FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO

Developing a Framework to Improve Reproducibility in AI Models for the Health Domain

Pedro Luis Gomes Elias

Mestrado em Engenharia de Software

Supervisor: Ademar Aguiar

Second Supervisor: Artur Rocha

July 29, 2021

Developing a Framework to Improve Reproducibility in AI Models for the Health Domain

Pedro Luis Gomes Elias

Mestrado em Engenharia de Software

Approved in oral examination by the committee:

Chair: Prof. Nuno Honório Rodrigues Flores

External Examiner: Prof. Ângelo Manuel Rego e Silva Martins

Supervisor: Prof. Ademar Manuel Teixeira de Aguiar

July 29, 2021

Abstract

In scientific work, reproducibility increases trust in the experiment's results, by allowing it to be replicated by peers and reviewers and obtain similar results. The work in Artificial Intelligence (AI) was expected to be highly reproducible, as all the data and models needed to run an experiment are usually stored on a computer. This should allow them to be shared so peers can rerun the experiments. However, research that looked into many conference papers on Machine Learning (ML) experiments have shown that, in general, they were irreproducible.

This dissertation will approach specifically the reproducibility of AI experiments for the health domain. This work was created in association with Institute for Systems and Computer Engineering, Technology and Science (INESCTEC) to support their personal research in this area. The health care field adds an extra layer of complexity, as health data is protected under more rigid privacy constraints, thus being harder to reproduce.

This work aims at designing and creating a framework for researchers to share and catalog AI experiments in the field of healthcare, to improve reproducibility. This dissertation follows the development of the said framework, going through the investigation of the problem and available open-source tools. Passing through the life cycle of software development, research and requirements assessment, until the creation of a mockup and the implementation of an Minimum Viable Product (MVP). The framework was tested and the results were cataloged in this document, including possible improvements to provide more functionalities.

Keywords: reproducibility, framework, security

Resumo

Em pesquisas científicas a reprodutibilidade aumenta a confiança nos resultados do experimento, permitindo sua reprodução por pares e revisores. No campo da Inteligência Artificial (AI) sempre deveria ser possível reproduzir os experimentos, pois os dados e modelos necessários para executá-los são usualmente armazenados em um computador. Isto permite que estes sejam compartilhados com outros, para que possam refazê-lo de forma fiel. No entanto, pesquisas que analisaram diversos artigos de conferências sobre experimentos de Aprendizado de Máquina (ML) mostraram que estes eram em geral, irreproduzíveis.

Esta dissertação abordará especificamente a reprodutibilidade de experimentos de AI na área da saúde. Foi criado em associação com o Instituto de Engenharia de Sistemas e Computadores, Tecnologia e Ciência, para dar duporte aos seus pesquisadores que estejam investigando esta área. Este tópico adiciona uma camada extra de complexidade, pois os dados de saúde são protegidos por restrições de privacidade mais rígidas e, portanto, se tornam mais difíceis de reproduzir.

Este trabalho tem como objetivo o design e criação de um framework para que pesquisadores possam partilhar e catalogar experimentos de Inteligência Artificial no campo da saúde, facilitando assim a reprodutibilidade. Esta dissertação tratou de investivgar o problema mencionado, ferramentas de código, o ciclo de vida de desenvolvimento do framework, da pesquisa, elaboração de requisitos, criação de um protótipo, até a implementação de uma versão funcional do framework. O framework foi testado, e os resultados foram apresentados neste documento, incluindo possíveis melhorias para oferecer mais funcionalidades.

Keywords: reprodutibilidade, framework, segurança

Acknowledgements

I would like to start by expressing my appreciation to my supervisors, Ademar Aguiar and Artur Rocha, for their support and knowledge provided through all the stages of this work.

I also wish to extend my gratitude to all of this master degree's professors, who guided me through new knowledge.

Last, but not least, I wish to thank my parents and my sister for their love and unconditional support whenever I needed it. Additionally, I wish to express my gratitude to my wife, Sílvia, for accepting to embark with me on this journey and keep pushing me forward. And to all my friends and family, my sincere thank you.

Pedro Elias

"Knowing how to think empowers you
far beyond those who know only what to think."

Neil deGrasse Tyson

Contents

List of Figures

List of Tables

Abbreviations

AI	Artificial Intelligence
AIRR	Adaptive Immune Receptors and Repertoires
API	Application Programming Interfaces
GDPR	General Data Protection Regulation
HIPAA	Health Insurance Portability and Accountability Act
HIT	Health Information Technology
INESC-TEC	Institute for Systems and Computer Engineering, Technology and Science
JSON	JavaScript Object Notation
JWT	JSON Web Tokent
LGPD	Lei Geral de Proteção de Dados
ML	Machine Learning
MVP	Minimum Viable Product
SVM	Support vector-machines
SWEBOK	Software Engineering Body of Knowledge

Chapter 1

Introduction

This Master's dissertation in Software Engineering is being developed in association with Institute for Systems and Computer Engineering, Technology and Science (INESCTEC).

This chapter aims to provide an introduction to this work. Starting by section 1.1 where the context of this dissertation is presented. This is followed by section 1.2 with the motivation behind its creation, including a brief description of the problem tackled. Next is presented the framework's objectives in section 1.3. Finally, the section 1.5 disclosure its expected contributions.

1.1 Context

Health Information Technology is an integral part of a modern Health Care System. The industry has followed the global tendency of digitization adopting methods to store, recover, share, analyze and use health care information [19]. Thus making available a large, and constantly increasing, amount of data. [30]

Gathering and analyzing data is a core element of the Scientific Method and an important part of the research. With large datasets, Artificial Intelligence techniques became an additional tool for extracting knowledge from it, allowing the recognition of patterns and unusual events [38].

As people build trust in AI to make complex decisions, they can start to be applied to critical fields. such as healthcare, to help to solve problems like improving disease diagnosis, predict disease outbreaks, personalize treatments, analyze behavior modification [32] and support medicines development.

The majority of experiments using ML models are very hard to reproduce, because of missing documentation of the experiment, such as environment, model version, hyperparameters, dataset, data format, description of the features [24]. This is particularly complex with health-related data due to the privacy of patients' records that do not allow the creation of databases to train models, and so require techniques to mitigate these concerns [30].

1.2 Motivation

Reproducibility is an integral part of scientific research [24]. A reproducible AI experiment allows other researchers or teams to verify the achieved results. It also enables the improvement of models in order to get better results.

Increasing reproducibility of AI experiments, specifically in the field of health, can help us gain trust in those models' decisions. An experiment that can't be reproduced can't be refuted, and so, its results become dubious [24]. At such a high stakes field, validity is a must.

1.3 Objectives

The objective of this dissertation is to conceptualize and develop a web framework to document, benchmark, and compare ML models in the field of health research.

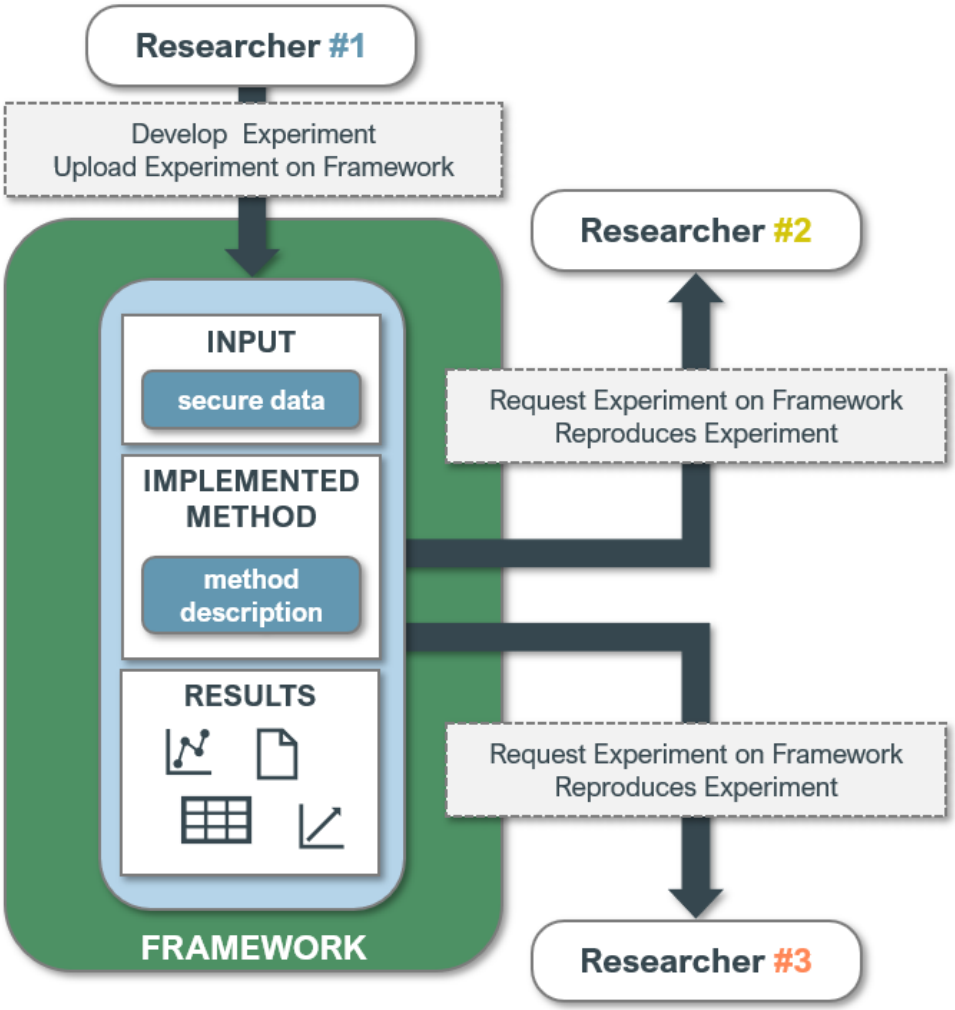

Figure 1.1: Availability of experiments at framework.

The proposed framework should allow researchers to share their data, models, and results within this scientific research field, illustrated at figure 1.1. Other researchers should be able to access such data, models, and results on this online platform and be able to reproduce those experiments using similar processing. They should also be able to interchange the data or model to run a new experiment and produce similar results, exemplified by figure 1.2 and figure 1.3.

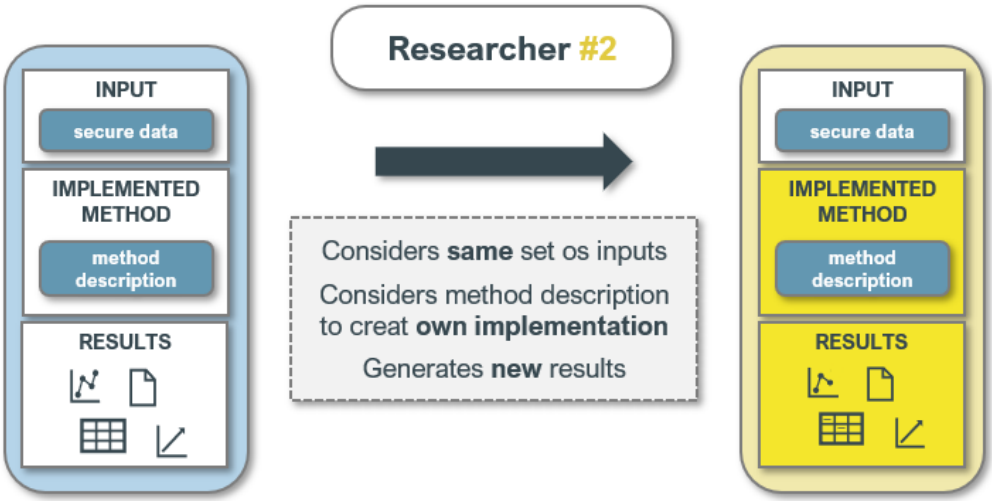

Figure 1.2: Example of one attempt to reproduce experiment by changing implementation method and utilizing same parameters.

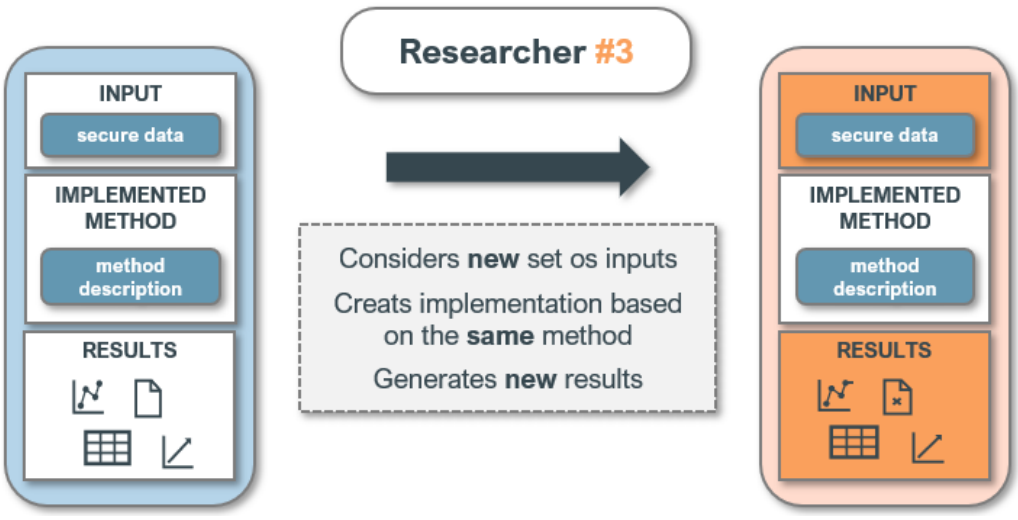

Figure 1.3: Example of one attempt to reproduce experiment by reproducing implementation method and utilizing new parameters.

After the conceptualization of the framework, a prototype will be tested in conjunction with a research team that is focused on ML for health. The result should clarify whether or not the framework made the documentation phase easier and supported, indicated in figure 1.4.

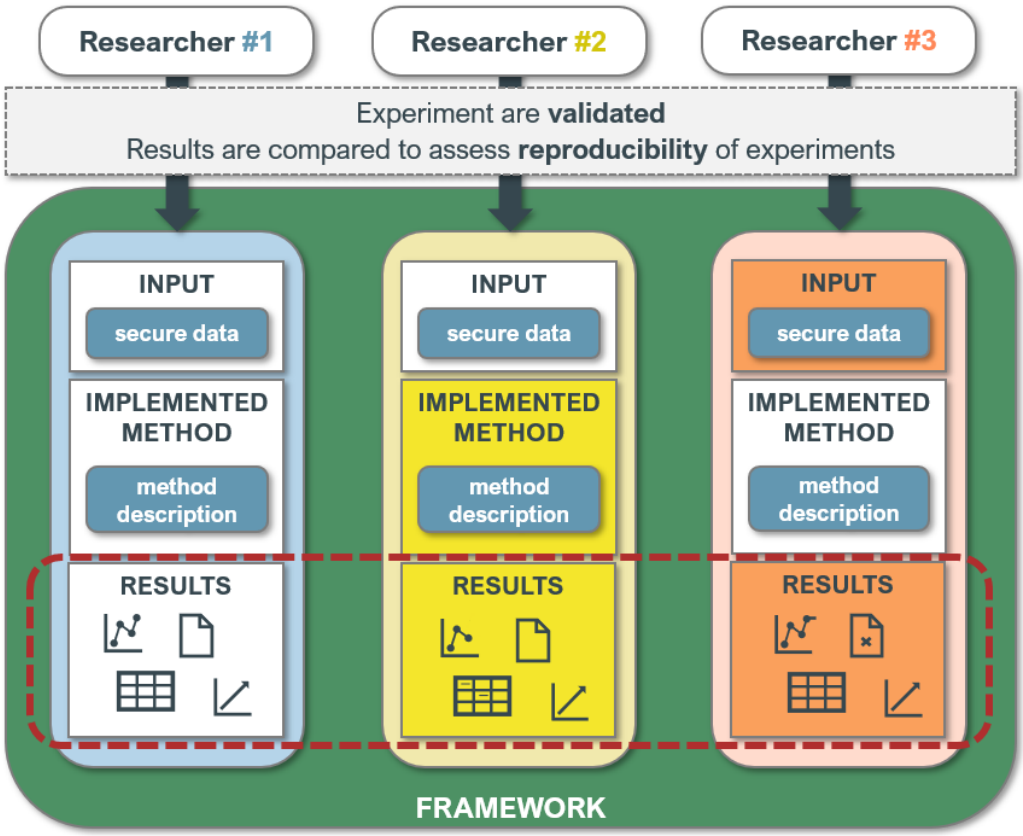

Figure 1.4: Validation of results and assessment of reproducibility.

1.4 Contributions

This work's desired contribution would be to facilitate the documentation and support of AI experiments in the research field of health, in order to make it reproducible by other researchers. The researchers should be able to incorporate the Framework during the training and testing of ML models, and it will be responsible to register the data and parameters used, as well as the results produced.

In addition to that, the framework should provide visibility to the advances made on the various stages of the training process. This could be made by providing a way to share not only the results but the full experiments. This should allow a secure diffusion of up-to-date ML models to be applied in Health Care.

1.5 Document Structure

This document is structured with multiple chapters, starting by chapter 1 (p. 1) the introductory chapter that presented the context of the upcoming dissertation, its motivation, objectives, and proposed contributions.

These are followed by chapter 2 (p. 7) that provides the State of the Art (SOTA), which is required to the fully comprehension of this work. It goes through the background for the work, presenting the Machine Learning methodology and terminology to understand Software Frameworks. After this section, we present the privacy constraints that are current in place and how health data presents itself as a special case of secure data for privacy. This section also describes the reproducibility in AI and the reproducibility guidelines for scientific research and examples of how could be implemented. Finishing this chapter, are approaches to achieve reproducibility and methods for developing frameworks.

Next, is chapter 3 (p. 17) that goes through the software requirements to the design of the software in accordance with Software Engineering good practices. Starting with requirements elicitation and follow by their specifications, used to develop an MVP. Finalizing, are the requirements validation process, that include a mockup and its validation.

Following up is chapter 4 (p. 23) with the development of the framework. Starts displaying some tools that were used to support this development, and followed by its design and implementation. Which is succeed by framework validation and examples.

And at the end is chapter 5 (p. 35) presenting the conclusion of the work and future works that could derive from the Framework developed in this dissertation.

Additionally, it is presented the appendix A (p. 37) with the source code of the tests used for the training of the Framework.

Chapter 2

State of the art

The State of the Art chapter provides some necessary information to understand the work described in this document. Including, but not limited to: definitions, methods, constraints, guidelines, examples, and terminologies that will appear in this dissertation.

Section 2.1 describes the background to follow this document, including Machine Learning methodology, presents terminologies to understand software frameworks, the privacy constraints that are currently in place, and how health data presents itself as a special case of secure data for privacy.

Follow up section 2.2 dives on reproducibility in AI and the reproducibility guidelines for scientific research and examples of implemented approaches to achieve Reproducibility

Finally, section 2.3 presents one of the knows methods for developing frameworks.

2.1 Background

This section provides some information to set up the background knowledge required for the full comprehension of the document. Though the information presented is already part of the body of knowledge of Software Engineers, it might be useful for less specialized readers. Therefore, in the spirit of scientific research and knowledge diffusion, such sections should always be included in works that will be publicly available and can be read and used by people of diverse backgrounds. Furthermore, the purpose of the Framework is to support researchers in the field of Health, that might not have Software Engineering knowledge.

2.1.1 Machine Learning

Machine Learning is a process in which a computer algorithm learns through previous experience and/or data fed as an example, without being explicitly programmed. It is a branch of AI, as seen in figure 2.1 [15].

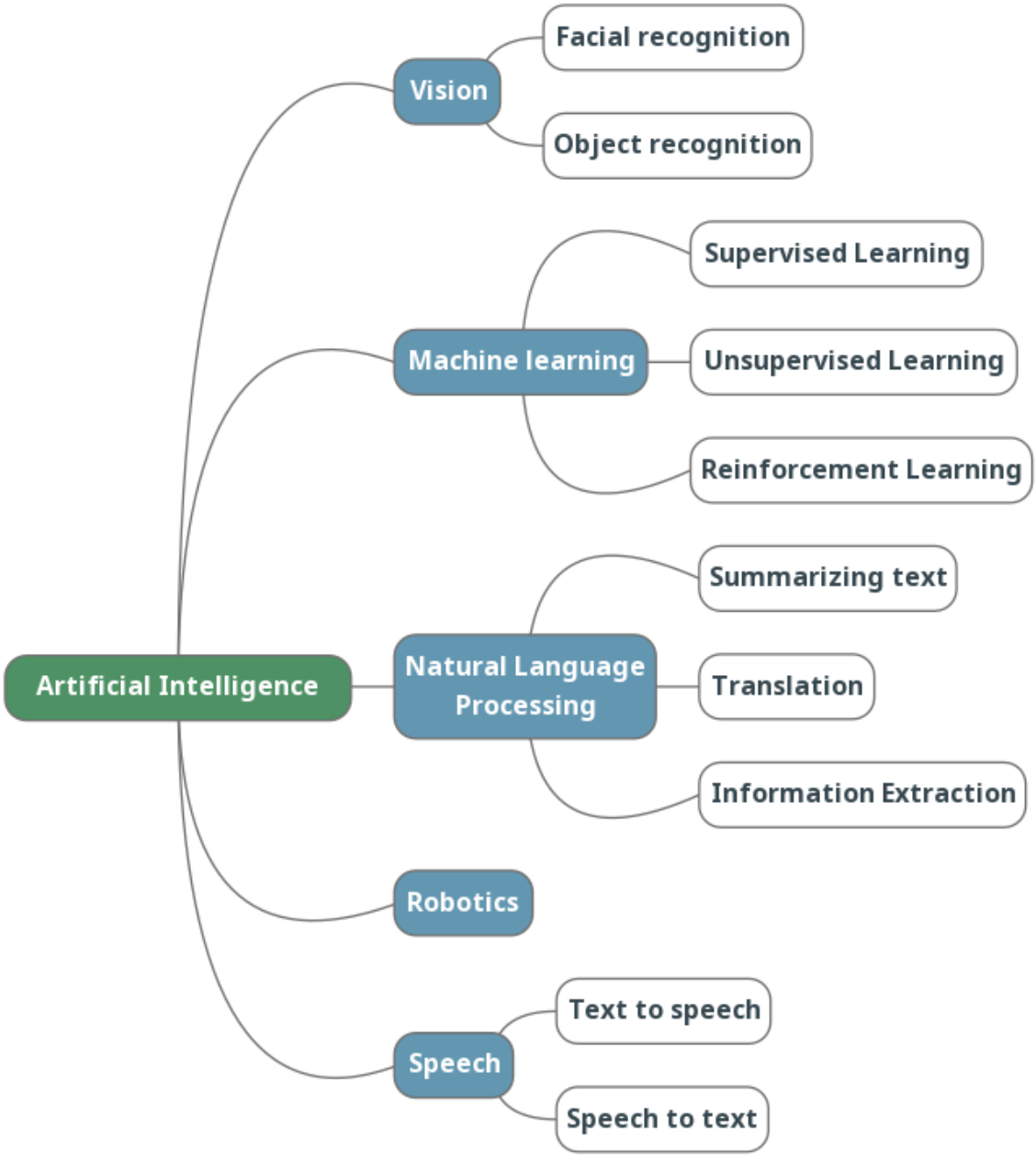

Figure 2.1: Artificial intelligence fields.

It is commonly divided into three categories, according to the nature of the inputs or feedback provided to the model:

- **Supervised Learning** - These kinds of techniques are applied so a model can learn to identify an input based on previous experiences where the labeled data were fed by the supervisor [15]. It tries to model the correlations between the input and the label based on the data provided during training.

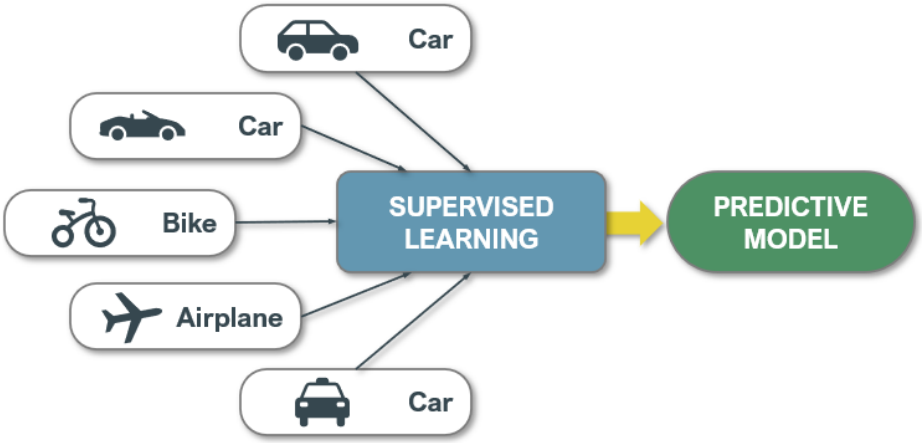

Figure 2.2: Supervised Learning.

Figure 2.3: Supervised Learning Result Acquisition.

- **Unsupervised Learning** - Unlike the previous case, on unsupervised learning, the model is not fed the data previously labeled. It is fed with different inputs and learns, without the supervisor's intervention, to find characteristics common to part of the data [15]. A good example is the use of unsupervised learning in the creation of clusters. The method receives the inputs and separates them in accordance with the similarities. There is not an explicit result perceived as a goal. But rather the segregation of data.

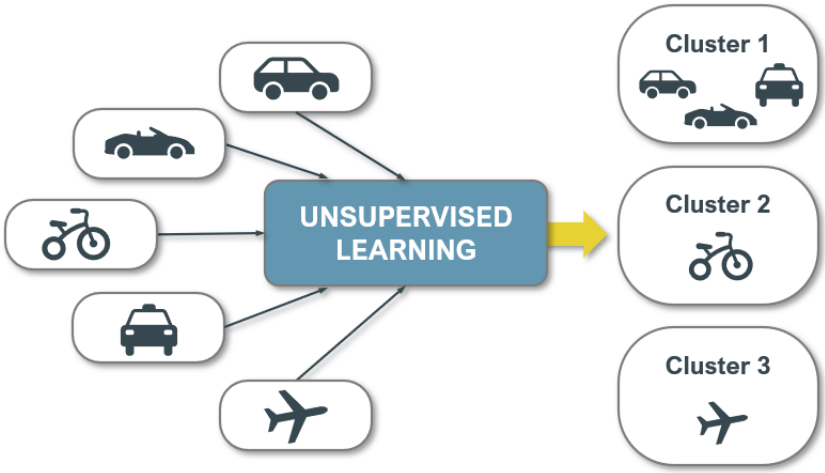

Figure 2.4: Unsupervised Learning.

- **Reinforcement Learning** - This method aims to teach an agent how to take actions in an environment in order to maximize a cumulative reward. This means that the environment

and the reward function need to be modeled so the agent can experiment with different actions given similar states, and learn what behaviors are the most rewarding ones.

In terms of outputs, it can also be divided in the following categories:

- **Regression** - The goal is to find the relationship between the input and output, and predict from a new input a continuous quantity.
- **Classification** - Based on a classified dataset, it attempts to predict the class of new input.
- **Clustering** - The data is divided into groups, based on the similarities of the data points that are in the same group, and dissimilarities of the data points that lie in different groups.

Different models have been developed to predict or explore data. Below the definition of some types of models can be seen:

- **Linear regression** is able to predict a target, by fitting a line that has the minimum sum of squared errors on the two-dimensional data [28].
- **Artificial neural networks** are networks made of artificial neurons, which are functions that receives one or multiple weighted inputs, sums them, and pass through an activation function to produce an output [27].
- **Decision trees** builds a set of decision rules based on the data features in order to predict a target value [27].
- **Bayesian networks** are a graphical model that identifies the relationship between variables and calculate the probabilities of one causing another. It uses Bayes inference to calculate those conditional probabilities [27].
- **Support vector-machines** constructs one or multiple hyperplanes that separates the data into two categories. It then can be used to classify new data by looking at where it lies on the space [27].
- **K-nearest neighbours** classifies new data based on the class of the nearest points [27].

2.1.2 Software Frameworks

Frameworks are by definition solutions made by parts of or all of software designed to build applications. It is designed by groups of abstracts classes and individualized by the way instances of their subclasses collaborate within each other [35]. The methods defined by the user to adapt it to its needs will often be acquired by the framework itself, rather they by the user's application code [26].

The development of a framework becomes advantageous when many applications will be developed in a similar structure for a similar goal [35]. They can also facilitate the usage by giving

a user enough tools to create multiple types of functional compositions without knowledge of the software and even without previous programming skills [35].

To understand the development of a framework, some terminology should be well defined, such as:

- **Object** - Is a runtime entity that supplies state and operations to consult and alter said state, representing a phenomenon from a domain. It is not immutable and should be able to change from its creation to the point of its possible elimination [34].
- **Class** - Represents a group of objects. It can be either *abstract* or *concrete* [26]. And should also be classified by its *class type*, which is determined by the behavior of the objects that make the said class. It can be defined for example by the rules of transitions between states or by the operations that trigger them [34].
- **Abstract Class** - Does not define instances variables, but rather methods of implementation for its Subclasses [26].
- **Concrete Class** - Gives a definition for the represented data. And some will include multiple Subclasses that differ in representation [26].
- **Subclass** - Comprises smaller sets of related operations withing a class (that by the subclass perspective can be refereed as its *Superclass*) [34].

The way a framework can be designed by the interaction of these objects will be covered in section 2.3.

2.1.3 Privacy Constraints

Multiple countries have established laws to regulate the boundaries and conditions for personal data collection, storage, and treatment. Some examples of current efforts towards personal data protection are the GDPR by the European Union and the LGPD from Brazil.

The European Strategy for Artificial Intelligence defined through the European Commission multiple orientations to the use of AI in a beneficial way for individuals and for the society as a whole [4] revolving around three pillars:

1. Boost investments in the field [37].
2. Prepare the community for socioeconomic changes resulted from AI dissemination [37].
3. Guarantee ethical and legal framework [37].

As part of these Guidelines, item 3 aims that, between other things, the technologies should assure that the fundamental rights to privacy and dignity of the individual are respected [4].

2.1.3.1 Health Data and Privacy

Health Data is information of diagnosis, exams results, surgery records, previous treatments, mental and physical evaluations [1], and other relevant records of an individual health through life.

Different countries deal with the privacy of health care data differently. It is broadly accepted that health-related data are a sensitive category and should be regulated over special conditions. This is applicable for both the GDPR [3] and LGPD [5], that include sections with specifications and exceptions for health-related data. Other countries have created new laws specifically targeting the protection of individuals' medical records and other personal health information. Without it being a portion of a wider privacy protection regulation. This is the case of the United States of America that created the HIPAA [2]. The regulations should be taken into consideration by the researchers in order to properly comply with the rules of the countries' repositories.

According to the GDPR Health-related data is considered as a special category that has to allow access to information in a way that both protects patient's privacy while also allowing data to be shared to the benefit of the individual and the community for Scientific Research [3].

In order to maintain the privacy of the individual whose data is addressed a number, symbol, or particular assignment shall be used to uniquely identify the natural person. This data could include all health data pertinent to its condition. Including past, present or future health status [1]. And by doing so, allowing researchers to access data without violating individual privacy and the trust placed upon the health care community.

2.2 Reproducibility in AI

As in any empirical research, reproducibility is a fundamental part of AI experiments. In the case of Machine learning, most researches depend only on data and models that are available on a computer to train and verify the results obtained [17], which should facilitate reproducing the experiments.

The term *reproducibility* may create some confusion with other terms like repeatability and replicability [23]. In [24] reproducibility is defined as "the ability of an independent research team to produce the same results using the same AI method based on the documentation made by the original research team" and three degrees of reproducibility in terms of the result are presented:

1. **Experiment reproducible** - when the results can be reproduced by executing the same implementation of the AI method using the same data.

2. **Data reproducible** - when the results can be reproduced by executing a different implementation of the same AI method using the same data.

3. **Method reproducible** - when the results can be reproduced by executing a different implementation of the same AI method using different data.

This definition illustrates also the amount of documentation necessary for each of the cases. For the results to be *Experiment reproducible*, it is enough to have the data and executable ready

to run, and minimum documentation is necessary to show how to run. To be *Data reproducible*, the AI method need to be documented in order for other researchers to be able to implement their version. Lastly, to be *Method reproducible*, information on how the data was collected and processed needs to be available. The best scenario possible in terms of reproducibility would be to have the executable and data files, and the documentation needed to collect new data and implement the AI method.

Based on that definition, Gundersen and Kjensmo (2018) [24] showed that most research papers in the field of AI are irreproducible, as they document only part of the variables needed to reach any of the degrees of reproducibility.

2.2.1 Reproducibility Guidelines

This section presents some guidelines that researchers have found to improve the reproducibility of computational research. Those guidelines include suggestions to the scientific community in general, to researchers in the field of health, and data providers in order to take maximum advantage of the voluminous amount of data that is produced and stored. They also contribute to increasing the general trust in Artificial Intelligence to make critical decisions.

- **Shared research resources** - If the privacy concerns are taken into account, the creation of shared research resources allows medical data to be anonymously published, and contributes to creating large datasets available to multiple research teams [30].

- **Keep track of every result** - Usually when researchers are training the model, multiple parameters can be used, which can generate different results. When this happens, the researchers need to keep track of how each result was produced by saving the set of parameters used [30].

- **Register data processing steps** - When processing raw data, all the scripts that manipulate it should be recorded. This allows the same process to be run by other researchers, with the same or even new data [39].

- **Store the version of programs** - Archive every previous version of the project and any external program used in the exact version that was used. Maintain a recording of every version of the working model, to not only use as a backup but also to allow responding to reviewers and to provide proper support despite current alterations [39]. The same should be made for external programs that are used in your project as new versions might be developed and generate slightly different results, or even be incompatible. Also, old versions might no longer be available for acquisition [36].

- **Register intermediary results** - It is common to process the output of models in order to generate tables or graphs, so it is necessary to record both the processing and the data that was generated [36].

- **Record the seed used when the model needs randomness** - For models that depend on the generation of random numbers to run, it is important to record the seed that was used. Using the same seed to reproduce the experiment guarantees that the exact same result will be produced, instead of a slightly different result [36].

- **Make scripts, runs, and results publicly available** - Submit at least the main data and source code online. Preferentially, do so for your complete work, including input data, program versions, source code, hyperparameters, and achieved results [36].

- **Develop data standards** - The use of data standards allows other researchers to better understand the experiment conceptually, as they can be familiarized with that format. It also gives the advantage of being able to test the experiment with new data [30].

This set of guidelines could be followed individually by researchers to achieve the reproducibility of their AI experiments. However, this would require the use of several tools, uploading each of the artifacts, and keeping track of the versions of each of the dependencies. The lack of specific tools that can automate those tasks shows that there is great room for improvement.

2.2.2 Approaches for Reproducibility

There have been many approaches on how to make computational researches reproducible. Some of them are no longer applicable in recent years as they rely on obsolete technologies. However, they are worth mentioning as the concepts can be recreated in modern ways.

To document the reported results Claerbout and Karrenbach (1992) [21] proposed sharing the whole experiment on a CD-ROM, making it re-runnable by anyone that wishes to read the research and look at the experiments. This approach is outdated, as CD-ROM is hardly used, and depends on physical sharing of the media.

Buckheit and Donoho (1998) [18] developed a Matlab package called Wavelab that could reproduce their figures from their papers that contained results. However, this is a package to be used in specific fields, and it does not apply to Machine learning research.

Gundersen and Kjensmo (2018) [24] proposed sharing access to a virtual machine, that contains everything needed to reproduce the results: the data, runnable, documentation, and source code. This is a more generic approach, that can fit any supervised or unsupervised learning experiments. An obstacle in this approach is the high cost of keeping a running virtual machine for that purpose.

Choi et al. (2021) [20] presents a solution in the field of Environmental modeling, by using containerization technology, notebooks, and Application Programming Interfaces to encapsulate and document the experiments.

2.3 Developing Frameworks

This dissertation will use as a reference the path to designing a framework from Evolving Frameworks: A Pattern Language for Developing Object-Oriented Frameworks (Roberts, 1996) [35], by which an effective framework should be adaptable, reusable, and easily configurable.

The Evolving Framework design method follows the steps described in figure 2.5. These are not linear, but rather overlap and circle back to, as the framework should be evolving constantly and being adapted as the designer goes through it.

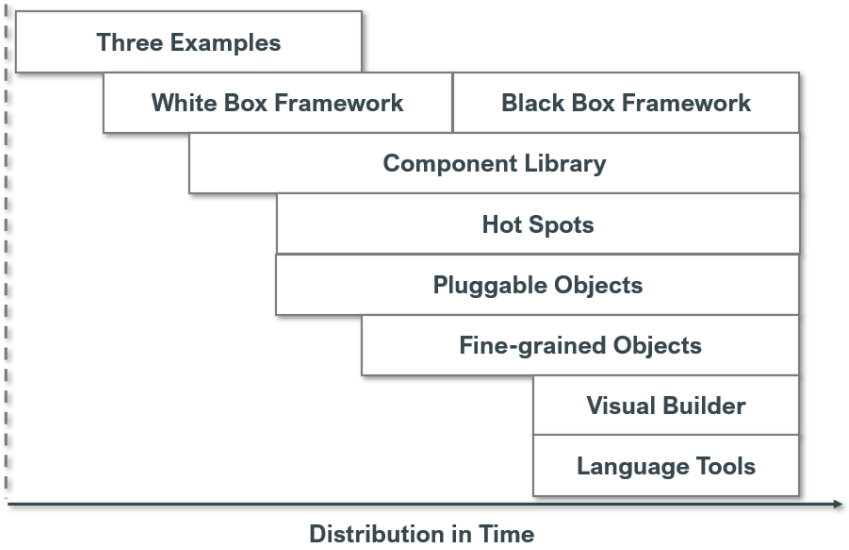

Figure 2.5: Relationship between the patterns in the pattern language, extracted from Evolving frameworks: A pattern language for developing object-oriented frameworks. [35].

- **Three Examples** - Consists in selecting multiple (at least three) applications where your framework will be used to resolve a problem. By building these applications you will be able to identify common abstractions to start to design your framework [35].
- **White Box Framework** - A construction method based on inheritance [34]. By which new classes are created by inheriting the behavior from an existing class in the individual applications selected on *Three Examples* [35]. Framework designed solely on white-box modeling requires that the user understands their implementation, as to properly change and adapt the application to its use [26].
- **Component Library** - Composed by similar objects that will be common across the majority of the solutions. To start the design all the concrete sub-classes accumulated from the applications should be added to the component library. If after several uses they are not used consistently, they should be discarded. Otherwise will be maintained in the code and create the component library. [35].
- **Hot Spots** - Parts of the code that show to be recurrently written [35].

- **Pluggable Objects** - Design to adapt sub-classes configurable with parameters such as deliverable messages, indexes to access, boxes, or any other that discriminates one subclass from the others. This avoids the creation of unnecessary sub-classes where only one method is overruled [35].
- **Fine-grained Objects** - When identified classes that show multiple behaviors with small variations between them, these can be replaced with several smaller classes encapsulating each one of that behavior. The original class can be replaced with relationships that recreates the targeted behavior [35].
- **Black Box Framework** - Construction method based object composition [34]. By which components are plugged together to execute a task, not being relevant how they perform individually [35]. Frameworks that include black box design allow its use by only understanding the external interface of components [26].
- **Visual Builder** - Creation of graphic aids to make the framework more user-friendly. It should represent the interaction between the objects of the application [35].
- **Language Tools** - Helps understand and debug the framework. Specialized tools to evaluate the compositions are required to aid the user [35].

Using concrete examples (as per *three examples*) as a form to generalize abstraction is the core value that the white box framework brings to the framework design. White box framework should be used at the start of the design to not make it too rigid and not reusable. While black-box-based frameworks rely on the interfacing between their parts. It is easier for the user as no knowledge of implementation of the parts is required, only the understanding of the external interfaces. As the framework evolves, it should do so towards a black box relationship [26] with a user-friendly interface.

2.4 Summary

So far the research shows that many experiments in the field of ML are not reproducible [24]. Through the years efforts have been made to try to improve said reproducibility [30]. Though the solutions available are too general in terms of improving the reproducibility of AI models as seen in [21], [18], [24] and [20], their base concepts can be integrated to tackle a specific problem in the field of health.

With the information described in this chapter, this work aims to develop a framework that can assist researchers of health to make more reproducible researches. The framework will be developed utilizing the guidelines of [35] to present a solution that is user-friendly and can be refined with use. This should make the work of researchers easier as it will allow to automatically record all variables and code necessary to run the experiment, as well as the documentation that can clarify how the experiment was conducted. Also, will focus on the availability of the experiments by allowing them to be shared within the scientific community.

Chapter 3

Problem and Perspective of Solution

This chapter will dive into the design of the solution before the development stage. The intent was to follow as much as possible the good practices for Software Engineers on the subject of software development, and so SWEBOK [16] was used as reference.

The next step of the process is to derive from our problem the requirement elicitation, requirement analysis, software requirements specification, and prototyping. The prototype was followed by validation with the supervisors.

3.1 Problem

The software requirements should express the need of a product that resolve some real-world problem [16] that affect your target user. In this work the motivation was to create a framework to try to surpass the obstacles that affect reproducibility in ML experiments to facilitate the work of INESCTEC's researchers.

Because ML models have high variability in terms of inputs, parameters, and the environment they are trained to perform with optimal results. Because of this, it is difficult to reproduce experiments without having the exact same conditions that those utilized the first time the experiment was conducted [24].

Without being able to reproduce experiments, other researchers are not able to verify the published results. They are also not able to interchange their parts, such as data, model, or parameters, to achieve better results. This should be a major concern on scientific method [24].

For health related experiments the access to the same conditions become more complicated due to the privacy restraints, compromising the diffusion of models and sharing of knowledge. Additionally, these aspects are highly challenging due to the diversity and voluminous amount of data.

3.2 Requirements Elicitation

The project scope should be designed based on the stakeholders' needs before the development of the Framework starts [16]. Thus the elicitation of requirements was the first step of its design, which is concerned with the origins of the requirements, how and from where they can be collected.

This process started with the contact with INESCTEC to assess their most important requirements to be satisfied. As stated in the previous section, their intent was to possess a platform that would allow scientific researches on the field of health to be shared between peers for validation, knowledge diffusion, and reproducibility purposes. The next step was to search and study available tools that work similarly to their intended requirements.

As part of the Requirement Elicitation some tools that have purposes similar to INESCTEC's objective, or that include functionalities that are required by them, were analyzed. This sub-section will describe some of these Open Source applications and collaborative environments that could have been used for the construction of the Framework; or in an extreme case, could have been found to serve its purpose perfectly. This will also explain which requirements were extracted from these tools, in the cases where they perform appropriately, and which derived from places where those lack in performance for the scope of this work.

3.2.1 Jupyter Notebook

Jupyter is a web application for interactive computing that allows users to create and share documents [8]. It was developed under the Jupyter Project, with the work of multiple contributors, for uses such as data cleaning and transformation, statistical modeling, numerical simulation, ML, data visualization, and others. The *Notebooks* (representation of the content) can contain live code, equations, visualizations, and narrative text in multiple programming languages, and can be shared with others [8].

Jupyter is available for online use or installation at their website [8].

This tool is a good way to allow users to publicly share their codes. It has a good level of flexibility for being web-based but also allowing you to download it to use locally. The capability that allows the user to visualize the code, textual documentation, and rendering of plots in the same place, creates a user-friendly environment and facilitates the revision of the code. However, one of the disadvantages related to use this tool under the scope of this work is that it demands the libraries to be installed, thus requiring additional set-up. It also does not store the multiple runs of the experiment, causing that the progress of the work is lost and only the final results, usually already optimized, are shared between peers.

3.2.2 Google Colaboratory

Colaboratory is an interactive environment that allows users to write and execute Python in a web browser, by executing them on Google's cloud servers and also making use of pre-installed

libraries [14]. It was developed by Google as an easy-to-use and light way to write and execute code. The web-based tool doesn't use the computer processing capacity for being cloud-based and is also integrated with Google Drive, thus reducing risks of losing your work and allowing it to be a more collaborative tool where you can share your code, including simultaneous view and edit [14].

Google Colaboratory is available for use online at their website [14].

The tool was considered to be integrated at the Framework to allow users to publicly share their code if desired, while also reducing the need for software setup and GPU usage from peers to review said work. However, this should be considered for future work and was not implemented in this version of the Framework.

3.2.3 HydroShare

HydroShare is an Open Source collaboration environment, that enables users to share data, models, and codes with peers (in private groups or publicly) [7]. It was developed by the Consortium of Universities for the Advancement of Hydrologic Science, Inc. and designed to attend the field of hydrological research [7].

HydroShare code source is available at GitHub [7].

Although the objective of this tool is rather similar, the difference in the focus of the researches made the tool too specialized, so it was hard to adjust to the health domain needs. It was, however, a good model to be studied and used for research.

3.2.4 Sciunit

Sciunit is a system developed for containerizing, sharing, and tracking scientific applications in the field of scientific researches, in order to allow better reproducibility [12]. It was developed by researchers from DePaul University, with the work of multiple contributors, to create a type of research object that encapsulates the workflows of a researchers' work. The reusable research objects that containerize and stores applications to facilitate sharing and collaboration by easing the tasks of environment setup, and execution steps of the shared work [12]. Sciunit also integrates with HydroShare, so it can publish the generated containers directly into HydroShare. By doing that, other users are able to download these containers and run the experiments by themselves.

Sciunit is available for installation at their website [12].

This tool was considered a good model to be studied and used for the research of ways to improve reproducibility for scientific research works.

3.3 Requirement Specifications

The software requirements establishes the agreement on what the Framework should do, as well as what it is expected not to do [16]. From the product analysis the description of the requirements could be stated as:

R.1 - The framework should be easily accessed within the organization.

- **R.1.1** - The framework should present a user interface that allows users to visualize the artifacts that were uploaded.
- **R.1.2** - The framework should be accessible by other applications, in order to upload artifacts programmatically.
- **R.1.3** - The users should be able to share their artifacts with other users, or even make them public, so the community may collaborate and give feedback.
- **R.1.4** - The user should be able to publish a specific version of an experiment and generate a unique identifier and link, so it can be properly cited.

R.2 - The framework should provide an easy way to reproduce the experiment.

- **R.2.1** - The framework should be able to store all the artifacts needed to run the experiment in a new environment and compare results: models, parameters, metrics, and dependencies.
- **R.2.2** - The framework should encapsulate the experiment executable and its dependencies, so other researchers can run the experiment without doing any software setup.
- **R.2.3** - The framework should provide a way to visualize the source code of the experiment.
- **R.2.4** - The framework should keep track of all the runs for each experiment, by registering the parameters used for each individual run.

R.3 - The framework should identify the user, and after that, grant access to the user's resources.

- **R.3.1** - The framework should not deliver any information to unauthorized entities.
- **R.3.2** - As the framework will deal with Health information security, the user should be able to control the access over its artifacts, by telling which are public or private.
- **R.3.3** - The framework should be integrated with INESCTEC's authentication server in order to provide Single-Sign-On access to the user without the need for new credentials.
- **R.3.4** - The framework should be able to discriminate if the requests are done by a user or an application.

R.4 - All the data must be stored in a secure server.

3.4 Minimum Viable Product

The Minimum Viable Product is a development technique that allows the developer to shorten the life cycle of a product while assessing the features needed for its first introduction to the users. The release of an MVP should allow the user to start using the software and provide feedback to improve the final product [9].

Its use is also often associated with a way to save resources: cost and time. Under the scope of this work, the MVP is used to provide a first version of the product [31]. The addition of the features not included in it will be highlighted under the *future works* segment at section 5.2.

For the proposed Framework's MVP the following requirements were considered: **R.1.1**, **R.1.2**, **R.2.1**, **R.2.4**, **R.3.1**, **R.3.3**, **R.3.4** and **R.4**.

With these implemented, it is expected that the user is able to access a web-based interface (**R.1.1** and **R.1.2**). Where they can log into the framework (**R.2.1**), considering that it already was already registered by INESCTEC and with access to the organization (**R.3.3**). Once authorized, they should be able to generate an application key, as well as to use them to log their experiments and visualize all the artifacts that were logged in the interface (**R.2.4**, **R.3.1**, **R.3.4** and **R.4**).

3.5 Requirements Validation

Requirements should be validated to ensure they were correctly understood [16]. The documentation of the requirements, as described in section 3.3, should describe the software as the users expect it to be. The validation should be used not only as a way for the developer to validate their interpretation of the framework requirements but also as a process that could expose any problems before the development starts [16].

3.5.1 Non-functional Prototype

The creation of a prototype was chosen as the validation method for the requirements. By creating a scenario that gives the user the context of the framework so they can experiment, a better understanding of the final goal can be provided. And with their feedback, the requirements can be changed and fine-tuned, so the framework may achieve a better result [16].

In the scope of this work is used a non-functional prototype, a low-fidelity model also known as mockup [16]. figures 3.1 and 3.2 below illustrates the initial version of the framework constructed, as how it was conceptualized.

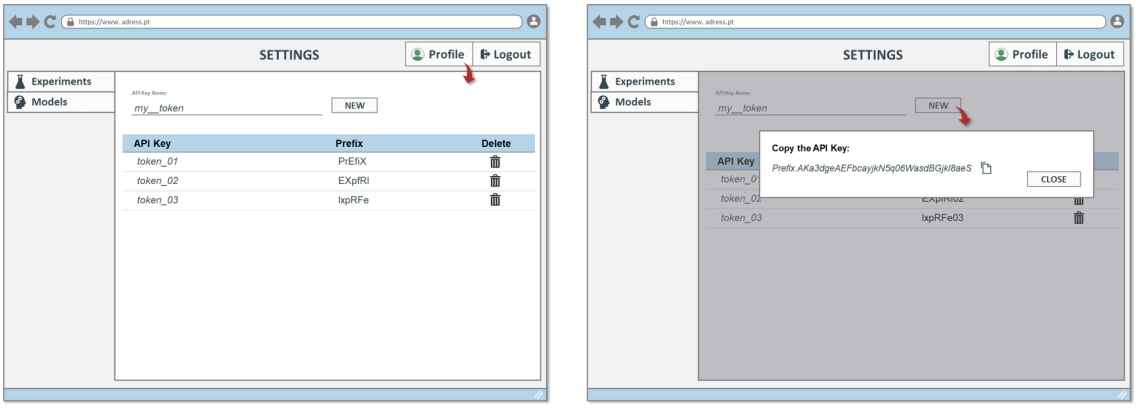

Figure 3.1: Low Fidelity Mockup - Screens for the Settings and API Key creation and visualization.

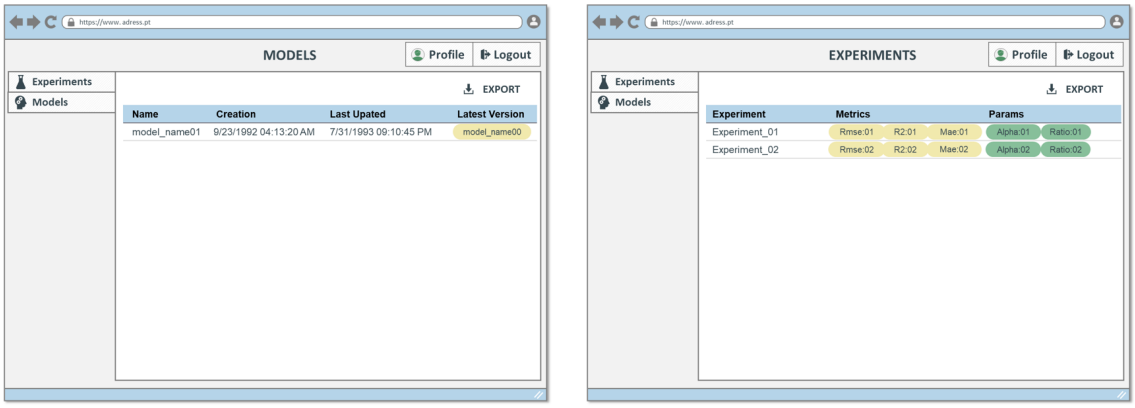

Figure 3.2: Low Fidelity Mockup - Screens for the Artifacts: experiments and models.

For this framework, the decision was to implement the interface as a browser-based interface, so the mockups reflect this format (**R.1**, **R.1.1**). The mockup includes the following concepts:

- **Sign In page** - Allows user to sign into the platform in a secure way. Thus allowing authenticated user to manage its private information, experiments and models. This page is provided by INESCTEC's authentication provider, and redirects the user to our platform after successfully authentication (**R.3**, **R.3.3**).
- **Experiments page** - Display detailed information of all experiments made by or shared with the user (**R.1.1**).
- **Models page** - Display the trained models in each experiment made by or shared with the user (**R.1.1**).
- **Settings page** - Allow user to create, view and delete API keys (**R.1.2**, **R.3.4**).
- **Application Bar** - At the right side it should also include a *Settings* and *Logout* buttons under a dropdown for authenticated users.

During the development process, some alterations were made, and so the final result differs slightly from the initial concept. Those will be better explored in chapter 4 including the reasoning behind those changes.

3.5.2 Prototype Validation

Considering the use of a non-functional prototype, the validation consists of demonstrating how the mockup is related to the functionalities that should be performed and how each page corresponds to each requirement. The requirements described on section 3.3 could be directly correlated to the pages at section 3.5.1 as presented above.

The stakeholder validated that the presented concept would be user-friendly and should, in theory, provide the required functionalities. Thus the work progressed to the development stage.

Chapter 4

Developing a Framework for Reproducible Experiments

This chapter will describe the development stage of the Framework.

For the creation of the Framework, the use of Open Source tools already available was taken into consideration, when applicable. section 4.1 dives into some of those options that were applied to facilitate the construction and aggregate high-level functions that were already developed by others.

The following section 4.2 describes the Framework design and development, including the architecture design and the description of components behavior. Including a dive into the back-end composition elements and functionalities, and a similar approach for the front-end. The section also includes a dedicated segment for the authentication of the user.

Closing this chapter is section 4.3 with the validation of the Framework.

4.1 Tools

This section will describe some of those Open Source platforms, management tools, and collaboration environments that supported the development of the Framework. Those were created by different organizations and improved by multiple users evolving their applications and functionalities. They have served to simplify functionalities, support design, and guarantee security, as so, should be included as an integral part of this research.

4.1.1 MLflow

MLflow is an Open Source platform that manages the machine learning life cycle including experiment tracking, model management, model deployment, and registry, increasing the reproducibility of the model [22]. It was developed by Databricks and designed to be a wide range of tools to help developers manage data preparation, model training, and deployment with an open interface

working with multiple libraries, algorithms, programming languages, and deployment tools and collaborative libraries [22].

MLflow provides a REST API, called Tracking API, that allows applications to communicate with the server to log artifacts. Additionally, libraries that implement this communication with the REST API are available for Python, R, and Java and can be used to facilitate integration. This allows better documentation of all the runs of an experiment, by logging all variables that are subject to change between runs.

MLflow also offers the possibility to run experiments made by others, given that a descriptor file is present at the source code. It is called MLflow Projects, in which the descriptor file contains information about the programming language that was used, and all dependencies that the code may have. That allows MLflow to run the experiment, without having to set up any of the dependencies. There are two kinds of descriptors based on the environment that the project is set to run: Docker and Conda.

Although it offers great functionalities towards reproducibility, this tool does not address important concerns with data access control. All data that is sent to the MLflow server is visible to every user that has access to the server address. The organization that uses it needs to add a layer of authentication in front of the server, and authorization is not simple to implement.

MLflow is available at GitHub and was used in its version 1.17.0 from May 8th, 2021 [22].

This tool was used as a building block for our framework because of the functionalities it already offers. The main functionality of interest for the MVP was the Tracking API, which could be used to log all the runs of an experiment, as well as save the trained model in a file. The MLflow Projects also play a very important role in the reproducibility, but will not be integrated into this initial phase. The data access control will be addressed by the framework, to improve the usability across an organization.

4.1.2 Keycloak

Keycloak is an Open Source Identity and access management [25]. It is under the stewardship of Red Hat and was designed to provide an easier way to do authentication and security, using single-sign-on and being available for multiple platforms and programming languages [25].

Keycloak is available for download at [25] and was used in its version 13.0.1 from May 25th, 2021.

The tool was used to guarantee that private information could not be access by unauthorized people, while also simplifying the implementation and guaranteeing ongoing secure and updated authentication methods. In addition, our framework does not have the need of saving credentials, as it is already handled by Keycloak, thus minimizing the security risks of having a breach in the system that could compromise those credentials. Keycloak is already used by INESCTEC as a standard authentication method across the organization for researchers, students, and partners. As an authentication method was required, Keycloak was not only a good but rather a natural choice to guarantee the integration with their services.

4.1.3 Lagom

Lagom is an Open Source framework to build systems with Reactive micro-services [29]. It is under the stewardship of Lightbend and was designed to enable the developer to better define responsibility, do releases with reduced risk and make better use of modern computing environments [29].

Lagom is available for download at [29] and was used in its version 1.6.5 from April 9th, 2021.

As our framework will have much space for growth, using Lagom facilitates the evolution of the software architecture, by guaranteeing responsiveness, resilience, scalability, and elasticity.

4.1.4 React

React is a library for JavaScript used for building user interfaces. It is Open Source and mainly maintained by Facebook.

React is available at [33] and the version used was 17.0.0 from October 20th, 2020.

This framework allows developers to implement interface components that can be easily reused in other projects. As a consequence, there are a huge amount of Open Source components available to be used, which can drastically reduce the time spent developing a new interface.

4.1.5 ImmuneML

ImmuneML is an Open Source platform for ML-based analysis and classification of AIRR [6]. It was developed by researchers from the University of Oslo, with the work of multiple contributors, to train ML models, apply already trained ML models into new datasets, as well as explore properties of datasets and simulate synthetic data [13].

ImmuneML is available at Github and was used as base for alterations in its version 1.2.5 from April 27th 2021 [13].

As this library already encapsulated much of the code needed for training models, it was needed to make additions to the source code, in order to connect to our framework and log experiments data, namely: the parameters used, metrics generated and model trained. These alterations were made by creating a fork of the source code and implementing the usage of the MLflow library to achieve these purposes.

4.2 Software Design and Development

Software Design is an important part of the development process which includes the architecture design and the detailed design. The architecture is a high-level design that shows how the software is organized into components, as illustrated in figure 4.1. While the detailed design is the description of the behavior of the components [16].

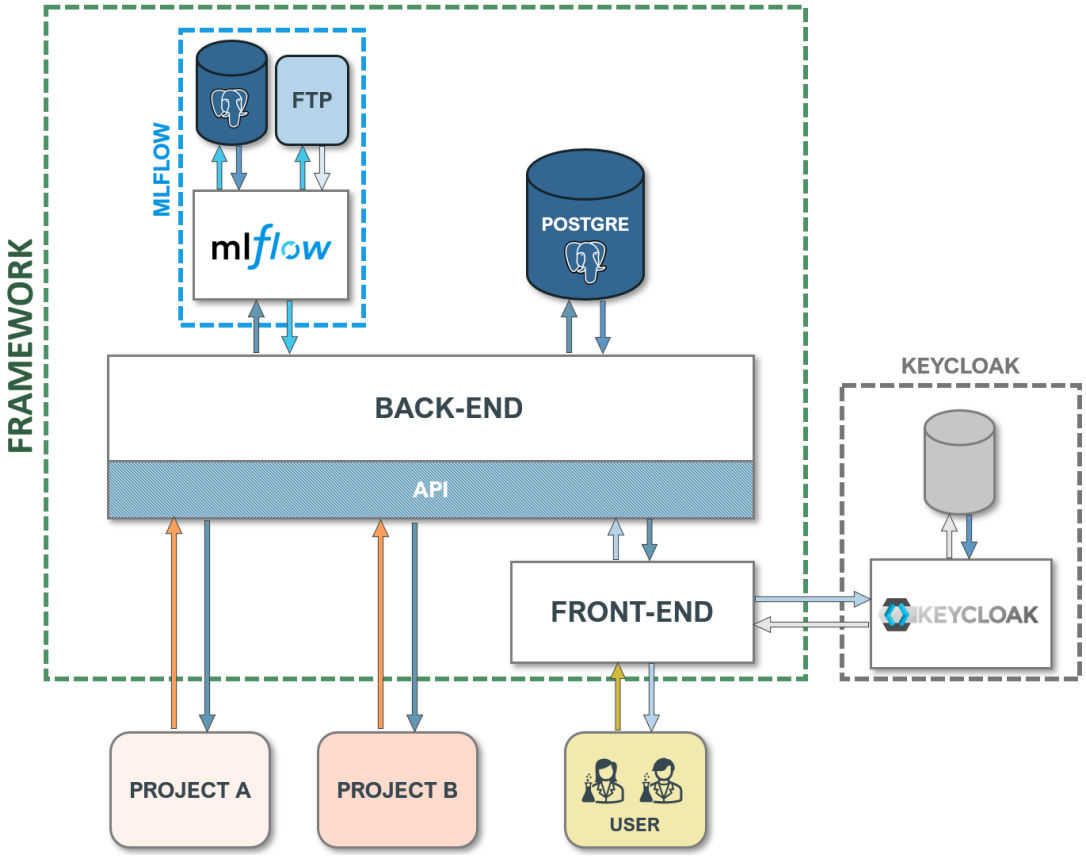

Figure 4.1: Technical Drawing illustrating the connections between the elements that compose the Framework and the outside access.

It is fundamental to decompose, organize and package the software components as well as the properties that affect the components' performance or semantics in systemic ways (aspects) [16] and could be decomposed within the scope of this Framework as:

- **Concurrency** - Part of the design concerned with breaking the software into tasks, processes, and threads, and also dealing with issues of efficiency, synchronization, and scheduling [16]. Lagom is used to handle concurrent requests made from outside the framework to its API.
- **Data Persistence** - Part of the design concerned with handling long-living data that requires storage [16]. The data related to user authentication, namely the user id and a simple profile with basic information about the user such as name, email, user id and whether the email was verified, and also the data that links the user to their resources are persisted on PostgreSQL, connected to the back-end of the Framework.
- **Error and Exception Handling and Fault Tolerance** - Part of the design concerned with dealing with exceptional conditions, including preventing, tolerating, and processing errors [16]. Lagom also provides the tool to implement error and exception handlers, while also guaranteeing a fault tolerance system, by recovering from most of the errors.

- **Interaction and Presentation** - Part of the design concerned with presenting the information and how to structure and organize communication with the users of the software [16]. The interaction with them can either be made through the website or via REST API.

- **Security** - Part of the design concerned with restricting information and other resources accessible to the user. Should prevent unauthorized creation, change, erasure, and visualization of the artifacts. It also considers the use of cryptography and the prevention of security-related attacks [16]. The authentication is handled by Keycloak, which communicates with the front-end, which interfaces it with the back-end. The back-end also should be able to identify applications that make requests directly to the framework.

4.2.1 Back-end

The Back-end, as the server-side of the framework, handles the connections between the front-end and/or applications and the MLflow server. These connections can be seen in figure 4.1.

It adds a layer of authentication and also persists the users' IDs to remember which one has access to what. It is part of the back-end responsibility to generate the API Key for authenticated users, so they can use that as a credential for applications to connect to the framework.

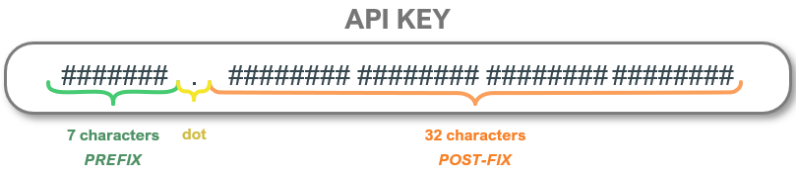

Figure 4.2: Process of Authentication.

When a new API key is requested by the user, the back-end generates one composed of 3 parts, as illustrated in figure 4.2: a random prefix of 7 characters; a character '.'; and a postfix of 32 random characters. Then a hash algorithm is applied to the key, in this case, SHA-256 was used. After that, the hashed key is persisted in the database together with the user ID that came in the request, and the prefix in plain text. Finally, the API key is returned to the user and displayed on the front-end, one time only. This process can be seen in figure 4.3

The API key is not persisted in plain text for security reasons. If the database suffers a malicious attack, and the data is compromised, the attacker will not possess the user API keys, thus not allowing them to operate on behalf of the users. For this reason, the back-end cannot return the same API Key after the generation process, creating the necessity for the user to save it in a secure place. Saving a small prefix in plain text serves the purpose of informing the user what is the name of each key already generated, if, by any chance, the user has many keys. It also allows the user to revoke the access of a certain key, based on the prefix that matches the key.

The API key can then be added to Machine Learn applications, by using the MLflow library, to connect and authenticate with our framework. This communication is handled by implementing a REST API that follows the contract established by the MLflow API, allowing the framework to

intercept the requests that would go to the MLflow Server. This enables the framework to act as a middle-man, and persist an association between the resources and the requesting user.

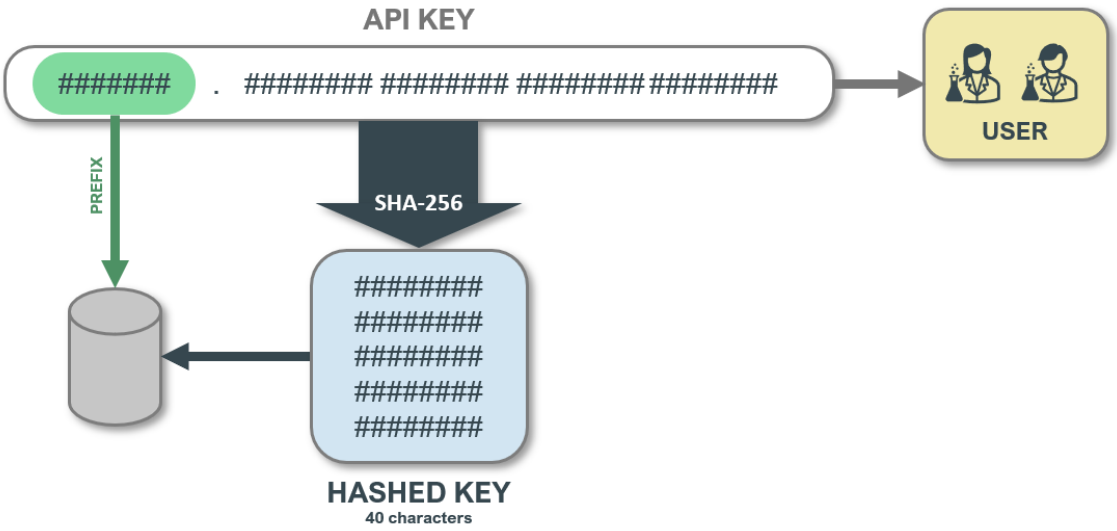

Figure 4.3: Process of Authentication.

4.2.2 Front-end

The Front-end, as the client-side of the framework, allows the user to visualize the artifacts in a website by connecting them to the API and the authenticating provider. These connections can be seen in figure 4.1.

It redirects unauthenticated users to a login page provided by Keycloak. After successfully authentication it receives a token and stores it on the user browser. On every request the front-end sends the token on the header, guaranteeing the security of the process by allowing users to only visualize the artifacts persisted by them. The authentication process will be explained in section 4.2.3 below.

4.2.3 Authentication

The Framework uses the OpenConnect ID protocol to handle the authentication. Therefore, when users enter the website a verification occurs to validate whether they are authenticated, the flow can be seen in figure 4.4. In case of unauthenticated users, they will be redirected to a Keycloak page where they should insert their credentials. After successful authentication, the user is redirected back to the website, and in this redirection request, an access code is carried on the header. With the access code, the website is able to retrieve a JSON Web Token from Keycloak, which is persisted in the browser. Then, the front-end can send requests to the back-end with this authentication header that carries said token. To retrieve the identity of the user, the back-end should possess a public key provided by the Keycloak, in order to decrypt the encrypted part of the token, and verify that the sender really is who it says it is.

In case the organization removes access from the user, the OpenConnect ID defines a standard way to handle this case, preventing the user to keep his access. When the authentication with Keycloak occurs, a set composed of two tokens is retrieved. One access token with a small expiration time, by default 30 minutes. And a second refresh token that can have a larger expiration time, set for many days later. This will be configured depending on how long the users should stay logged before the front-end requests their credentials again. The refresh token serves the only purpose of retrieving a new valid token by making a new request to the Keycloak server, and by doing so, verifying if the user still has permission to access the framework.

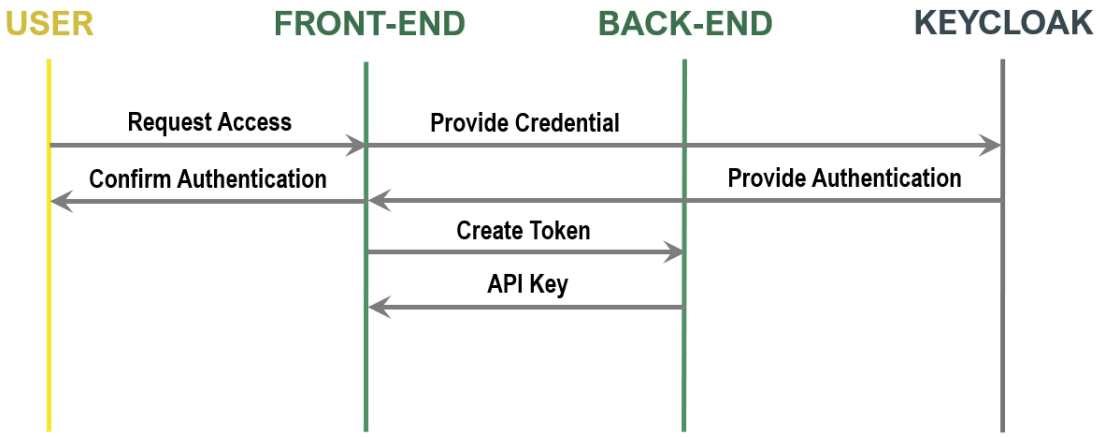

Figure 4.4: Process of Authentication from user to Keycloak.

4.3 Validation of Framework

The verification and validation of the results would determine whether the product conforms to the requirements and satisfies its intended use and the users' needs [16].

The Framework was validated by setting up two ML projects that make use of the available functionalities. One project was created with a simple training process that fits a linear model. The other project was created using the ImmuneML library with an example code that trains a model over a sample dataset. For this case, the altered version of the ImmuneML library was used and configured to publish the artifacts to our framework.

The source code presented in the Appendix A.1 represents the adjustment of one of the MLflow training models to work within the Framework. Once altered, the experiment run and achieved the results illustrated in the figures 4.5 and 4.6 below.

```
Elasticnet model (alpha=0.700000, l1_ratio=0.700000):
  RMSE: 0.8593197064520263
  MAE: 0.6481230414016983
  R2: 0.04626127866200236
Registered model 'ElasticnetWineModel' already exists. Creating a new version of this model...
2021/06/28 05:33:19 INFO mlflow.tracking._model_registry.client: Waiting up to 300 seconds for model version to finish creation.
  Model name: ElasticnetWineModel, version 7
Created version '7' of model 'ElasticnetWineModel'.
```

Figure 4.5: Results of the MLflow training viewed at the application log.

Id ↑	Status	Data.metrics	Data.params
80e2c09003fe4f988a31d1795042338e	FINISHED	rmse: 0.859319706452026 r2: 0.0462612786620024 mae: 0.648123041401698	alpha: 0.7 l1_ratio: 0.7

Figure 4.6: Results of the MLflow training viewed at the website.

Similarly, the code presented in the Appendix A.2 represents the descriptor file for a training pipeline using ImmuneML. With the alterations mentioned in the section 4.1.5, some configuration parameters could be added to connect to the Framework and log the desired artifacts. Then, the experiment run and achieved the results illustrated in the figures 4.7 and 4.8 below.

```
2021-06-28 03:28:47.046518: Instruction 1/1 has finished.
2021-06-28 03:28:47.058485: Generating HTML reports...
2021-06-28 03:28:47.088549: HTML reports are generated.
2021-06-28 03:28:47.230890: Pusblishing parameters used to mlflow
2021-06-28 03:28:47.345231: Pusblishing obtained metrics of model to mlflow
2021-06-28 03:28:47.468312: Pusblishing obtained model to mlflow
2021-06-28 03:28:47.604757: ImmuneML: finished analysis.
```

Figure 4.7: Results of the ImmuneML training viewed at the application log.

Id ↑	Status	Data.metrics	Data.params
2f23eaa403bb431bac2edbec2c2b3287	FINISHED	auc: 1 balanced_accuracy: 1 precision: 1 recall: 1	model_selection_cv: False model_selection_n_folds: -1 C: 0.5 penalty: l1

Figure 4.8: Results of the ImmuneML training viewed at the website.

4.4 Use Case Examples

This section will present the some usage examples of the developed Framework. Similar to the prototype's validation, the example will be presented in the form of how the Framework's website is related to the functionalities that should be performed and how each page corresponds to the requirements described in section 3.4. Each part will be accompanied by an explanation of the

process following a single flow from a user's first access, going through the use of the Framework in all the available functionalities, and eventually logout from the framework.

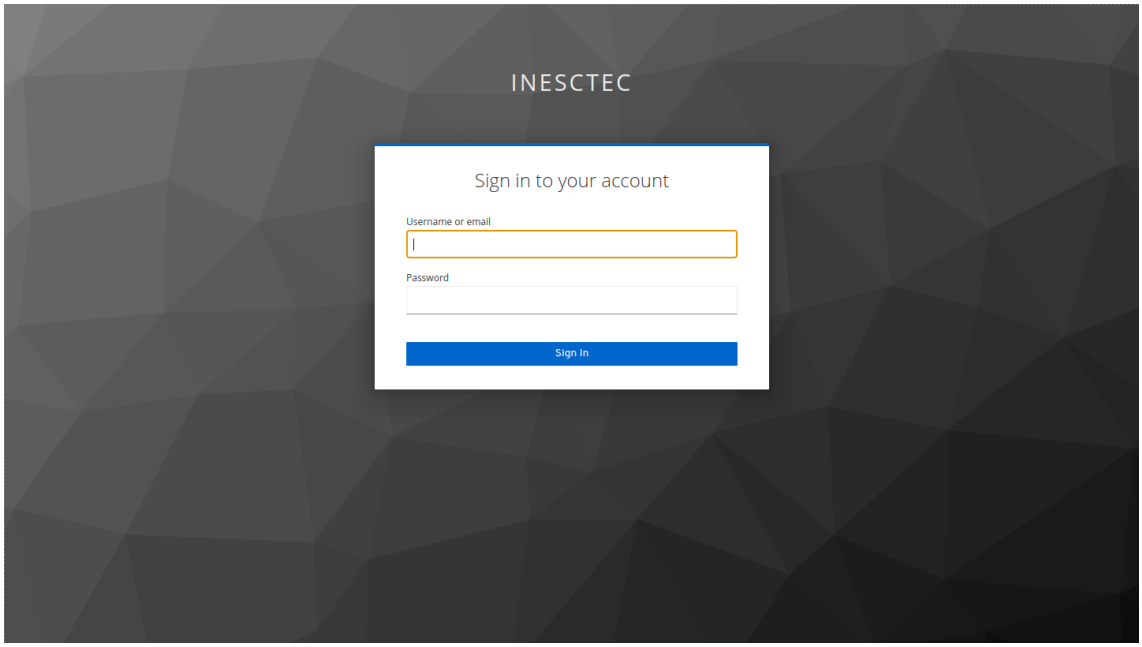

Figure 4.9: Keycloak sign in page.

As described in section 4.2.3 when the user accesses the website through any page within the domain, the front-end will verify if the browser already has the user's credential stored. Assuming the first interaction where the user is not yet logged in, they will automatically be redirected to a Keycloak sign-in page as illustrated in figure 4.9.

On this page, users should insert their credentials. This data will be validated with the registered information in the INESCTEC Keycloak database. And only after successful authentication occurs, the user will be redirected to the main page and granted access to the framework in its integrity. At this point, the front-end received a token and persisted in it in the web browser, allowing the user to remain logged in.

The authenticated user will have access to two options when clicking on the *user profile* icon at top right corner of the page: *Settings* and *Logout*, as illustrated on figure 4.10.

By clicking in *Settings* the user will be redirected to said page, as illustrated on figure 4.11. This page displays the list of tokens created by the user and allows them to create new ones.

To create a new API key, as described in section 4.2.1, the user only has to provide a unique name and click on the *new* button. A pop-up will open, as illustrated in figure 4.12, with a warning to copy and register the API key in a safe place, as it will not be displayed again and can not be recovered. Once the user clicks on *Understood* this pop-up will close, and the new key will be added to the list.

Back to the Settings screen, the tokens created by the user are presented in a table including:

1. API Key name as given by the user.

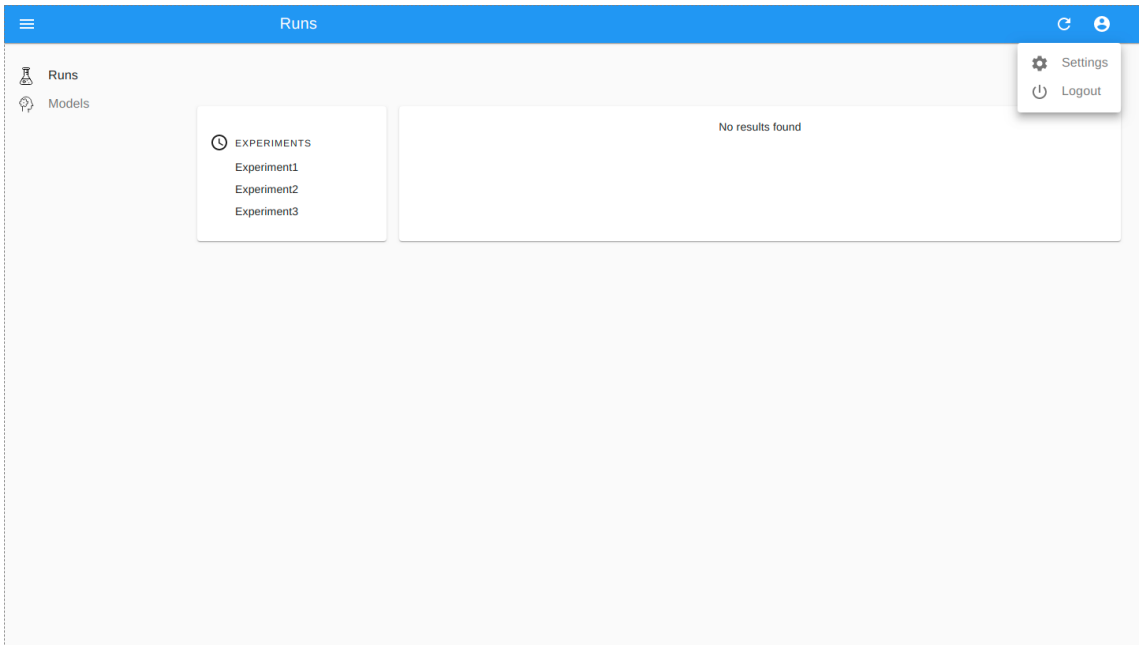

Figure 4.10: The location of the settings menu.

2. Prefix of the API Key, before being hashed.

3. Delete button to exclude and invalidate the API key.

The user can have multiple API Keys to distribute through their applications. The API Keys are not single-use, thus the user can correlate them with as many applications as they would like and at their own discretion.

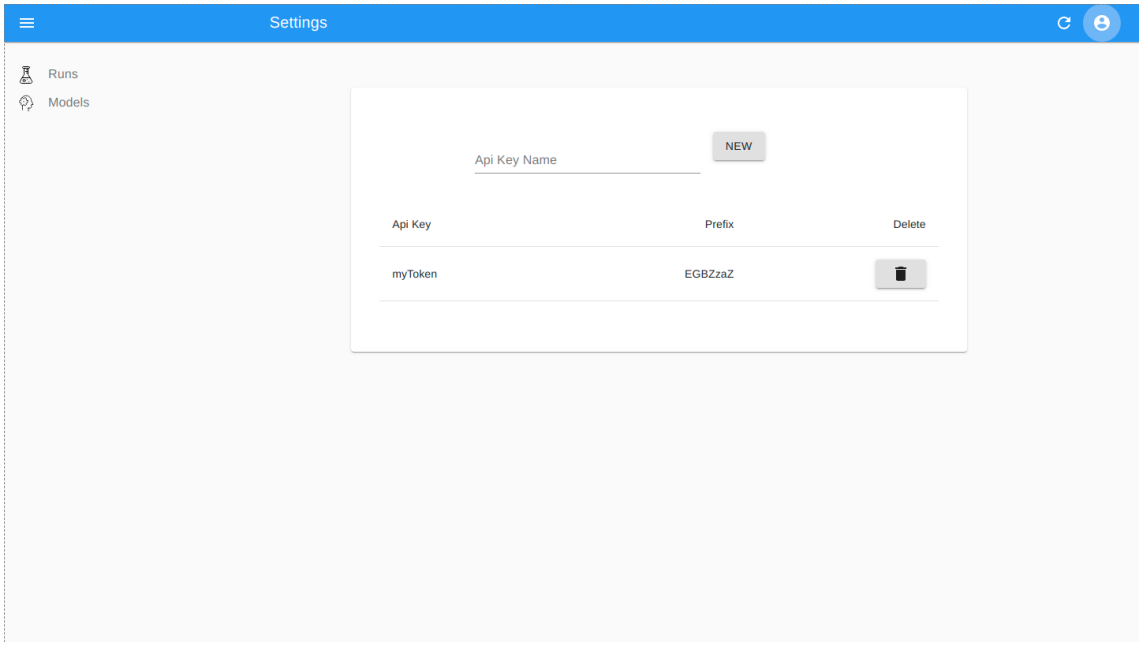

Figure 4.11: The settings page, that displays the list of tokens created by the user.

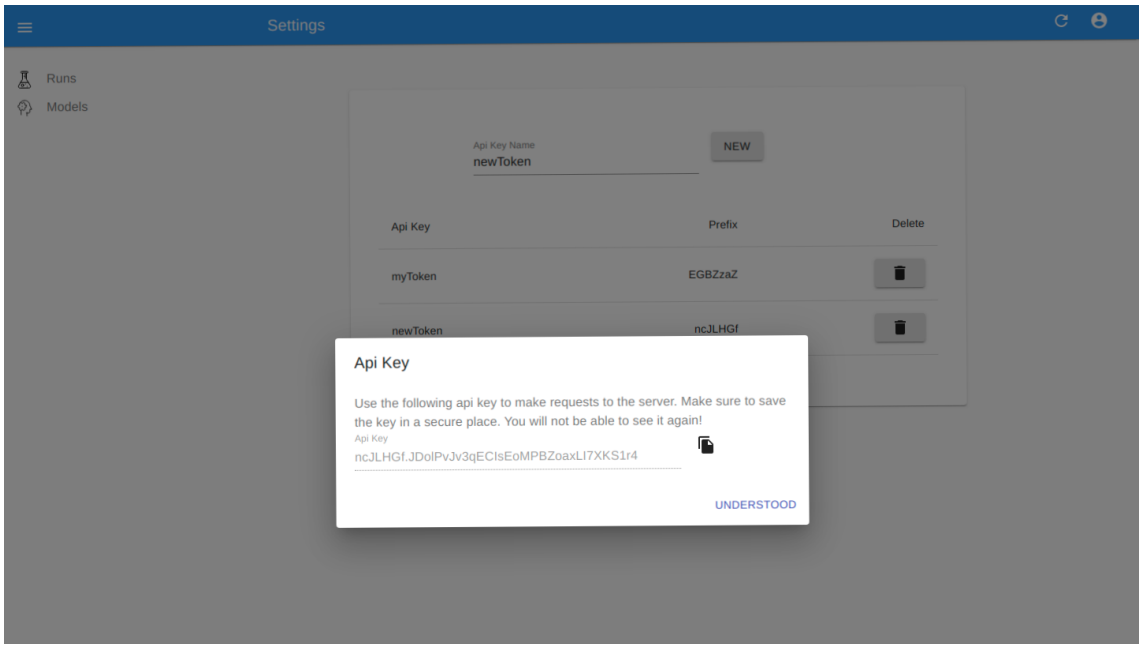

Figure 4.12: Dialog that displays the API Key upon creation.

At the left side of all the pages the user can view a menu that includes two options: *Runs* and *Models*. Depending on the sizing of the screen this menu can be hidden, and in this case, would be necessary to click on the button on the top left corner of the page to show the side menu with those options.

By clicking in *Runs* the user will be redirected to said page, as illustrated on figure 4.13. This page displays the experiments available for them to visualize, indicated by the name given by the user when it was created.

Once the user selects one of the experiments available the page will load its run's information in a table, including:

1. The run's ID, randomly generated.
2. The status of the run, which can be: *running*, *schedule*, *finish*, *failed* or *killed*.
3. The calculated metrics correspondent to the performance of the trained model in each run.
4. The parameters used at the model in each run.

The stored runs are the register of the various iterations of an experiment, maintaining a log of the multiple attempts to achieve the optimal results. This allows the researcher to track which set of parameters were tried and see how they affected the results.

By clicking in *Models* the user will be redirected to said page, as illustrated on figure 4.14. This page displays the models saved by the user in the form of a table including:

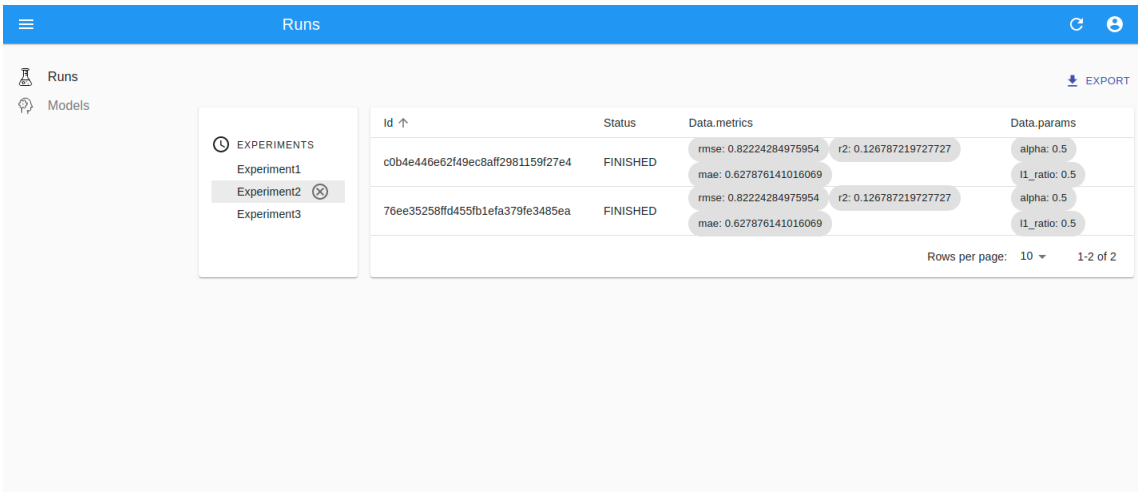

Figure 4.13: Experiment page displaying the runs for a specific experiment.

1. The model's name, as given by the user.
2. The creation timestamp of the first version of the model.
3. The last update timestamp for the most recent version of the model.
4. The last versions of the model saved.

It also includes in the right corner of the page, below the toolbar, an *export* button, which lets the user generate a CSV file with the table's information.

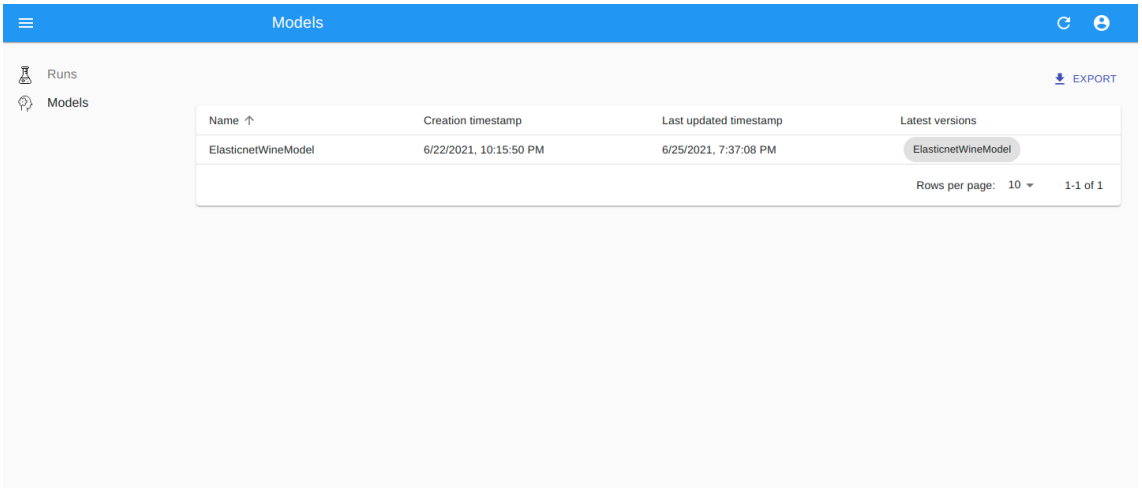

Figure 4.14: Model page, displaying the different models that were uploaded.

Finally, once the user finishes making use of the Framework, they can logout of it by clicking in the *logout* button. This will end the session, by erasing the stored token from the browser, and to be able to access the pages again, a new login will be required.

Chapter 5

Conclusions

The project is considered complete as the plans and features established as the scope of the MVP in section 3.4 were achieved. This chapter's section 5.1 dives into the goals that were set and achieved by the framework.

Improvements on the software can be made based on the remaining requirements that were listed in section 3.3 and not included under the MVP. These can be further explored and developed in future projects and could be used by INESCTEC as a guideline for follow-up work, as suggested in section 5.2.

5.1 Contributions

The main goal of the work developed was to provide an evolving Framework that could aid researches to be more reproducible. Though the motivation for this work came from the health domain, the end result could be applicable not only to this field, but also expanded to others.

The software was successfully developed under this scope attending the MVP's requirement. By presenting a simple and intuitive user interface that allows a researcher to visualize their experiments with its multiple runs and their models, as well as detailed tracking of their parameters, metrics, and dependencies. These are stored in a persistent way to allow the user to revisit and compare the results. Additionally, the artifacts could either be uploaded by them directly or through other applications that can communicate with the framework, and these types of requests are discriminated. It does not disclose any of its information, that is stored in a secure server, to unauthorized users. And is fully integrated with the INESCTEC authentication server. Additionally, it can eventually be open to researchers outside of the organization.

By combining the MLflow to ImmuneML's libraries the Framework also demonstrated that the integration of additional libraries to it does not add that much extra complexity.

5.2 Future Work

The Framework would benefit from the addition of functionalities that attend to the remaining requirements analyzed in this work.

Some of each could be the control of the information's access authorizations with different categories such as private, shareable with a specific user, or totally public. This would allow, between other things, for artifacts to be shared with different users. This feature should also be accompanied by the capability to encapsulate the experiment executable and its dependencies, so other researchers can run the experiment more easily, without the need for software setup. The encapsulated work should also have the possibility of being published with a specific version of an experiment with a unique identifier and link for work citations.

As a suggestion for future works the Framework could evolve into a final product that attends to all the customer's requirements in the most optimal way and maintains space to improve and be expanded. Some of the features that could attend the aforementioned requirements are:

- Support logging figures and plots and then display them on the front-end.
- Integrate with MLflow Project, in order to provide a better way to encapsulate the experiments. This would all other researchers to re-run the experiments without the need for setup. This also establishes a link between the experiment and the source code.
- Integrate with a notebook-like environment, such as Colaboratory and Jupyter, to provide a way that other users can view and run the experiment directly on their browser. This could facilitate peer review by reducing the need for setup and waste of resources. This environment also allows simultaneous access and edition to a work as a nice feature.
- Allow the user to publish an experiment as a persisted copy of all artifacts and the source code related to it. And also have the published version assign a Digital Object Identifier and make it publicly available to the community.
- Benchmarking experiments that may implement different models to achieve the same goals.

As this dissertation shows, this field still has a lot of room for improvements. It is expected that the features previously mentioned could enhance reproducibility and help researchers to create valid and more trust-worthy AI models. The progress with this work, could also be expanded to other fields that not healthcare, maintaining that they share similar requirements.

Appendix A

Source Code

This Section includes supplementary information for the reader's understanding.

A.1 MLflow Sample Code

The code described below received some alteration in relation to the original version, a *MLflow training models* available at [10]. The alterations were made to: allow access to the dataset in a local domain and name the experiments.

```
# train.py
# The data set used in this example is from http://archive.ics.uci.edu/ml/datasets/Wine+Quality
# P. Cortez, A. Cerdeira, F. Almeida, T. Matos and J. Reis.
# Modeling wine preferences by data mining from physicochemical properties.
# # In Decision Support Systems, Elsevier, 47(4):547-553, 2009.

import warnings
import sys

import pandas as pd
import numpy as np
from sklearn.metrics import mean_squared_error, mean_absolute_error, r2_score
from sklearn.model_selection import train_test_split
from sklearn.linear_model import ElasticNet
from urllib.parse import urlparse
import mlflow.sklearn

import logging

logging.basicConfig(level=logging.WARN)
logger = logging.getLogger(__name__)


def eval_metrics(actual, pred):
    rmse = np.sqrt(mean_squared_error(actual, pred))
    mae = mean_absolute_error(actual, pred)
    r2 = r2_score(actual, pred)
    return rmse, mae, r2

```

```
30
31  if __name__ == "__main__":
32
33      warnings.filterwarnings("ignore")
34      np.random.seed(40)
35
36      # Download the CSV from
37      # http://archive.ics.uci.edu/ml/machine-learning-databases/wine-quality/winequality-red.csv
38      # Read the wine-quality csv file from the CSV
39      csv_url = (
40          "wine-quality.csv"
41      )
42      try:
43          data = pd.read_csv(csv_url, sep=",")
44      except Exception as e:
45          logger.exception(
46              "Unable to download training & test CSV, " +
47              "check your internet connection. Error: %s", e
48          )
49
50      # Split the data into training and test sets. (0.75, 0.25) split.
51      train, test = train_test_split(data)
52
53      # The predicted column is "quality" which is a scalar from [3, 9]
54      train_x = train.drop(["quality"], axis=1)
55      test_x = test.drop(["quality"], axis=1)
56      train_y = train[["quality"]]
57      test_y = test[["quality"]]
58
59      alpha = float(sys.argv[1]) if len(sys.argv) > 1 else 0.5
60      l1_ratio = float(sys.argv[2]) if len(sys.argv) > 2 else 0.5
61
62      mlflow.set_experiment("Experiment2")
63      with mlflow.start_run():
64          lr = ElasticNet(alpha=alpha, l1_ratio=l1_ratio, random_state=42)
65          lr.fit(train_x, train_y)
66
67          predicted_qualities = lr.predict(test_x)
68
69          (rmse, mae, r2) = eval_metrics(test_y, predicted_qualities)
70
71          print("Elasticnet model (alpha=%f, l1_ratio=%f):" % (alpha, l1_ratio))
72          print("  RMSE: %s" % rmse)
73          print("  MAE: %s" % mae)
74          print("  R2: %s" % r2)
75          print("artifact_uri: {}".format(mlflow.get_artifact_uri()))
76
77          mlflow.log_param("alpha", alpha)
78          mlflow.log_param("l1_ratio", l1_ratio)
79          mlflow.log_metric("rmse", rmse)
80          mlflow.log_metric("r2", r2)
81          mlflow.log_metric("mae", mae)
82
83          print(mlflow.get_artifact_uri())
84
85          tracking_url_type_store = urlparse(mlflow.get_tracking_uri()).scheme
86
87          # Model registry does not work with file store
```

```
        if tracking_url_type_store != "file":

            # Register the model
            # There are other ways to use the Model Registry, which depends on the use case,
            # please refer to the doc for more information:
            # https://mlflow.org/docs/latest/model-registry.html#api-workflow
            mlflow.sklearn.log_model(lr, "model", registered_model_name="ElasticnetWineModel")
        else:
            mlflow.sklearn.log_model(lr, "model")
```

A.2 ImmuneML Sample Code

The code described below received some alteration in relation to the original version, a *YAML specification* available at [11]. The alterations added configuration to set up a connection with the Framework and enable the log-in of artifacts.

```
# specs.yaml
definitions:
  datasets:
    my_dataset: # user-defined dataset name
      format: AIRR
      params:
        # we are importing a repertoire dataset
        is_repertoire: true
        # path to the folder containing the repertoire .tsv files
        path: quickstart_data/repertoires/
        metadata_file: quickstart_data/metadata.csv

  encodings:
    my_kmer_frequency: # user-defined encoding name
      KmerFrequency:   # encoding type
        k: 3           # encoding parameters

  ml_methods:
    my_logistic_regression:
      # user-defined ML model name: ML model type (no user-specified parameters)
      LogisticRegression:
        C: 0.5
        penalty: l1

  reports:
    # user-defined report name: report type (no user-specified parameters)
    my_coefficients: Coefficients

  output:
    my_mlflow:
      MLflow:
        experiment_name: immuneML
        url: http://localhost:9000/
        log_params: true
        log_metrics: true
        log_model: true

```

```yaml
instructions:
  my_training_instruction: # user-defined instruction name
    type: TrainMLModel

    dataset: my_dataset # use the same dataset name as in definitions
    labels:
    - signal_disease    # use a label available in the metadata.csv file

    settings: # which combinations of ML settings to run
    - encoding: my_kmer_frequency
      ml_method: my_logistic_regression

    assessment: # parameters in the assessment (outer) cross-validation loop
      reports:  # plot the coefficients for the trained model
        models:
        - my_coefficients
      split_strategy: random   # how to split the data - here: split randomly
      split_count: 1           # how many times (here once - just to train and test)
      training_percentage: 0.7 # use 70% of the data for training

    selection: # parameters in the selection (inner) cross-validation loop
      split_strategy: random
      split_count: 1
      training_percentage: 1 # use all data for training

    # the metric to optimize during nested cross-validation when comparing multiple models
    optimization_metric: balanced_accuracy
    metrics: # other metrics to compute for reference
    - auc
    - precision
    - recall

    number_of_processes: 4 # processes for parallelization

output:
  destination: MLflow
  format: HTML
```

References

[1] Directive 2011/24/eu. Available at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX:32011L0024, mar 2011.

[2] What does the hipaa privacy rule do? Available at https://www.hhs.gov/hipaa/for-individuals/faq/187/what-does-the-hipaa-privacy-rule-do/index.html, jul 2013.

[3] Regulation (eu) 2016/679. Available at https://eur-lex.europa.eu/legal-content/EN/TXT/HTML/?uri=CELEX:32016R0679&from=EN, apr 2016.

[4] Artificial intelligence for europe. Available at https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=COM:2018:237:FIN, apr 2018.

[5] Lei geral de proteção de dados pessoais (lgpd). Available at http://www.planalto.gov.br/ccivil_03/_ato2019-2022/2019/lei/l13853.htm, jul 2019.

[6] Airr community. Available at https://docs.airr-community.org/en/stable/, January 2021.

[7] Hydroshare. Available at https://www.hydroshare.org/, March 2021.

[8] Jupyter. Available at https://jupyter.org/, March 2021.

[9] Minimum viable product in software development: Getting it right. Available at https://codetiburon.com/minimum-viable-product-in-software-development-getting-it-right/, June 2021.

[10] Mlflow: Training the model. Available at https://www.mlflow.org/docs/latest/tutorials-and-examples/tutorial.html#training-the-model, May 2021.

[11] Quickstart: command-line interface with yaml. Available at https://docs.immuneml.uio.no/latest/quickstart/cli_yaml.html#step-2-writing-the-yaml-specification, May 2021.

[12] Sciunit. Available at https://sciunit.run/, March 2021.

[13] Welcome to the immuneml documentation! Available at https://docs.immuneml.uio.no/index.html, April 2021.

[14] What is colaboratory? Available at https://colab.research.google.com/notebooks/intro.ipynb?hl=en#scrollTo=5fCEDCU_qrC0, April 2021.

[15] S. Badillo, B. Banfai, F. Birzele, I.I. Davydov, L. Hutchinson, T. Kam-Thong, J. Siebourg-Polster, B. Steiert, and J.D. Zhang. An introduction to machine learning. *Clinical Pharmacology and Therapeutics*, 107(4):871–885, 2020.

[16] P. Bourque and R.E. Fairley. *Guide to the Software Engineering Body of Knowledge*. IEEE Computer Society, version 3.0 edition, 2014.

[17] M.L. Braun and C.S. Ong. *Open science in machine learning*. 2014.

[18] Jonathan Buckheit and David Donoho. Wavelab and reproducible research. *Wavelets and Statistics*, Vol. 103, 11 1998.

[19] Neil Calman, Kwame Kitson, and Diane Hauser. Using information technology to improve health quality and safety in community health centers. *Progress in community health partnerships : research, education, and action*, 1:83–8, 03 2007.

[20] Y.-D. Choi, J.L. Goodall, J.M. Sadler, A.M. Castronova, A. Bennett, Z. Li, B. Nijssen, S. Wang, M.P. Clark, D.P. Ames, J.S. Horsburgh, H. Yi, C. Bandaragoda, M. Seul, R. Hooper, and D.G. Tarboton. Toward open and reproducible environmental modeling by integrating online data repositories, computational environments, and model application programming interfaces. *Environmental Modelling and Software*, 135, 2021.

[21] J. Claerbout and M. Karrenbach. Electronic documents give reproducible research a new meaning. 1992.

[22] Databricks. Mlflow. Available at https://www.mlflow.org/, May 2021.

[23] Chris Drummond. Replicability is not reproducibility: Nor is it good science. *Proceedings of the Evaluation Methods for Machine Learning Workshop at the 26th ICML*, 01 2009.

[24] O.E. Gundersen and S. Kjensmo. State of the art: Reproducibility in artificial intelligence. pages 1644–1651, 2018.

[25] Red Hat. Keycloak. Available at https://www.keycloak.org/, May 2021.

[26] Ralph Johnson and Brian Foote. Designing reusable classes. *Journal of Object-Oriented Programming*, 1:22–35, 06 1988.

[27] S.B. Kotsiantis. Supervised machine learning: A review of classification techniques. *Informatica (Ljubljana)*, 31(3):249–268, 2007.

[28] David M. Lane. *Introduction to Statistics*. 2003.

[29] Lightbend. Lagom. Available at https://www.lagomframework.com/, April 2021.

[30] M. B. A. McDermott, S. Wang, N. Marinsek, R. Ranganath, M. Ghassemi, and L. Foschini. Reproducibility in machine learning for health. 2019.

[31] Kari Liukkunen Nirnaya Tripathi, Markku Oivo and Jouni Markkula. Startup ecosystem effect on minimum viable product development in software startups. *o Journal of Information and Software Technology*, 3, June 2019.

[32] Arjun Panesar. *Machine Learning and AI for Healthcare: Big Data for Improved Health Outcomes*. Apress, USA, 1st edition, 2019.

[33] React. React. Available at https://reactjs.org/, April 2021.

[34] Dirk Riehle. Framework design: A role modeling approach. *Softwaretechnik-Trends*, 20, 01 2000.

[35] Don Roberts and Ralph Johnson. Evolving frameworks: A pattern language for developing object-oriented frameworks. 01 1996.

[36] G.K. Sandve, A. Nekrutenko, J. Taylor, and E. Hovig. Ten simple rules for reproducible computational research. *PLoS Computational Biology*, 9(10), 2013. cited By 291.

[37] Nathalie A. Smuha. The eu approach to ethics guidelines for trustworthy artificial intelligence. *Computer Law Review International*, 20(4):97 – 106, 2019.

[38] Darrel M West and John R Allen. How artificial intelligence is transforming the world. *Brookings*.

[39] Greg Wilson, Jennifer Bryan, Karen Cranston, Justin Kitzes, Alexander Nederbragt, and Tracy Teal. Good enough practices in scientific computing. *PLOS Computational Biology*, 13, 08 2016.