusion	51	4	49	6	49	6	49	6	49	6
matrix	81	224	73	232	63	242	86	219	85	220

ROC-AUC	0.922	0.916	0.915	0.895	0.886
Accuracy	0.764	0.780	0.808	0.744	0.747
Sensitivity	0.927	0.891	0.891	0.891	0.891
Specificity	0.734	0.761	0.793	0.718	0.721
F1-score	0.545	0.553	0.587	0.516	0.519

Table 6: Results of the supervised approach [1] (Armañanzas *et al.*).

Bibliography

[1] R. Armañanzas, A. Díaz, and S. Mazuelas. Derivation of a cost-sensitive COVID-19 mortality risk indicator using a multistart framework. In *Proceedings of the European Conference on Machine Learning and Knowledge Discovery in Databases*, 2021. Submitted. See pages vii, 3, 7, 23, 41, and 53.

[2] World health organization, who coronavirus (covid-19). https://worldhealthorg.shinyapps.io/covid/. Accessed: 2021-05-11. See page 1.

[3] Atieh Pourbagheri-Sigaroodi, Davood Bashash, Fatemeh Fateh, and Hassan Abolghasemi. Laboratory findings in COVID-19 diagnosis and prognosis. *Clinica Chimica Acta*, 510:475–482, November 2020. See page 2.

[4] André Filipe de Moraes Batista, João Luiz Miraglia, Thiago Henrique Rizzi Donato, and Alexandre Dias Porto Chiavegatto Filho. COVID-19 diagnosis prediction in emergency care patients: a machine learning approach. preprint, Epidemiology, April 2020. See pages 2, 3.

[5] Konstantinos Bartziokas and Konstantinos Kostikas. Lactate dehydrogenase, COVID-19 and mortality. *Medicina Clínica*, 156(1):37, 2021. Publisher: Elsevier. See page 2.

[6] David Martinus Johannes Tax. *One-class classification: Concept-learning in the absence of counter-examples*. PhD thesis, Technische Universiteit Delft, 6 2001. See pages 3, 14, 15, 20, and 24.

[7] Manuel Sánchez-Montañés, Pablo Rodríguez-Belenguer, Antonio J. Serrano-López, Emilio Soria-Olivas, and Yasser Alakhdar-Mohmara. Machine Learning for Mortality Analysis in Patients with COVID-19. *International Journal of Environmental Research and Public Health*, 17(22):8386, January 2020. Number: 22 Publisher: Multidisciplinary Digital Publishing Institute. See page 3.

[8] Stephen R. Knight, Antonia Ho, Riinu Pius, Iain Buchan, Gail Carson, Thomas M. Drake, Jake Dunning, Cameron J. Fairfield, Carrol Gamble, Christopher A. Green, Rishi Gupta, Sophie Halpin, Hayley E. Hardwick, Karl A. Holden, Peter W. Horby, Clare Jackson, Kenneth A. Mclean, Laura Merson, Jonathan S. Nguyen-Van-Tam, Lisa Norman, Mahdad Noursadeghi, Piero L. Olliaro, Mark G. Pritchard, Clark D. Russell, Catherine A. Shaw, Aziz Sheikh, Tom Solomon, Cathie Sudlow, Olivia V. Swann, Lance CW Turtle, Peter JM Openshaw, J. Kenneth Baillie, Malcolm G. Semple, Annemarie B. Docherty, and Ewen M. Harrison. Risk stratification of patients admitted to hospital with covid-19 using the ISARIC WHO clinical characterisation protocol: development and validation of the 4c mortality score. *BMJ*, 370:m3339, 2020. Publisher: British Medical Journal Publishing Group Section: Research. See pages 3, 5.

[9] Bassam Mahboub, Mohammad T. Al Bataineh, Hussam Alshraideh, Rifat Hamoudi, Laila Salameh, and Abdulrahim Shamayleh. Prediction of covid-19 hospital length of stay and risk of death using artificial intelligence-based modeling. *Frontiers in Medicine*, 8:389, 2021. See page 3.

[10] Hm hospitales, covid data save lives. https://www.hmhospitales.com/coronavirus/covid-data-save-lives. Accessed: 2021-02-15. See pages 3, 7.

[11] Li Yan, Hai-Tao Zhang, Jorge Goncalves, Yang Xiao, Maolin Wang, Yuqi Guo, Chuan Sun, Xiuchuan Tang, Liang Jing, Mingyang Zhang, Xiang Huang, Ying Xiao, Haosen Cao, Yanyan Chen, Tongxin Ren, Fang Wang, Yaru Xiao, Sufang Huang, Xi Tan, Niannian Huang, Bo Jiao, Cheng Cheng, Yong Zhang, Ailin Luo, Laurent Mombaerts, Junyang Jin, Zhiguo Cao, Shusheng Li, Hui Xu, and Ye Yuan. An interpretable mortality prediction model for COVID-19 patients. *Nature Machine Intelligence*, 2(5):283–288, May 2020. See page 5.

[12] Muhammad Attique Khan, Seifedine Kadry, Yu-Dong Zhang, Tallha Akram, Muhammad Sharif, Amjad Rehman, and Tanzila Saba. Prediction of COVID-19 - pneumonia based on selected deep features and one class kernel extreme learning machine. *Computers & Electrical Engineering*, 90:106960, 2021. See page 5.

[13] Bernhard Schölkopf, Robert Williamson, Alex Smola, John Shawe-Taylor, and John Platt. Support vector method for novelty detection. In *Proceedings of the 12th International Conference on Neural Information Processing Systems*, NIPS'99, page 582–588, Cambridge, MA, USA, 1999. MIT Press. See pages 5, 12, and 16.

[14] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *Nature*, 521(7553):436–444, 2015. Number: 7553 Publisher: Nature Publishing Group. See page 6.

[15] Jürgen Schmidhuber. Deep learning in neural networks: An overview. *Neural Networks*, 61:85–117, 2015. See page 6.

[16] Lukas Ruff, Robert Vandermeulen, Nico Goernitz, Lucas Deecke, Shoaib Ahmed Siddiqui, Alexander Binder, Emmanuel Müller, and Marius Kloft. Deep one-class classification. In *International Conference on Machine Learning*, pages 4393–4402. PMLR, 2018. ISSN: 2640-3498. See page 6.

[17] David M.J. Tax and Robert P.W. Duin. Support vector data description. *Machine Learning*, 54(1):45–66, 2021. See pages 6, 16.

[18] World health organization, classification of diseases (icd). https://www.who.int/standards/classifications/classification-of-diseases. Accessed: 2021-05-29. See page 7.

[19] Mary M. Moya and Don R. Hush. Network constraints and multi-objective optimization for one-class classification. *Neural Networks*, 9(3):463–474, 1996. See pages 9, 10.

[20] T. Minter. Single-class classification. *LARS Symposia*, 1975. See page 9.

[21] Pramuditha Perera, Poojan Oza, and Vishal M. Patel. One-class classification: A survey, 2021. See page 10.

[22] David M. J. Tax and Robert P. W. Duin. Uniform object generation for optimizing one-class classifiers. *The Journal of Machine Learning Research*, 2:155–173, 2002. See page 10.

[23] Shehroz S. Khan and Michael G. Madden. One-class classification: taxonomy of study and review of techniques. *The Knowledge Engineering Review*, 29(3):345–374, 2014. Publisher: Cambridge University Press. See pages 12, 15.

[24] Oleksiy Mazhelis. One-class classifiers: A review and analysis of suitability in the context of mobile-masquerader detection. *South African Computer Journal*, 36:29–48, 2006. See page 15.

[25] Vladimir N. Vapnik. *The Nature of Statistical Learning Theory*. Springer-Verlag, 1995. See page 16.

[26] Markus M. Breunig, Hans-Peter Kriegel, Raymond T. Ng, and Jörg Sander. Lof: Identifying density-based local outliers. *SIGMOD Rec.*, 29, 2:93–104, May 2000. See page 17.

[27] Karl Pearson. LIII. on lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 1901. See page 20.

[28] Christopher M. Bishop. *Neural Networks for Pattern Recognition*. Oxford University Press, Inc., 1995. See page 20.

[29] Nathalie Japkowicz, Catherine Myers, and Mark Gluck. A novelty detection approach to classification. In *In Proceedings of the Fourteenth Joint Conference on Artificial Intelligence*, pages 518–523, 1995. See page 20.

[30] Hasam Khalid and Simon S. Woo. OC-FakeDect: Classifying deepfakes using one-class variational autoencoder. In *2020 IEEE/CVF Conference on Computer Vision adnd Pattern Recognition Workshops (CVPRW)*, pages 2794–2803, 2020. ISSN: 2160-7516. See page 21.

[31] Richard Bellman. *Dynamic programming*. Princeton University Press, 1957. Published: Princeton, New Jersey: Princeton University Press. XXV, 342 p. (1957). See page 24.

[32] Litao Zhang, Xinsheng Yan, Qingkun Fan, Haiyan Liu, Xintian Liu, Zejin Liu, and Zhenlu Zhang. D-dimer levels on admission to predict in-hospital mortality in patients with covid-19. *Journal of thrombosis and haemostasis: JTH*, 18(6):1324–1329, 2020. See pages 24, 42.

[33] Jing Yuan, Rougrong Zou, Lijiao Zeng, Shanglong Kou, Jianfeng Lan, Xiaohe Li, Yanhua Liang, Xiaoyan Ding, Guoyu Tan, Shenghong Tang, Lei Liu, Yingxia Liu, Yanchao Pan, and Zhaoqin Wang. The correlation between viral clearance and biochemical outcomes of 94 COVID-19 infected discharged patients. *Inflammation Research: Official Journal of the European Histamine Research Society ... [et Al.]*, 69(6):599–606, 2020. See pages 24, 43.

[34] Kenneth O. Stanley and Risto Miikkulainen. Evolving neural networks through augmenting topologies. *Evolutionary Computation*, 10(2):99–127, 2002. See page 28.

[35] W. J. Youden. Index for rating diagnostic tests. *Cancer*, 3(1):32–35, 1950. See page 32.

[36] Dominic Stringer, Philip Braude, Phyo K. Myint, Louis Evans, Jemima T. Collins, Alessia Verduri, Terry J. Quinn, Arturo Vilches-Moraga, Michael J. Stechman, Lyndsay Pearce, Susan Moug, Kathryn McCarthy, Jonathan Hewitt, Ben Carter, and COPE Study Collaborators. The role of c-reactive protein as a prognostic marker in COVID-19. *International Journal of Epidemiology*, 2021. See page 43.

[37] J. MacQueen. Some methods for classification and analysis of multivariate observations. *Proceedings of the Fifth Berkeley Symposium on Mathematical Statistics and Probability, Volume 1: Statistics*, pages 281–297, 1967. Publisher: University of California Press. See page 43.