# Self-evaluating Machine Learning and Transfer Learning in Healthcare Analytics

**Roser Cantenys Sabà**

Bachelor's Degree Thesis
Degree in Data Science and Engineering
Universitat Politècnica de Catalunya
Amalfi Analytics

Director: Laura Aviñó Esteban
Co-director: Jose Munuera Mora
Speaker: Marta Casanellas Rius

July, 2021

## Acknowledgements

First of all, I would like to thank Prof. Ricard Gavaldà for giving me the opportunity of doing my bachelor's thesis in his company, Amalfi Analytics, and for all the help that he has given me during these months.

I would want to thank my director, Laura, in particular, for working with me on some of the topics and issues of my thesis, as well as for all of her patience and hours spent introducing me to the company's architecture. And thank you to each one of the members of Amalfi Analytics' techteam for taking an interest in my work, coming to our meetings and sharing their advise and opinions, and making me feel like I was a part of the group from the very first day.

Finally, I would want to thank my family for all of their support and assistance throughout the years, especially my parents for instilling in me a love of learning, knowing, and discovering.

**Abstract**

In many cases, hospital resources are mismanaged due to the difficulty of forecasting different services. We have seen that, in order to control a health center many features must be known. Hence, we need a highly adaptable and flexible model that allows us to forecast several distinct aspects.

In this project we present a Meta-Model, an agnostic predictor and classifier that combines several predictors and classifiers from different libraries. It can self-evaluate its performance; incrementally learn from data, when scheduled, i.e it is highly adaptable; be integrated with servers where it is parallelizable. The proposed model stands out for being very flexible in the sense that it allows the user to easily incorporate her or his own desired functions and modules or modify and extend the existing ones.

This model can be very useful in a wide variety of problems where data is changing. In this work, specifically, we propose to use data from hospitals and health centers such as the emergency department arrival time, patient diagnoses, patient waiting times in the emergency room before being attended, healthcare professionals sick level percentage in each department... to predict the daily, hourly or weekly hospital flow which will help to predict the resources needed at any given time as well as improving the supply management.

***Keywords***— Machine Learning, Meta-Model, Agnostic Data Predictor and Classifier, Machine Learning in the Wild

**Resum**

En molts casos, els recursos hospitalaris són mal gestionats a causa de la dificultat de predir diferents serveis. Hem vist que, per controlar un centre de salut, cal conèixer moltes característiques d'aquest. Per tant, necessitem un model molt adaptable i flexible que ens permeti predir diversos aspectes diferents dels centres de salut.

En aquest projecte presentem un Meta-Model, un predictor i classificador agnòstic que combina diversos predictors i classificadors de diferents llibreries. Aquest model pot autoavaluar el seu rendiment; aprendre gradualment de les dades, és a dir, és altament adaptable i integrar-se amb servidors on sigui paral·lelitzable. El model proposat destaca per ser molt flexible en el sentit que permet a l'usuari incorporar fàcilment les seves funcions i mòduls desitjats o modificar i ampliar els ja existents.

Aquest model pot ser molt útil en una gran varietat de problemes en què les dades canvien. En aquest treball, concretament, proposem utilitzar dades d'hospitals i centres de salut com ara l'hora d'arribada del servei d'urgències, els diagnòstics dels pacients, els temps d'espera dels pacients a la sala d'urgències abans de ser atesos, el percentatge de baixes laborals dels professionals sanitaris a cada departament ... per predir el flux hospitalari diari, per hores o setmanal, cosa que ajudarà a predir els recursos necessaris en cada moment, així com a millorar la gestió del subministrament.

***Keywords***— Machine Learning, Meta-Model, Agnostic Data Predictor and Classifier, Machine Learning in the Wild

**Resumen**

En muchos casos, los recursos hospitalarios están mal administrados debido a la dificultad de prever diferentes servicios. Hemos visto que para controlar un centro de salud se deben conocer muchas características. Por lo tanto, necesitamos un modelo altamente adaptable y flexible que nos permita pronosticar varios aspectos distintos de los centros de salud.

En este proyecto presentamos un Meta-Modelo, un predictor y clasificador agnóstico que combina varios predictores y clasificadores de diferentes librerías. El modelo puede autoevaluar su rendimiento; aprender gradualmente de los datos, es decir, es altamente adaptable y estar integrado con servidores donde es paralelizable. El modelo propuesto destaca por ser muy flexible en el sentido de que permite al usuario incorporar fácilmente sus propias funciones y módulos deseados o modificar y ampliar los existentes.

Este modelo puede resultar muy útil en una amplia variedad de problemas en los que los datos están cambiando. En este trabajo, en concreto, se propone utilizar datos de hospitales y centros sanitarios como la hora de llegada a urgencias, diagnósticos de pacientes, tiempos de espera de los pacientes en urgencias antes de ser atendidos, porcentaje del nivel de enfermedad de los profesionales sanitarios en cada servicio... para predecir el flujo hospitalario diario, por horas o semanal, cosa que ayudará a predecir los recursos necesarios en un momento dado, así como a mejorar la gestión de suministros.

***Keywords***— Machine Learning, Meta-Model, Agnostic Data Predictor and Classifier, Machine Learning in the Wild

# Contents

# List of Figures

# List of Tables

# 1 Introduction

The amount of data being digitally collected and stored is huge and growing exponentially. In particular, the extensive processing of health data through electronic health records, genetic sequencing, and digital health wearables has led to an increasingly expanding amount of biomedical "big data" [1, 2]. As a result, the science of data management and analysis is also evolving to enable organizations to convert this immense resource into information and knowledge that will assist them towards achieving their objectives. In the last few years, many new discoveries and methodologies have been published in the field of big data in health care [3, 4].

The volume of digital information available to physicians is becoming bigger and bigger and most of the time there is a lot of information which is not processed neither used. Big data applications such as machine learning (ML), and more recently deep learning, are key techniques that have demonstrated the ability to present different opportunities to discover new knowledge and create novel methods to improve the quality of care [5] and reduce healthcare costs [6]. Some examples of these new trends in the application of ML in healthcare are Natural Language Processing for extracting information from unstructured data in electronic health records [7] and Image recognition and classification for cancer detection and categorizing [8, 9]. Hence, these applications are mostly intended to assist doctors in making their final decisions.

While ML in healthcare is a very active research topic [10, 11], most of the health data collected is never used for building predictive models that are successfully integrated in the clinical setting [12], with only 15% of hospitals currently using ML on a daily basis for some limited purposes [13]. Amalfi Analytics has observed that lately a lot of ML has been applied to the field of medicine obtaining impressive results in personalized medicine [14, 15]; for instance, but the management of most healthcare centers are not benefiting from these technologies yet. They agree that simplified, yet readily understandable models will have a positive impact to care centers and health care more on a day to day basis. Amalfi Analytics believes that applying ML to unused health center data could help them to go from a reactive management model to a proactive one, as they could use and take advantage of the inherent predictive information in data. Therefore, they propose to explore this field.

In this project, we want to build a predictive model to improve the hospital resource management, specifically personnel and bed managements, offering an

improvement in the quality of care while reducing healthcare costs. We want to use already existing and easy methods to extract data, so the predictive model can be easily used in real environments in the hospitals. Therefore, we want a model that exhibits "Machine Learning in the Wild" [16] characteristics. We want to create analytic tools with methods and models that can perform "on their own", without the guidance or supervision of a data scientist.

There is currently a major problem in the management of hospital emergencies, Emergency Department (ED). In several cases, patients wait more than six hours in the waiting room before being assisted by a doctor, saturating this service. The congestion of this unit, following the Pareto Principle [17], determines the overall stability of the hospital.

The influx of people into this service varies greatly depending on several different factors: day of the week, weather, flu periods, etc. which makes the work of the hospital administration people who assign the number of health professionals (doctors, nurses, nursing assistants, etc.) as well as the resources (beds, instruments, etc.) to this department very difficult. Furthermore, these influx peaks are usually accompanied by a higher number of sick leave professionals, which makes management even more difficult.

Typically, projects that predict occupations create one-time trained models that can perform well at first but lose accuracy over time. This is particularly true and takes even more importance in the present scenario, when the Covid-19 pandemic is drastically altering healthcare trends and patterns.

Therefore, in this work we propose a highly adaptable Meta-Model, which is created by combining stacked base models, to predict the occupation of emergency room (ER) as well as the number of sick leaves taken by health professionals of any desired department. This model will help hospital managers to assign the resources needed, minimizing queues and avoiding overcrowding in the emergency care unit, being able to offer a better service to patients.

The main purpose of this project is to aid the optimization of health care resource management by creating and implementing methods and models that estimate the amount of resources required in each targeted department at any given moment. The contributions of this work are twofold; (i) the creation of a Meta-Model, an agnostic data predictor with the ability to self-evaluate its performance, to continuously learn from the data, to be integrated with servers and parallelizable and to transfer models learned in one context to another, i.e., from one hospital to another; (ii) the simulation of the influx and flow in the

Emergency Department generating synthetic data.

The rest of the document is organized as follows. In Section 2, we place the task in the context of previous works reported in Amalfi Analytics as well as in the literature. Then, Section 3 describes the problem as well as its objectives and specifications. In Section 4 we describe the system architecture with the specific aforementioned requirements. In Section 5, we provide experimental details regarding model configuration and we compare our experimental results, and in Section 6, we propose new architectures to improve the model.

# 2 Related work

In the last few years, especially thanks to the recent advancements in the field of Deep Learning, Machine Learning (ML) has drawn a lot of attention. As stated by R. Bonetto and V.Latzko [18], one of the key reasons for the ML hype is that it provides a coherent platform for introducing intelligent decision-making into a variety of domains.

## 2.1 Machine Learning

ML is a vast area of research that is primarily concerned with finding patterns in empirical data. It refers to a system's ability to acquire, and integrate knowledge through large-scale observations efficiently, and to improve, and extend itself by learning new knowledge rather than by being programmed with it [19]. Sometimes after viewing the data, we may be unable to interpret the pattern or extract information from the data. In that case, we apply ML [20].

There are some variations of how to define the types of ML Algorithms but commonly they can be divided, according to their purpose, into the following four categories: Supervised Learning, Unsupervised Learning, Semi-supervised Learning and Reinforcement Learning.

In this project we will focus on Supervised Learning [21], the ML task of learning a function that maps an input to an output based on example input-output pairs. Depending on the target type can be either classification (discrete values) or regression (continuous values).

In Supervised Learning, the input dataset, during the setup, is divided into train and test dataset. The train dataset has an output variable which needs to be predicted or classified. All algorithms learn some kind of patterns from the training dataset and apply them to the test dataset for prediction or classification [22]. During this project, we will mainly use two well-known famous supervised ML algorithms: Decision Trees [23] and Random Forests [24].

On the other hand, ML Algorithms can also be divided, according to their learning scenarios, into: Multi-Task Learning, Active Learning, Online Learning, Transfer Learning, Reinforcement Learning and Ensemble Learning.

In this project we will focus on Transfer Learning, Online Learning and Ensemble Learning.

### 2.1.1 Transfer Learning

Transfer learning (TL) is a machine learning method where a model developed for a task is reused as the starting point for a model on a second task, i.e, it uses a source domain to help a target domain learning [25]. TL makes three basic assumptions: (i) the source domain has a large amount of labeled training data, (ii) the target domain has few or no labeled training data but a large amount of unlabeled data, because of the expense or difficulties of gathering the data or the data being rare or inaccessible, (iii) the source is very similar to the target.

TL leverages the source labeled data to help learning in the target domain. TL only uses the source domain to help the target domain learning, so it is one-directional. Finally, it is important to remark that it does not accumulate the learned knowledge.

### 2.1.2 Online Learning

Online Learning [26] is a machine learning methodology in which data is made accessible in a sequential manner and is used to update the best predictor for future data at each step, as opposed to batch learning approaches that create the best predictor by learning on the full training data set at once. Online learning is a frequent strategy used in areas of machine learning when training across the full dataset would be computationally infeasible, requiring the employment of out-of-core techniques. It is also needed when the algorithm must dynamically adapt to new patterns in the data or when the data itself is created as a function of time.

It is important to note that Online Learning algorithms may be vulnerable to catastrophic interference, an issue that incremental learning techniques can overcome.

### 2.1.3 Ensemble Learning

In statistics and ML, ensemble methods use multiple learning algorithms to obtain better predictive inference—predictive performance than could be obtained from any of the constituent learning algorithms [27]. Sometimes, multiple learning algorithms can be the same algorithm, but implemented in different ways: various sets of instances or attributes, with different parameters, etc.

## 2.2 Meta-Learning

In this project, we propose to use the aforementioned learning techniques along with the previous Supervised ML algorithms to design a predictive and classifying Meta-Model based on Meta-learning.

### 2.2.1 Meta-learning

Meta-learning allows machine learning systems to benefit from their repetitive application. If a learning system fails to perform efficiently, one would expect the learning mechanism itself to adapt in case the same challenge is introduced again [28]. Meta-learning differs from base-learning in the scope of the level of adaptation; while learning at the base-level is concerned with accumulating experience on a single activity (e.g., credit ranking, medical evaluation, mine-rock discrimination, fraud detection, etc.), learning at the meta level is concerned with accumulating experience on the success of various implementations and applications of a learning method. Meta-learning can also be viewed as an important feature of self-adaptive systems [29], that is, learning systems that increase their efficiency through experience.

In a nutshell, meta-learning makes use of the relationship between tasks or domains and learning algorithms. Rather than starting from scratch on each new task, meta-learning allows for the assessment and comparison of learning algorithms on a wide range of previous tasks, identifies benefits and drawbacks, and then recommends the learning algorithm, or combination of algorithms, that maximizes any utility function on the new task.

The efficiency or usefulness of a given learning algorithm is often calculated by a mapping between a characterization of the task and the algorithm's expected performance [30]. In general, meta-learning can recommend more than one algorithm. Typically, the number of suggested algorithms is much lower than the total number of feasible and available algorithms [31].

### 2.2.2 Time series

A time series is a collection of observations of well-defined data items obtained through repeated measurements over time. Thus, time series have patterns that show different seasonal effects and trends, both of which can be additive, multiplicative or non-existent.

The phrase "meta-learning" in the context of time series was first used in [32] and is a recent concept coined by the general machine learning community to describe the method of automatically acquiring knowledge for time series forecasting model collection.

Traditionally, experts visually inspected time series characteristics and fit models according to their judgement. C. Lemke and B. Gabrys in [33] investigated an automatic approach, since a thorough time series analysis by humans is often not feasible in practical applications that process a large number of time series in very limited time. The approach was then extended to determine weights for a combination of individual models based on data characteristics.

Since the 1990s, researchers have been attempting to pick a suitable forecasting model by using characteristics of univariate time series. The first systems were rule-based and built on a combination of judgemental and quantitative approaches. Collopy and Armstrong used time series features to generate 99 rules [34] for weighting four different models; features were obtained by a mix of visual inspection of the time series and domain knowledge. Adya later modified this system and reduced the necessary human input [35]. However, they did not abandon expert intervention completely. Thereafter, Vokurka [36] extracted features automatically to weight between three individual models and combined them using a rule-base that was built automatically. Nevertheless, the latter approach also required manual review of the outputs. Hence, completely automatic systems were first proposed in [37], where a generated rule base selected between six forecasting methods was selected. Since then, many systems have been proposed [33].

### 2.2.3 Self-Adaptive Systems

The ever-increasing sophistication of information programs that must manage their targets 24 hours a day, seven days a week while working under confusion motivates the need to equip systems with structures to handle change and transition while in operation [38].

The use of "internal structures" such as exceptions (as a characteristic of a programming language) and fault-tolerant protocols is a popular approach to dealing with transition. The use of such mechanisms is frequently domain-specific and closely bound to the code. This increases the expense of developing, modifying, and reusing solutions.

Shift, on the other hand, can be managed using "external processes" built

on the idea of a feedback loop. Self-organization is a significant paradigm in this sense, in which relatively basic elements use local laws to change their relationships with other elements in response to changing situations in order to cooperatively realize framework goals [39]. Another important paradigm is control-based adaptation, which uses the mathematical foundations of control theory to construct feedback loop processes and analyze and ensure key properties [40].

In this project, we concentrate on architecture-based adaptation, which is a well-studied and widely used approach to dealing with change [41, 42]. Architecture-driven adaptation is based on a feedback loop that tracks the system and its environment and adapts the system to achieve its goals or gracefully degrade if appropriate. The use of runtime models [42] that allow the system to reason about change and make adaptation decisions is critical in tackling this challenge. The feedback loop isolates adaptation problems in distinct system components that can be analyzed, updated, and replicated through systems.

In this work, we aim to develop a completely automatic Self-Adaptive Meta-Learning system capable of dealing with different input data bases, including time series, using the functional programming paradigm.

## 2.3 Amalfi Analytics

Amalfi Analytics[1] is a spin-off of Univeristat Politècnica de Catalunya based in Barcelona. Its mission is to use Big Data technologies to help the managers of healthcare systems make them more efficient, sustainable, and fair. To achieve that, they have created tools for clinical management decision making.

Amalfi Analytics works with several sectors; for instance, with hospitals, health centers and other health institutions which have workload problems that can collapse the service. They have shown that intervening from a management point of view, several causes of this saturation can be solved. The company states that there are two areas where action can be taken at the management level to reduce the problem almost immediately: waiting times in the Emergency Room (ER) and the risk of return and readmission to hospitals. Amalfi Analytics' team believes that they can reduce waiting times in the ER by identifying in triage patients with higher risk of admission and of premature return to the emergency room and by facilitating the search and allocation of beds in

[1] https://amalfianalytics.com/en/

associated centers in case of transfer. Regarding the risk of return and readmission to hospitals, i.e, the risk of people who have been hospitalized and who have to be readmitted to the hospital in a short period of time, they believe that this risk could be significantly decreased by providing an accurate view by Diagnosis Related Groups -a system to classify hospital cases into one of originally 467 groups, to treat subgroups of patients differently- and by anticipating the risk of return and evaluating the potential impact of preventive actions. They work with some of the most important hospitals of Catalonia such as *Hospital de Sant Pau*, *Hospital del Mar* or *Hospital Vall d'Hebron*.

This project builds on background researched and development activities previously carried out at Amalfi Analytics. Amalfi Analytics' products generate prospective analyses and models to predict and anticipate events using ML and, in addition, the tech team of the company has already modeled the influx of ER. Nonetheless, their approach is to use some simple and individual ML models such as Random Forest [43] or Decision Trees [44] and some Deep Learning algorithms composed by embedded LightGBMs [45] which achieve brilliant results at the cost of many hours of developer. Thus, this project aims to provide better estimations, more flexibility and model adaptation. This will be achieved using a Meta-Model, a more complex ML layer, which will include self-evaluation. This characteristic gives autonomy to the model and hence less developing effort when creating the model. Furthermore, this new functionality will improve the performance of the models as when they do not perform, they will be retrained. This new component will provide a metric to the hospital administration people to find out how much they can trust the results of the model. In addition, the Transfer Learning approach will improve the performance of the model in new or modified environments. For example, it will enhance results in new small hospitals where only little data is available or in hospitals which have changed the registration techniques and there is a lack of data.

This Final Degree Project has been carried out in the context of a formal collaboration agreement between Amalfi Analytics and UPC on topics related to Machine Learning and data analytics.

### 2.3.1 SOIL

Amalfi Analytics works with several hospitals and health centers, each of which has its own data and data storage methods. As a result, they needed a processing

data platform in order to standardize how data scientists can access data from each center. To solve this problem, they created a platform called *SOIL* that abstracts the applications from the specific data storage mechanism, including whether the data is locally or remotely stored. At the moment, the database used by Amalfi behind *SOIL* is Elasticsearch [46].

*SOIL* allows to accelerate the deployment process and to decouple research and algorithms from the product. Thus, when a new client, hospital, arrives with only a few configuration files they can work with their data. In addition, as all the algorithms are generic and adaptable to *SOIL*, Amalfi Analytics does not need a costumed algorithm for each client. On the other hand, the flexibility provided by this customized platform allows Amalfi Analytics tech team to be split into three roles: algorithms, integration and platform development. The algorithm role is composed by data scientists who develop algorithms and do research. The integration role consists on generating configuration files to be able to integrate new hospital's data and architectures to SOIL and therefore use their data. And the platform development role consists on improving the platform itself by parallelizing the way data is uploaded or adding federated learning tools, for example.

*SOIL* is composed of three abstractions: Data Structures, Modules and Scripts.

**Data Structures.** Data structures is the abstraction which contains information about serializing data (the process of translating a data structure or object state into a format that can be stored or transmitted and reconstructed later) and metadata, information about data itself, as for example the name of the columns of a dataset, or the creation date.

**Modules.** Modules are the basic computational unit of Amalfi Analytics and they are composed by the implemented algorithms. These modules are independent on the data, they are data agnostic, and have to do with the data transformation and not with the data itself. Nevertheless, these modules have a precondition: data must be on a specific format, data structure, and a post-condition: data must be also returned in a data structure format. Therefore, modules can be seen as a black box which given some data in a particular format returns some data in another specific form and can be encapsulated which gives the code a lot of versatility.

**Scripts.** Scripts are the ones which define the data transformation pipeline. They take some data and then call the modules, with some specific hyperparameter configuration, which return a data structure that is saved by the scripts to an alias. Hence, these scripts are programmed by the team of integration developers.

There are mainly three kinds of scripts: new data, to incorporate and integrate new data; schedulers, they call training, prediction, evaluation, etc. algorithms; and an extra one to add API services.

Therefore, from a technical point of view, they do not have several products but several algorithms and configurations which can be recombined to generate the desired products.

### 2.3.2 RAIN

*RAIN* is a set of services in charge of calling *SOIL* when needed and receiving data batches from customer systems via secure channels, and storing them in *SOIL*, thereby orchestrating the data pipeline. *RAIN* allows dividing pipelines into data, where new data is uploaded; schedules, where scripts are called; and setup.

# 3 Problem definition, goals and specifications

In this section we define the problem, its main goals and the specification that the solution must meet in order to be used in real environments in hospitals by Amalfi Analytics.

## 3.1 Problem definition

In many cases, hospital resources are mismanaged due to the difficulty of forecasting different services. For example, if at some point it is estimated that there will be few patients in the ED, the hospital administration services will assign there few health professionals and few beds. However, if the prediction is not fulfilled and many patients arrive at the emergency room, this service will collapse and consequently other hospital services will do so following Pareto's Principle [17]. The simplest solution to avoid this problem would be to devote unlimited money and resources to each service, but hospitals' budgets are not infinite, but rather very limited. Hence, this alternative is not plausible and another approach must be found.

Nowadays, hospitals and health centers constantly collect a wide variety of data, such as that collected in ED: arrival time, patient diagnoses, patient waiting times in the emergency room before being attended, percentage of sick healthcare professionals in each department... which is not being used for predictive analysis.

Therefore, we propose to use all these data in order to predict the daily hospital flow which will help to predict the resources needed at any given time as well as improving the supply management. This is not the ultimate goal of this project, but a running example through it to motivate the more technical goals.

## 3.2 Goals

We have seen that, in order to control a hospital, we need to know how to predict several features. Hence, we need a highly adaptable and flexible model that allows us to forecast several aspects whenever clinicians need it.

The main goal of this project is to build a predictive framework to improve the hospital resources management, specifically the allocation of personnel and beds managements, offering a bettering in the quality of care while reducing healthcare costs.

Therefore, we aim to design, evaluate and test an:

- **Adaptable.** We want an adaptable method that observes and learns new modifications from the input and output values, as well as their related characteristics. Furthermore, a model that learns from activities that may change consumer behaviour in real time and, as a result, retains its precision at all times. This functionality is specifically designed to be robust to concept drift changes in dynamic environments.
- **Flexible.** We define a flexible model as one that allows the user to easily incorporate his or her own desired functions and modules or modify and extend the existing ones. Experiments are easy to design, setup, and run.
- **Transferable.** We want our model to be capable of applying information learned when solving one problem to another yet similar problem.
- **Predictive.** We want our model to be a statistical method that attempts to model future events or outcomes by evaluating trends that are likely to predict future outcomes.
- **Incremental.** Learning models are created incrementally and are updated continuously or on demand when new data is available. They are suitable for big data applications where changing environments are present.

Meta-Model using already existing and simple methods to extract data, so that the predictive model can be easily used by Amalfi Analytics in production environments, that is, in healthcare centers.

## 3.3 Specifications

This project must meet certain conceptual and technological specifications listed and explained below.

### 3.3.1 Conceptual requirements

The resulting model must be:

- **F1:** Data agnostic, i.e. a system that operates independently of the exact nature of the data. This is because it should serve several and future applications at Amalfi Analytics.

- **F2:** Flexible, adaptable, well documented and defined so that users can conveniently program and modify their own modules without delving into the full code.
- **F3:** Able to integrate and unify different libraries such as Pytorch [47], Scikit Learn [48] or Scikit-multiflow [49].
- **F4:** Flexible, but we must ensure a certain level of predictability.

### 3.3.2 Technological requirements

The resulting model must be:

- **F5:** Integrated to *SOIL* platform explained in Section 2.3.
- **F6:** Integrated to servers using Dask [50], an open source library for parallel computing written in Python. Therefore it will be Dask parallelizable.
- **F7:** Technically tested using some software such as Pytest [51], a mature full-featured Python testing tool.
- **F8:** Functional. All executions must be lazy evaluation, as they can only be executed as needed.
- **F9:** Automated. It must be self-evaluated by some predefined or user-defined functions and raise alarms when performance is bad.

# 4 Solution

In order to fulfill these requirements, we propose the following Meta-Model[2], a predictive flexible and adaptable framework using already existing and simple methods.

We aim to create an agnostic data model since we want to use the resulting model in development for a variety of reasons, such as forecasting the number of health staff that will be absent from work, the number of beds that will be required on one level, or the number of patients in the emergency room at any given time.

## 4.1 Definition of the Meta-Model

The suggested Meta-Model consists of a mixture of different base models[3] and has a functional architecture to provide the user with optimum flexibility, i.e., the user would be able to choose any of the different models that she/he desires to create the Meta-Model. Furthermore, this model will self-evaluate its output and transfer knowledge learned in one context to another.

The Meta-Model can be mathematically describe as follows:

$$f_1(X) = y_1$$
$$f_2(X) = y_2$$
$$...$$
$$f_n(X) = y_n$$
$$H(\{f_i(X) = y_i\}_{i=1}^{n}, h_\theta)$$

where $f_i$ are the models, $X$ is the matrix of data, $y_i$ is a vector which stores the predictions of each model, $H$ represents the Meta-Model, the combination of all the base models, and $h_\theta$ is the set of functions used to combine all the models in order to create the Meta-Model.

In order to create a Meta-Model, first of all, the meta-learning algorithm has to train each individual model using a training data set. While doing so, it obtains a set of features such as the performance of each one. Here, it is important to know that even though all models receive the same data set, they

[2]The code of the implemented model can be found in https://github.com/AmalfiAnalyticsOrg/Meta-Model

[3]By base model, we mean the models that are included in the meta model; they can be any model, such as a Decision Tree, a Random Forest, a Linear Discriminant Classifier, or even a Meta-Model.

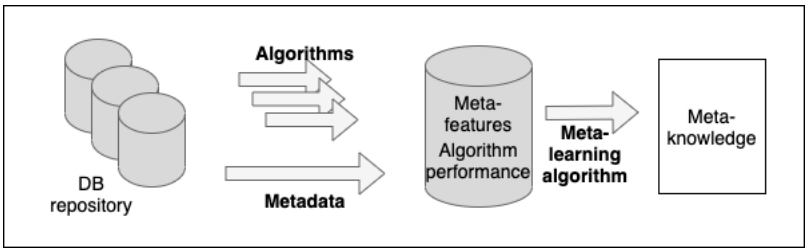


Figure 2: Meta-Model Pipeline.

can use different subsets of it. Secondly, the meta-learning algorithm optimizes a subset of the so-called Meta-Model hyperparameters, that in addition to the combination policy ($h_\theta$), will allow the Meta-Model to stack the predictions of each model along with its performance to feed another predictor, which can be, usually, some lineal combination, for instance, to obtain the Meta-Model predictions themselves.

To have a better understanding of the whole training and merging process, the Meta-Model pipeline is depicted in Figure 2.

#### 4.1.1 $f_i$ models

The aforementioned $f_i$ algorithms may be of any kind, like ML models from Sklearn [48] such as Decision Trees or Random Forests, Scikit multiflow [49], such as Hoeffding Trees, Pytorch [47], or any statistical library containing linear models, for instance.

By default, the architecture includes the following models:

- **Decision Tree.** One of the statistical modeling methods used in analytics, data processing, and machine learning is the decision tree. They are built using an algorithmic method that defines multiple ways to divide a data set depending on various conditions [23]. It is one of the most commonly adopted and functional supervised learning processes. Decision Trees are a non-parametric supervised learning approach that can be used for classification as well as regression tasks.

- **Random Forest.** Random Forest is a powerful machine learning algorithm that can be used for a wide range of tasks such as regression and classification. It is an ensemble system, which means that a random forest model is composed of a large number of small decision trees, known as estimators, each of which produces its own predictions [24].

- **Linear Discriminant Analysis.** The Linear Discriminant Analysis [52] is a generalization of Fisher's linear discriminant, a technique used in statistics and other fields to locate a linear combination of features that distinguishes or characterizes two or more types of objects or occurrences.
- **Hoeffding Tree Classifier.** A Hoeffding Tree [53] is an incremental, anytime decision tree induction algorithm that can learn from large data streams if the distribution producing examples does not change over time.

Nonetheless, one of the architecture's key requirements is that it must be very scalable and well documented so that users can add their very own modules and models. The above methods are the ones used in this particular thesis, however, it is not an exhaustive list of models supported by the meta-learning algorithm. The Meta-Model uses the class architecture of the library agnostically to what particular model algorithm is implemented. Therefore, when the core Meta-Model algorithm "knows" the architecture of a Library, it can use any other model with the same architecture, or at least that mimics it; allowing to easily switch models from the same library and allowing developers to use their algorithms by just adding a wrapper class of the supported library they prefer.

### 4.1.2 $h_\theta$ functions

The $h_\theta$ functions is a set of mathematical operations that defines how all of the base models are combined to form the Meta-Model. These functions, like those seen in the models, can be easily described by the user from scratch. Moreover they can be as general as desired.

Nonetheless, here we introduced two algorithms to assign importance weights to each base model, so that we can combine all the models' predictions in a non trivial manner.

The first technique called *accuracy weights* calculates the accuracy of each model and then assigns a weight to each model based on its accuracy as follows:

$$weight_i = \frac{accuracy_i}{\sum_{j=1}^{N} accuracy_j} \tag{1}$$

The second technique called *SVM weights* calculates the predictions of each model and uses it along data to fit a Support Vector Machine (SVM) [54]. After that, we assign the class weights provided by the SVM model [48] to each model and we use it for combining all of the models.

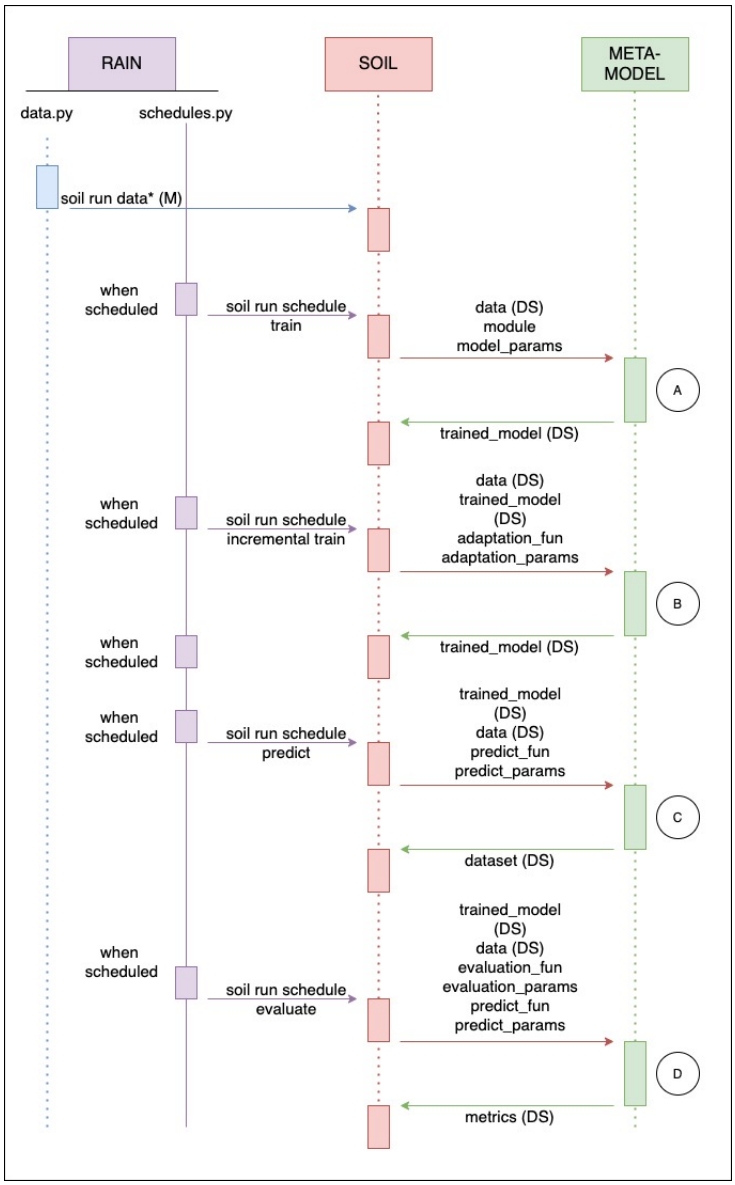


Figure 3: Meta-Model architecture integrated to SOIL and RAIN. A, B, C and D correspond to the train, incremental train, predict and evaluate model algorithms, respectively.

## 4.2 Architecture

The model architecture is fully integrated to *SOIL* and *RAIN*, as can be seen in Figure 3, and composed basically of:

- Four *SOIL* Scripts: *train, incremental-train, predict* and *evaluate-model*, which define the data transformation pipeline.
- A set of core modules, the basic computational unit of Amalfi Analytics which contain the algorithms and transformations themselves.
- *SOIL* data structures: *MetaModel, DataStructure, SKLearnModel, Predictions...*.

The relation of all these components has been depicted in Annex A.

All the modules are implemented as higher-order functions[4], a tool that allows to transform any function such as a ML algorithm to a SOIL module allowing it to read SOIL data structures and return everything in a SOIL data structure format too.

Then we continue by exposing the different scripts technically, paying special attention to the architecture and the required modules.

### 4.2.1 Train

Train is the script in charge of training the base models and optimizing the hyper-parameters of the Meta-Model.

**Training data.** This script reads a soil data structure where training data is available. This data structure, as we have previously mentioned, is composed by the data itself and the metadata, data which stores information about data. The later includes the list of variable names that will be used as parameters to train the models and the ids of the variables (independent predictive variables and the target) of the training set.

The *train script* calls three different *@modulify* modules: *train_model, assembling* and *training*.

- ***Train_model.*** This module is in charge of instantiating base models based on their constructor, hyper parameters, and ID. It will not train

[4]Using the @modulify decorator

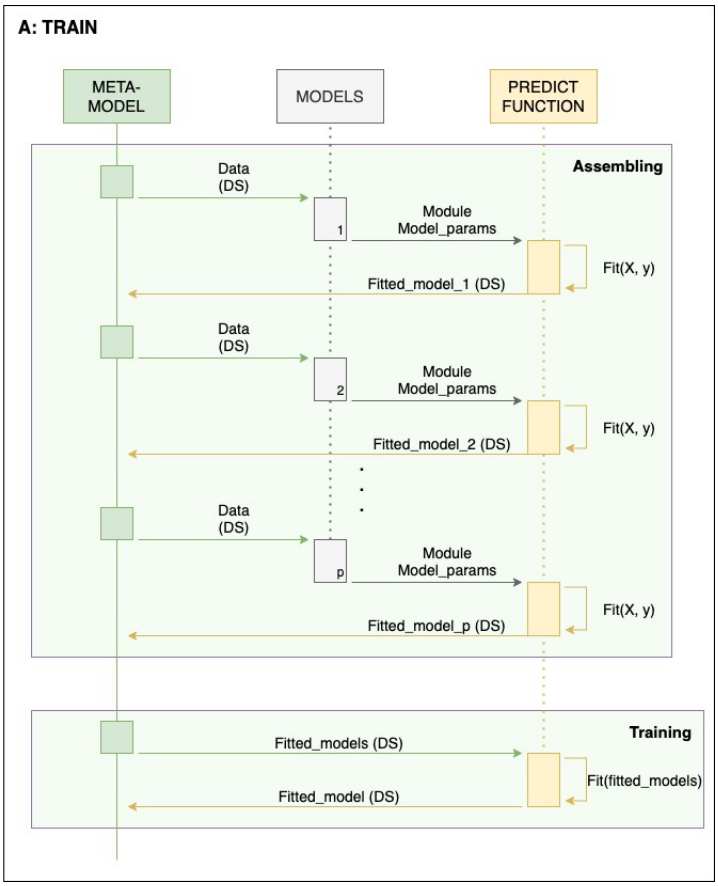


Figure 4: Training architecture. Assembling module where base models are trained in the fit function. Training module where the Meta-Model is trained, i.e, where weights to base models are assigned in order to create the Meta-Model.

the base models until data is available and provided to it, or until it is needed because they will be used. In order to obtain this lazy specificity, this module is decorated with a higher-order function: *trainable*, which allows the module to take one or more functions as arguments, returns a function as its result and unifies the way the trained modules are called in the API. This decorator allows the user to specify the common things that he/she is interested in doing in every training such as: the train-test split or cross-validation. Furthermore, this tool allows the user to access the model instance but not the trained model, which will not be created until data is available.

- ***Assembling.*** This module receives the base model instances, the training parameters, functions used to recombine the base models, training data and returns the Meta-Model as a whole, with the trained base models.
- ***Training.*** This later module receives the *Assembling*'s output, the data and the training parameters and returns the Meta-Model with its weights and specifications in a Soil Data Structure (SOIL DS).

In Figure 4, we depicted the architecture of the train script in more detail as well as the relationships between the aforementioned modules.

### 4.2.2 Incremental train

Incremental train is a script that allows users to retrain models using existing models instead of starting from scratch. It is shown that incremental learning techniques can overcome catastrophic interference of Online Learning and that is why we are really interested on this particular functionality. As a result, after receiving the trained Meta-Model, an adaptation function and training results, this model returns a retrained Meta-Model.

The strategies that have been adopted describe the approach that will be used to retrain the model. As a result, by programming them, one can use various adaptation features. There are only two key requirements that must be met in order for these functions to be generated. On the one side, we must ensure that they obtain the data as well as the templates in DS format. These functions, on the other hand, must return a Meta-Model in SOIL DS format.

In the current library three policies have been implemented:

- **Kick Out Older.** This policy essentially finds the Meta-Model's oldest base model using the *getOlder* module, removes it with the *removeModel* module and replaces it with the same model but trained with the most recent available data using the aforementioned *train model* module and adds it to the current Meta-Model using the *addModel* module.
- **Kick Out Worst.** This policy basically finds the Meta-Model's worst base model using the *getWorst* module and given a metric such as accuracy or R-squared, and substitutes it by retraining it with the most recent available data using *removeModel*, *train_model* and *addModel* modules.
- **Retraining Weights.** This strategy takes the weights allocated to each base model with the *getting_ids* module and retrains them using the previously described $h_\theta$ functions making predictions and getting evaluations

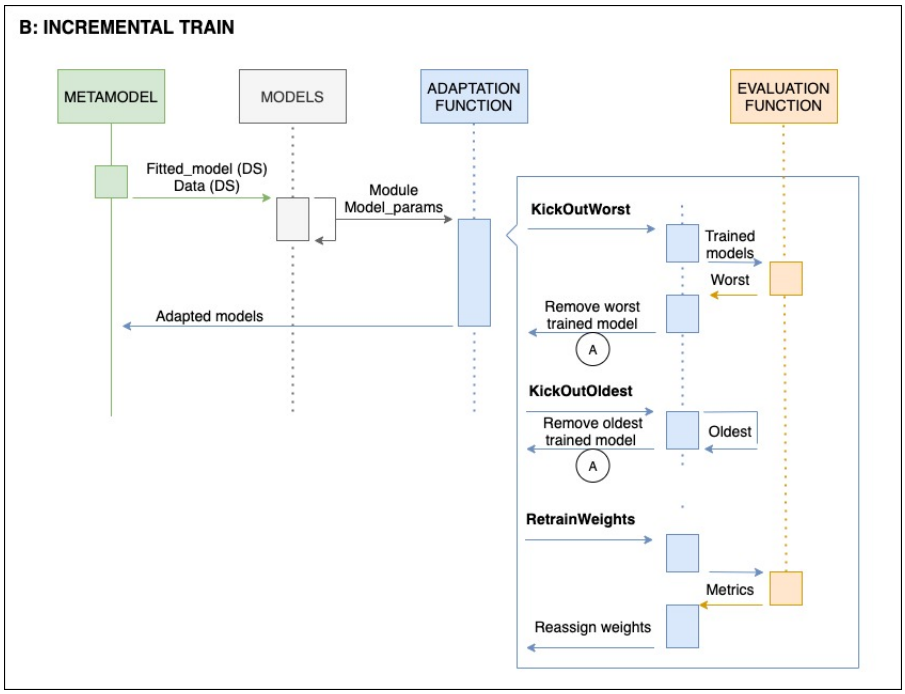


Figure 5: Incremental train architecture.

with their correspondent modules. As a result, this approach does not retrain any base model but rather the weights of the Meta-Model, yielding a new Meta-Model.

In Figure 5 the incremental train architecture is depicted.

### 4.2.3 Predict

Predict is the script in charge of making predictions either classification or regression which architecture is shown in Figure 6.

This script receives an unlabeled SOIL DS and the trained Meta-Model and returns the predictions in a SOIL DS. In order to do so, it invokes the *make predictions* module which makes the predictions for each model and then the predictions are merged using the combination strategy and the Meta-Model hyper-parameter to provide a unique output for each observation. We should note that the *make predictions* module is designed recursively since some base models which form the Meta-Model can also be of Meta-Model kind, and for these types of models, a pool of predictions and recombining techniques need to

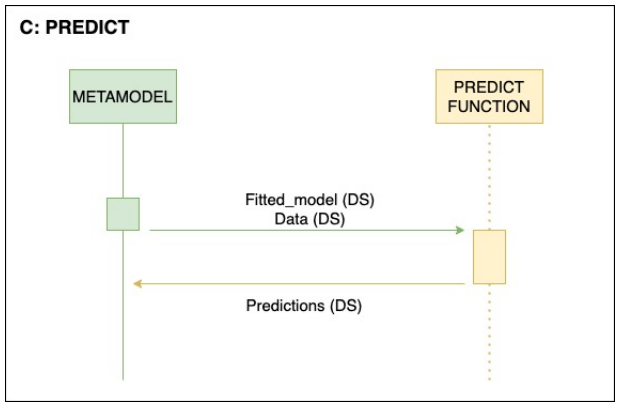


Figure 6: Predict Architecture.

be performed.

#### 4.2.4 Model Evaluation

This script evaluates the performance of the Meta-Model given an evaluation function.

To evaluate the model, we identified various horizons where we might be interested in evaluating the model. For example, if we were focusing on forecasting the influx of a hospital's ED over the next week, we would like to know how well our model has been doing recently. As a result, we would like to test our model using the most recent collection of data, say, the last week. On the other hand, if we were to predict the ED influx in a year's time, we would need to know the model's average output performance.

Therefore, this script has four main modules:

- **Get Retrospective Data.** Provided some historical data and a date, this module returns the collection of historical data from the date until the present in an SOIL DS.
- **Get Random Data.** When given any data and the desired sample size, this module returns a collection of data of the desired size in a SOIL DS.
- **Make Predictions.** This is the module that was defined in the Predict script.
- **Get Evaluation.** Given the predictions, the historical set of data that we want to use and an evaluation function that can be *accuracy score*, *pre-*

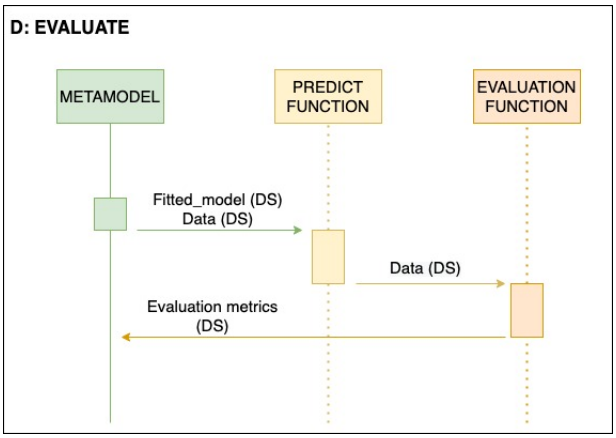


Figure 7: Model Evaluation Architecture.

*cision score* or *R-squared*, for instance, it returns the evaluation function score.

Hence, as depicted in Figure 7, given the predictions, the historical set of data that we want to use and any evaluation function, it returns the evaluation function score.

## 4.3 Meta-Model parallelization

The proposed Meta-Model has been integrated in SOIL which is in turn implemented with Dask [50]. Dask is a dependency processor that produces a dependency graph given all of the functions and their inputs and outputs. So, given a set of functions with interdependences, the system sorts and processes them as required. At every time step, it processes all the functions which have either no dependency or all dependencies solved. For instance, if a function (function A) receives the output of another function (function B), function B will be run before function A and the later will not be processed until the first is finished. As an example, in Figure 8 the graph dependencies generated during the taining process performed by Dask is shown.

As a result, the process in our model is automatically parallelized whenever we need to train or evaluate the base models, as well as when we need the stacked predictions of the base models to feed the predictor. Therefore, as shown in Figure 9, assembling and training modules are conducted in parallel during the Meta-Model training process, reducing runtime.

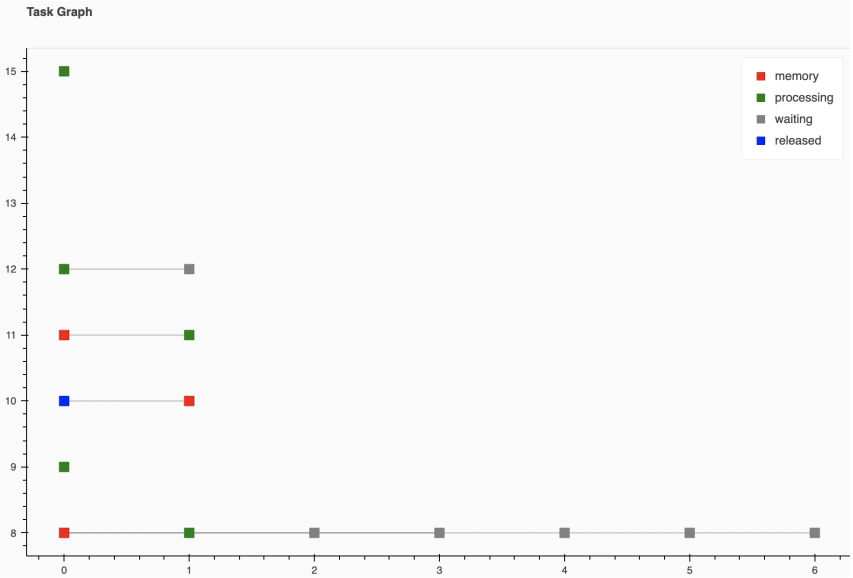


Figure 8: Task graph.

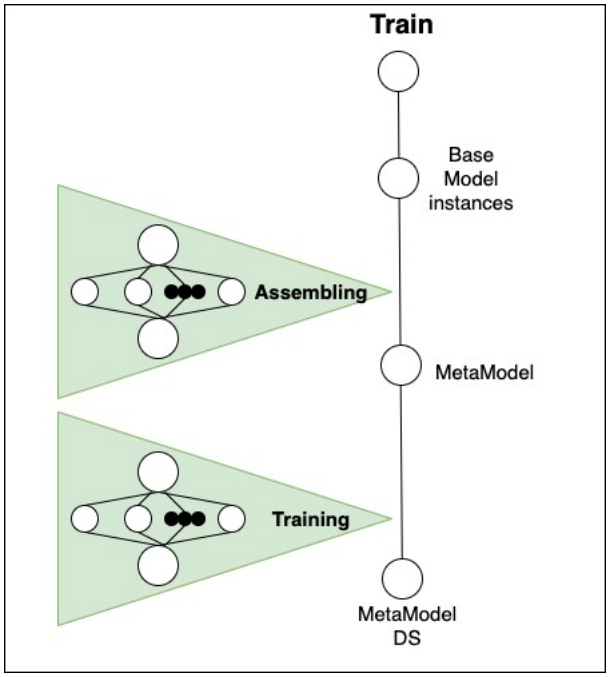


Figure 9: Parallel training scheme.

# 5 Experiments and results

In this section, we present the experiments we conducted with the designed Meta-Model, their correspondent results, as well as, the setup and data sets utilized to carry them out to demonstrate its features.

## 5.1 Data

Several datasets were used in this project to validate and test the presented model, as well as to prove the model's agnosticity. Below, we proceed to present and go through all of them in more detail.

- **Iris dataset.** This is perhaps the most well-known database in the pattern recognition literature. It is a multivariate data set introduced by the British statistician, eugenicist, and biologist Ronald Fisher in his 1936 paper *The use of multiple measurements in taxonomic problems* [55] as an example of linear discriminant analysis. The dataset is divided into three classes, each with 50 cases, and each class represents a different species of iris plant.

- **MIMIC-III dataset.** MIMIC-III ('Medical Information Mart for Intensive Care') is a massive, publicly accessible database that contains deidentified health-related data from over 40,000 patients who were admitted to the Beth Israel Deaconess Medical Center's critical care units between 2001 and 2012 [56]. Demographics, vital sign assessments taken at the bedside (1 data point per hour), diagnostic test findings, treatments, prescriptions, caregiver observations, pathology files, and mortality are all included in the archive (including post-hospital discharge).
  MIMIC promotes a wide variety of analytic research, including epidemiology, clinical decision-rule change, and the advancement of electronic tools. It is noteworthy for three reasons: it is widely accessible to researchers all around the world; it has a varied and vast population of ICU patients; and it provides extremely granular data such as vital signs, test outcomes, and prescriptions.

- **Cyclic synthetic datasets.** During the course of this project we have created some cyclic random generators presented in Appendix B which have been used to create this dataset. Specifically these datasets have been generated by the *2 classes generator* with different parameters. They

consist on two explanatory variables and a binary target.
In addition, we have created a dataset containing two binary explanatory variables and a Hidden State. When we are in the first state $\lambda_1$ the two explanatory variables are combined with an *AND* logic gate to generate our target, while state $\lambda_2$ uses an *OR* logic gate; and, in both cases, a normal noise with standard deviation of 0.1 has been added.

- **CMBD.** CMBD[5] means *conjunt mínim bàsic de dades* and it is a population registry, a minimum basic data set, that gathers information on pathology treated in Catalonia's health centers, which allows us to know the evolution of the pathology over time, the characteristics of the care provided and the distribution of activity in the territory; and is useful for planning, resource evaluation, and purchasing services.
  The data collected by the CMBD is organized into a set of common variables as well as others that are more specific:

  - Variables that identify the patient: personal identification code (CIP), date of birth, sex, residence and medical history.
  - Variables related to the process: service provider unit (UP), type of activity, economic regime, circumstance of admission and circumstance of affiliation, date of admission and date of affiliation.
  - Clinical variables: diagnoses and procedures coded according to the classification used in each of the areas and providers.
  - Specific variables of each system, such as those of the CMBD-RSS, which allow the grouping of patient assessments in the RUG system (resource use groups).

  The data used in this project comes from an undisclosed hospital with which Amalfi Analytics works. This dataset contains 213.476 episodes, i.e, observations. Amalfi Analytics may utilize this data in line with the terms of the agreement with the hospital. The hospital's ability to access the data it has for a product that predicts flows and patients features is what is gathered in the GDPR and is detailed in Annex E.

[5] https://catsalut.gencat.cat/ca/proveidors-professionals/registres-catalegs/registres/cmbd/index.html

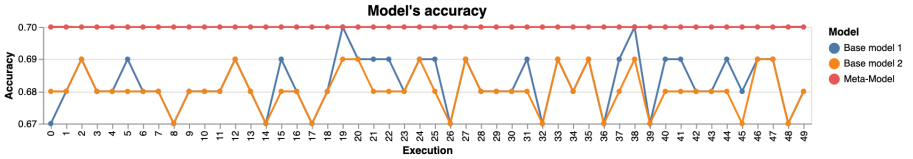


Figure 10: Accuracy obtained predicting whether a patient will die or not in the Intensive Care Unit taking into account its gender and diagnosis and using a batch of 8K observations to train each time. The base model 1 corresponds to a Decision Tree, the second one to a Random Forest and the Meta-Model has been obtained using the *accuracy weights* function.

## 5.2 Experimental setup

Our final models have been trained on a VPS, a virtual private server[6] with 1 CPU and 8GB of RAM.

## 5.3 Base models' accuracies vs. Meta-Model accuracy

In this section, we show that the performance of the Meta-Model is always at least as good as the best of the base models.

To do so, we performed a classification task with the MIMIC-III dataset [56]. Particularly, we tried to predict whether a patient will die or not in the Intensive Care Unit taking into account its gender and diagnosis. We generated a Meta-Model composed by a Decision Tree [23], *base model 1*, and a Random Forest [43], *base model 2*, and we trained and evaluated 50 iterations where each batch contains 8K different observations. For all the iterations we stored the performance of each base model and the overall performance of the ensemble rounded to the second decimal place. As can be seen in Figure 10, unlike the Meta-Model that always achieves an accuracy of 0.7, both base models reach an accuracy close but below 0.70.

Even though we are aware that it is a toy example, and a particular task, we empirically show that the Meta-Model always achieves higher accuracy in comparison to base models in each execution.

## 5.4 Model agnosticity with respect to data

We conducted experiments with various datasets containing varying types of data to demonstrate that the Meta-Model's implementation is agnostic to the

[6] A VPS is a virtual operating system that resides within a parent server and uses virtualization technology to provide dedicated (private) resources to other virtual servers.

| | **Variable 1** | **Variable 2** | **Target** | **Kind** |
|---|---|---|---|---|
| **Experiment 1** | gender | diagnosis | death | classification |
| **Experiment 2** | gender | diagnosis | length of stay | regression |

Table 1: Using the same dataset with different purposes by just changing the metadata.

type of data. By simply modifying configuration files, we may use the same dataset to perform different sorts of tasks, for instance swapping the target, or explanatory, variables.

As an example, we used again the MIMIC-III [56] dataset. In this case, we performed two different tasks just changing the configuration files. First, we performed a regression task where we used *gender* and *diagnosis* variables to predict the *length of stay* in days. Secondly, we using the same set of explanatory variables we did a binary classification, to predict mortality. Configurations for both experiments are shown in Table 1.

Apart from this database, we have also tested the architecture with others datasets to perform multi-label classification tasks, such as the "species" prediction of the Iris dataset [55], and time-series forecasting, as for example the influx at the ED of a hospital using its CMBD database.

## 5.5 Machine Learning in the Wild

In this section, we show that the Meta-Model hyperparameters can adapt to the changes in the incoming data without the intervention of a data scientist.

As a proof-of-concept, we generated a data set stream with two explanatory variables, a target variable, and a hidden variable. When the stream is set to hidden variable 1, the explanatory variables are combined using an AND logical gate, whereas the hidden variable is set to 2, an OR logical gate is used. In both cases, we then added a Normal noise with a standard deviation of 0.1. Each time we generate a new data batch of 100 observations, the stream randomly selects a hidden state each with a 50% probability.

Then, we trained an ensemble of experts. That is, one base model is only trained with data with the hidden variable set to 1, and the other is only trained with the hidden variable set to 2. Both base models are Decision Trees [23] with the same set of hyperparameters. We did not perform any additional training on the base models.

The idea now is that the Meta-Model, by its own, should adapt to changes in

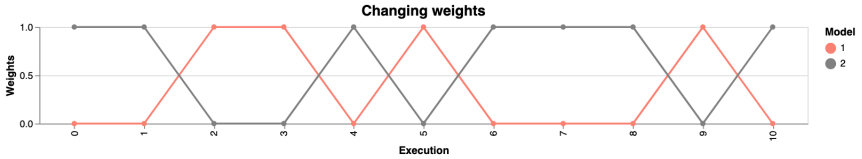


Figure 11: Meta-Model weights assigned to each base model (1 and 2) while changing the data randomly in each run.


Figure 12: Meta-Model weights assigned to each base model (1 and 2) while changing the data gradually from no data generated with the hidden variable set to 1 until all data generated with this hidden variable value.

the data by increasing the relative weight of the current hidden variable expert. And as we expected, and as shown in Figure 11, every time the data distribution changes the base model which has a higher importance in the Meta-Model also changes.

Therefore, in this case it seems better to have one base model expert in each data distribution, i.e, a base model trained in each data distribution, and perform an incremental train to adjust the weight assigned by the Meta-Model to each base model every time new data is arrived than retraining from scratch at each iteration since the Meta-Model is capable of adapting to the changes in the incoming data. However, note that in this specific experiment, each hidden variable changes radically the behaviour of the data. In real examples, however this is unlikely to happen, and most probably than not, the batches would present mixed distributions. In such cases, we could not expect having such a clear weight shifting, and maybe other adaptation strategies must be adopted.

For example, if we progressively altered the data, the weights assigned to each model would likewise vary gradually, but not as drastically as we have seen previously. As an example, we have proven this fact by taking a batch of 100

observations created with the hidden variable set to 2 and changing it gradually until we had all of the batch generated with the hidden variable set to 1. As seen in Figure 12, the weights allocated to each model vary gradually, according to the data, as expected.

In the future, it would be a good idea to repeat this experiment with many recommended strategies, such as the *kick out oldest* one.

## 5.6 Transfer learning

In this part we prove that Transfer Learning can be integrated and used by the Meta-Model to obtain better results. Here, we suggest reusing the features of a pre-trained base model for a specific job on a new issue, while using the information learned from the first task to increase generalization on the other base model. Particularly, this experiment has been carried on using the CMBD dataset of the undisclosed hospital.

We constructed a Meta-Model composed of two basic models, each with its own metadata and training variables. The first model is used to forecast the inflow in the ED at the undisclosed hospital, while the second one predicts the occupancy of the emergency room, i.e, the number of people in the waiting room, at the same hospital, using the first model's generated features as input to enhance its findings.

First of all, the Meta-Model is trained in two non-dependent phases. In one of the rounds, we train the models with influx prediction as its assigned task. The other training round is performed with the models with occupancy prediction as its goal. It is straightforward to predict influx, but, to get an occupancy prediction, the Meta-Model must carry out two sequential phases. The first one uses the influx predictors. Then, it continues with the next round of predictions for the occupation estimators, that uses the predictions of the former step as an explanatory variable.

After exploring the CMBD dataset, we decided to extract a time series of the rate of patients who arrive at the ED per hour. As explanatory variables, we specifically took the date of admission and transformed it into three variables: day of the week, the hour of the day, and month since as shown in Figure 13 and as we know, these can explain the ED arrivals behavior correctly enough, without increasing the complexity of the task. On the other hand, in order

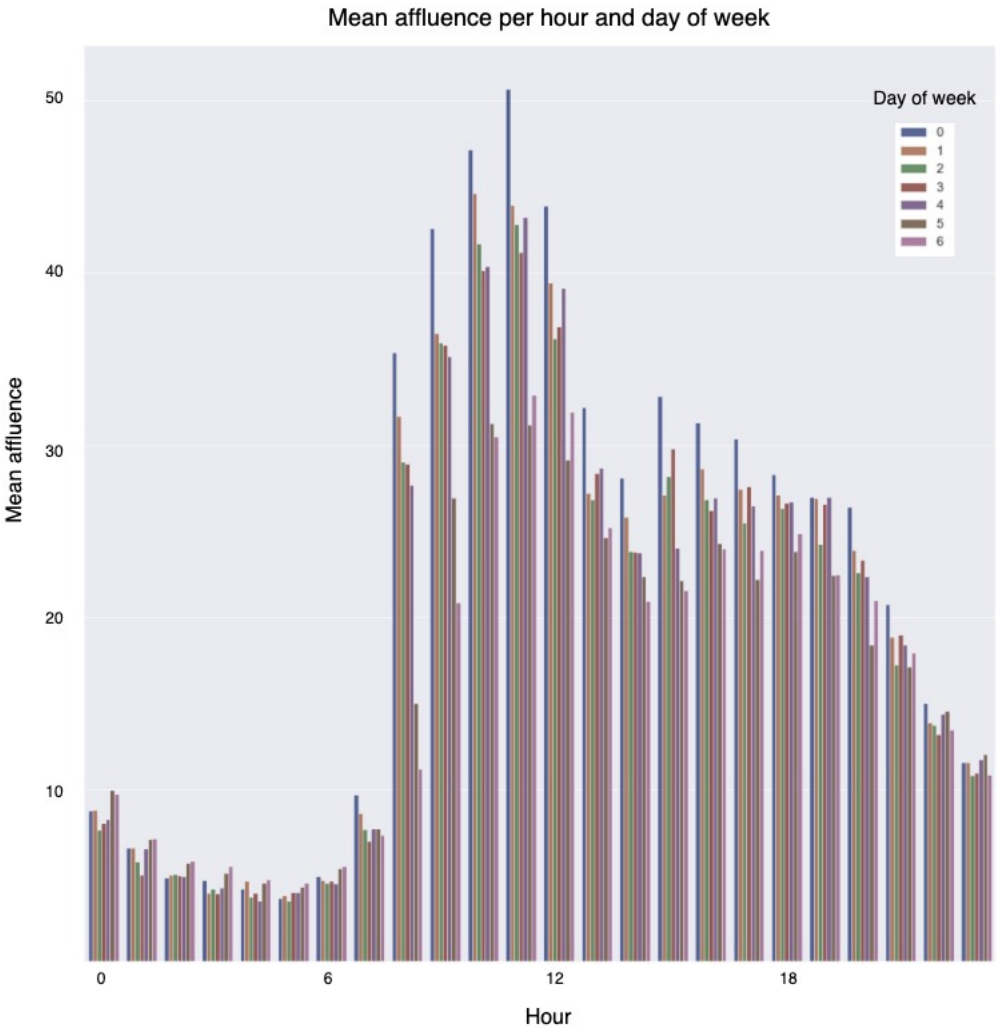


Figure 13: Mean affluence in the ED per hour and day of week.

to predict the occupancy we decided to use the same set of variables plus the predicted influx, ED arrivals, as we think it can improve the forecasting.

The method is depicted in Figure 14 to provide a better comprehension of the proposed pipeline.

To determine the impact of the Transfer Learning approach, we have trained two different Meta-Models both with 130K observations. The first Meta-Model is composed of a Random Forest and a Decision Tree where both of them predict

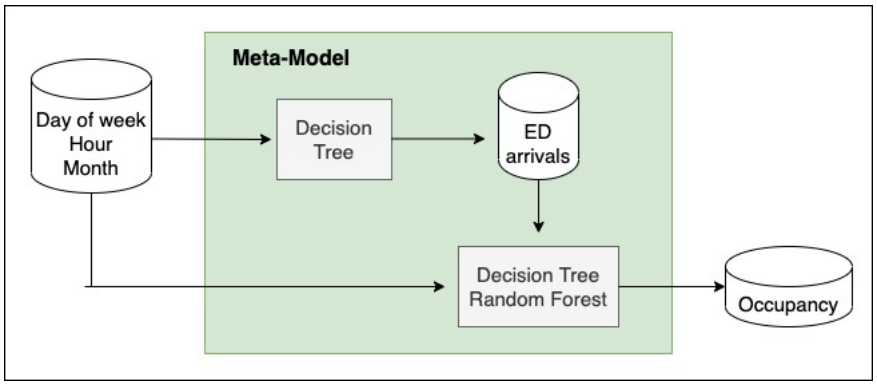


Figure 14: Transfer Learning pipeline to predict the occupancy.

| | **R-squared** | **Explained variance** | **Negative MSE** | **Negative MAE** |
|---|---|---|---|---|
| **MM1** | 0.75 | 0.76 | -18.31 | -2.6 |
| **MM2** | 0.87 | 0.86 | -14.24 | -3.03 |
| **DT** | 0.55 | 0.57 | -24.57 | -3.36 |
| **RF** | 0.56 | 0.57 | -24.53 | -3.38 |

Table 2: Evaluation metrics for different models predicting the occupancy. MM1 corresponds to the first Meta-Model explained in this section, MM2 to the second one, DT to a Decision Tree model and RF to a Random Forest.

the occupancy without using predicted influx as an explanatory variable and it reaches a **R-squared score of 0.75**. On the other hand, the second Meta-Model is composed of a Decision Tree, that predicts the influx, plus a Random Forest and a Decision Tree that predict the occupancy taking into account the features generated by the first Decision Tree. It achieves a **R-squared score of 0.87**, a better metric than the one provided by the first model. More evaluating metrics are presented in Table 2 from where we can deduce that better evaluations are gained with the second Meta-Model since the overall obtained metrics are higher.

As a result, we have illustrated that the Meta-Model architecture can support Transfer Learning methods, and that utilizing them in particular tasks, such as occupancy prediction, can enhance the predictions produced, i.e, the Meta-Model performance.

# 6 Future work

The goal of this project is to provide a proof-of-concept implementation of a data agnostic adaptable Meta-Model applied to the healthcare system.

We have tested our model on several prediction tasks on quite different datasets. This fact shows that the proposed model could predict any variable from any dataset, even from evolving data distributions, without data scientist oversight, achieving promising results while being parallelizable and combining several Machine Learning libraries at the same time that it is well documented and defined so that users can conveniently program and modify their own modules without delving into the full code. Hence, it is ready for other developers to use. However, there are several limitations that need to be addressed in the future such as the integration of new libraries or the study of different adaptations policies.

Future work concerns deeper analysis of particular mechanisms, new proposals to try different methods, or simply curiosity. There are some ideas that we would have liked to try during the description and the development of the architecture but have been left for the future:

- Determine the weights of the various base models in different ways.
- Compare the incremental train's distinct policies to determine which is better and under which circumstances one or the other is preferable. For example, some may be better for gradual change (say, the population assisted by the hospital ages over time) and some for sudden change (say, the sudden outbreak of a pandemic such as COVID-19).
- Integrate and unify some extra Machine Learning interesting libraries such as Pytorch [47] to add neuronal network models.
- During the course of this project, Amalfi Analytics has developed a mechanism to provide alerts to users and developers on the back-ends of their products. We could use them to alert the data scientist when self-evaluated performance begins to decline, or even to notify the users about the reliability of the predictions.
- Integrate more classical Transfer Learning tools so that we could transfer a pre-trained Meta-Model from one hospital to another when data from the second one is poor or not available.

- Study the models' scalability. During the experiments we have seen that while model training is fast, moving data is very expensive in terms of time. Therefore, it is convenient to optimize data transferring to speed up the process.

- Perform an incremental train analysis. That is, testing different strategies to update the meta-model at each time. For instance, it could help to determine in which cases it is better to retrain the whole Meta-Model from scratch and in which ones it is better to only retrain the weights of the incremental train policy and/or some specific base model.

- Analyze different meta-model configurations. Particularly, we could study for which context committees of experts would be more appropriate than a set of generalists (our default mode). As we seen in *Machine Learning in the Wild* experiment a committee of experts is very convenient when the dataset has clear differentiated data distributions. But it is not trivial to know how this approaches would differ in other contexts, such as the influx prediction. We could even explore hybrid strategies.

Still, this project has shown that there is opportunity to use this architecture in real environments such as hospitals, where its application might specifically help to resource management.

# 7 Conclusions

In this project we propose a Data Agnostic Adaptable Meta-Model, a predictive flexible and adaptable framework using already existing and simple methods to aid the optimization of health care resource management, to help the hospital management to assign the number of professionals and resources needed in the services in order to minimize queues and avoid overcrowding.

In this work, we conducted several simple experiments, primarily proof-of-concept experiments, that assisted us in determining whether the model had or did not have the expected behaviour.

Based on the outcome of the tests, and as expected, the Meta-Model always provides better performance metrics than base models. During the occupancy task, base models such Decision Trees and Random Forests only achieves a 0.57 of explained variance while the Meta-Model can obtain an explained variance of 0.76 without Transfer Learning techniques and 0.86 with the latter approach. In addition, we have seen that the Meta-Model is capable of detecting changing distributions and adapting to them by simply running an incremental train step without the guidance of a data scientist.

Therefore, we conclude that the proposed design seems to be promising and can be put into production. It can predict any variable from any dataset, even from evolving data distributions, without data scientist oversight, achieving promising results while being parallelizable and combining several Machine Learning libraries at the same time that it is well documented and defined so that users can conveniently program and modify their own modules without delving into the full code. Hence, we have designed a very useful architecture in varying hospital environments where its application could help to the resources management, offering a bettering in the quality of care while reducing healthcare costs.

## 7.1 Achieved objectives

Previously in this work, in Section 3.3, we have listed and explained the conceptual and technological specifications that the project must meet. In this section, we are going to evaluate whether we have or we have not achieved all of them.

Regarding the conceptual requirements, we have achieved the following features for our Meta-Model:

- **F1:** Agnosticity of data, the system can operate independently of the

exact nature of the data.

- **F2:** Flexibility and adaptability of the model so that users can conveniently program and modify their own modules without delving into the full code.
- **F3:** Ability to integrate and unify different libraries.
- **F4:** Flexibility, ensuring a certain level of predictability.

In terms of technological requirements, we have also met the following listed criteria:

- **F5 & F6:** Integration of the Meta-Model to *SOIL*,*RAIN* and servers where it is parallelizable.
- **F8:** Functional. All executions are lazy evaluation, as they are only executed when needed.
- **F9:** Automated. It can self-evaluate itself by some predefined or user-defined functions.

Hence, future opportunities for development include:

- Integration of the Pytorch [47] library (F3).
- Verification with Pytest [51] (F7).
- Rasing alarms for bad performance notifications (F9). While the model evaluates itself, when poor performance occurs, it does not generate alarms to notify about it to data scientists or to health care personnel since the creation of this tool by the company was extremely close to the project deadline and we were not able to include it in time.

Therefore, we have been able to create a flexible black box to combine the base models in such a way that the user does not need to go deep on the code neither understand *SOIL* or the architecture to create her or his own methods and modules with her or his desired specifications, recombination functions, evaluation functions, base models, etc.

Furthermore, because of its flexibility and adaptability, the Meta-Model protects data scientists against bugs and saves them development time.

## 7.2 Personal conclusions

From a more personal standpoint, being a part of the development of this project with an experienced team and the resources and tools provided by working in a company in the technology and health sectors has been really enriching and rewarding. Furthermore, the fact that the project will have an influence on the company's daily operations, since the generated model will be utilized in production to estimate the influxes and occupations of the hospital emergency department, has provided me with additional incentive and motivation.

This project has given me a better understanding of the technologies that are now being used in the corporate sector, as well as how difficult it can be to produce a project of this type. Prior to developing it, I had never had the task of creating a large-scale architecture in a non academic project.

I would also want to point out that integrating Machine Learning in a real-world scenario with real-world context and constraints is tough and challenging, especially when raw data is not clear, often incorrect and arrives when possible, when hospitals send them which is extremely unpredictable. The last point is critical since our predictors are not fixed but rather evolve with and rely on data.

Several obstacles emerged throughout the course of this project's development. Everything focuses around learning how to load data into ElasticSearch or how to retrieve it using SOIL, the company's own library. This library is surprisingly powerful and enables for a variety of processes to be carried out, such as parallelizing tasks easily. Nevertheless, it is in construction and still insufficiently documented, making the effort of implementing the whole MetaModel architecture inside it considerably more complicated and time-consuming than usual. Because of this lack of documentation, I had to check the source code several times to understand how to use the different methods. Conversely, I was able to provide the developers at Amalfi a lot of feedback about needs for documentation and even poorly designed features of SOIL.

Despite the obstacles, I have been able to address complicated challenges as a result of the creation of this project, which has improved my capacity for abstraction and problem-solving abilities. The research taught me that sometimes, in a corporate environment, the most challenging part of an algorithm is not the algorithm itself, but how to implement it within specific architectures; that is, joint implementation design is just as essential as the implementation itself. Furthermore, this project has compelled me to learn about technologies

used worldwide, such as Elasticsearch or working with servers and some specific and unknown technologies as the company's own library, for instance.

In addition, during the course of the project, we have written an article about the proposed Meta-Model and sent it to the 23rd International Conference of the Catalan Association for Artificial Intelligence[7].

The creation of this project has also shown me that there are many different things that can be done in the field of health using ML technology and tools. In particular, I have seen that, aside from being used in the world of health for medical research, the application of these methods and technologies can also be used to greatly improve the management of the health system and its public resources, resulting in better health care attention at a lower cost.

Finally, as a personal reflection, I believe that technology is one of the most important tools for creating a more sustainable and egalitarian future for all, but this power comes with a great deal of responsibility, particularly when dealing with personal data, and I believe we should be very aware of this fact.

[7] https://ccia2021.udl.cat/en/english/

# References


[1] Travis B. Murdoch and Allan S. Detsky. "The Inevitable Application of Big Data to Health Care". In: *JAMA* 309.13 (Apr. 2013), pp. 1351–1352. ISSN: 0098-7484. DOI: 10.1001/jama.2013.393. eprint: https://jamanetwork.com/journals/jama/articlepdf/1674245/jvp130007_1351_1352.pdf. URL: https://doi.org/10.1001/jama.2013.393.

[2] Arthur W Toga et al. "Big biomedical data as the key resource for discovery science". In: *Journal of the American Medical Informatics Association* 22.6 (July 2015), pp. 1126–1131. ISSN: 1067-5027. DOI: 10.1093/jamia/ocv077. eprint: https://academic.oup.com/jamia/article-pdf/22/6/1126/34145802/ocv077.pdf. URL: https://doi.org/10.1093/jamia/ocv077.

[3] Clemens Scott Kruse et al. "Challenges and Opportunities of Big Data in Health Care: A Systematic Review". In: *JMIR Med Inform* 4.4 (Nov. 2016), e38. ISSN: 2291-9694. DOI: 10.2196/medinform.5359. URL: http://www.ncbi.nlm.nih.gov/pubmed/27872036.

[4] S. Shilo, H. Rossman, and E. Segal. "Axes of a revolution: challenges and promises of big data in healthcare." In: *Nature medicine* (Jan. 2020), pp. 29–38. DOI: 10.1038/s41591-019-0727-5. URL: https://doi.org/10.1038/s41591-019-0727-5.

[5] Huagen Liang et al. "Cobalt-nickel phosphides@carbon spheres as highly efficient and stable electrocatalyst for hydrogen evolution reaction". In: *Catalysis Communications* 124 (2019), pp. 1–5. ISSN: 1566-7367. DOI: https://doi.org/10.1016/j.catcom.2019.02.012. URL: https://www.sciencedirect.com/science/article/pii/S1566736719300469.

[6] David W. Bates et al. "Big Data In Health Care: Using Analytics To Identify And Manage High-Risk And High-Cost Patients". In: *Health Affairs* 33.7 (2014). PMID: 25006137, pp. 1123–1131. DOI: 10.1377/hlthaff.2014.0041. eprint: https://doi.org/10.1377/hlthaff.2014.0041. URL: https://doi.org/10.1377/hlthaff.2014.0041.

[7] Seyedmostafa Sheikhalishahi et al. "Natural Language Processing of Clinical Notes on Chronic Diseases: Systematic Review". In: *JMIR Med Inform* 7.2 (Apr. 2019), e12239. ISSN: 2291-9694. DOI: 10.2196/12239. URL: http://www.ncbi.nlm.nih.gov/pubmed/31066697.

[8] Shantala Giraddi and S Vaishnavi. "Detection of Brain Tumor using Image Classification". In: Sept. 2017, pp. 640–644. DOI: 10.1109/CTCEEC.2017.8454968.

[9] Alaá Rateb Mahmoud Al-shamasneh and Unaizah Hanum Binti Obaidellah. "Artificial Intelligence Techniques for Cancer Detection and Classification: Review Study". In: *European Scientific Journal, ESJ* 13.3 (Jan. 2017), p. 342. DOI: 10.19044/esj.2017.v13n3p342. URL: https://eujournal.org/index.php/esj/article/view/8693.

[10] Alvin Rishi Rajkomar, Isaac Kohane, and Jeff Dean. "Machine Learning for Medicine". In: *New England Journal of Medicine* (2019). URL: https://www.nejm.org/doi/full/10.1056/NEJMra1814259?query=featured_home.

[11] Andrew L. Beam and Isaac S. Kohane. "Big Data and Machine Learning in Health Care". In: *JAMA* 319.13 (Apr. 2018), pp. 1317–1318. ISSN: 0098-7484. DOI: 10.1001/jama.2017.18391. eprint: https://jamanetwork.com/journals/jama/articlepdf/2675024/jama_beam_2018_vp_170174.pdf. URL: https://doi.org/10.1001/jama.2017.18391.

[12] William Weintraub, Akl Fahed, and John Rumsfeld. "Translational Medicine in the Era of Big Data and Machine Learning". In: *Circulation research* 123 (Nov. 2018). DOI: 10.1161/CIRCRESAHA.118.313944.

[13] Xueqiang Zeng and Gang Luo. "Progressive sampling-based Bayesian optimization for efficient and automatic machine learning model selection". In: *Health Information Science and Systems* 5.1 (Sept. 2017). ISSN: 2047-2501. DOI: 10.1007/s13755-017-0023-z. URL: http://dx.doi.org/10.1007/s13755-017-0023-z.

[14] Houman Sotoudeh et al. "Artificial Intelligence in the Management of Glioma: Era of Personalized Medicine". In: *Frontiers in Oncology* 9 (2019), p. 768. ISSN: 2234-943X. DOI: 10.3389/fonc.2019.00768. URL: https://www.frontiersin.org/article/10.3389/fonc.2019.00768.

[15] Nicholas J. Schork. "Artificial Intelligence and Personalized Medicine". In: *Precision Medicine in Cancer Therapy*. Ed. by Daniel D. Von Hoff and Haiyong Han. Cham: Springer International Publishing, 2019, pp. 265–283. ISBN: 978-3-030-16391-4. DOI: 10.1007/978-3-030-16391-4_11. URL: https://doi.org/10.1007/978-3-030-16391-4_11.

[16] Atieh R. Khamesi, Eura Shin, and Simone Silvestri. "Machine Learning in the Wild: The Case of User-Centered Learning in Cyber Physical Systems". In: *2020 International Conference on COMmunication Systems NETworkS (COMSNETS)*. 2020, pp. 275–281. DOI: 10.1109/COMSNETS48256.2020.9027329.

[17] R. Dunford, Quanrong Su, and E. Tamang. "The Pareto Principle". In: *The Plymouth Student Scientist* 7 (2014), pp. 140–148.

[18] Riccardo Bonetto and Vincent Latzko. "Chapter 8 - Machine learning". In: *Computing in Communication Networks*. Ed. by Frank H.P. Fitzek, Fabrizio Granelli, and Patrick Seeling. Academic Press, 2020, pp. 135–167. ISBN: 978-0-12-820488-7. DOI: https://doi.org/10.1016/B978-0-12-820488-7.00021-9. URL: https://www.sciencedirect.com/science/article/pii/B9780128204887000219.

[19] Beverly Park Woolf. "Chapter 7 - Machine Learning". In: *Building Intelligent Interactive Tutors*. Ed. by Beverly Park Woolf. San Francisco: Morgan Kaufmann, 2009, pp. 221–297. ISBN: 978-0-12-373594-2. DOI: https://doi.org/10.1016/B978-0-12-373594-2.00007-1. URL: https://www.sciencedirect.com/science/article/pii/B9780123735942000071.

[20] W. Richert and L.P. Coelho. *Building Machine Learning Systems with Python*. Community experience distilled. Packt Publishing, 2013. ISBN: 9781782161400. URL: https://books.google.es/books?id=UQTJmgEACAAJ.

[21] Vladimir Nasteski. "An overview of the supervised machine learning methods". In: *HORIZONS.B* 4 (Dec. 2017), pp. 51–62. DOI: 10.20544/HORIZONS.B.04.1.17.P05.

[22] Sotiris Kotsiantis. "Supervised Machine Learning: A Review of Classification Techniques." In: *Informatica (Slovenia)* 31 (Jan. 2007), pp. 249–268.

[23] Johannes Fürnkranz. "Decision Tree". In: *Encyclopedia of Machine Learning*. Ed. by Claude Sammut and Geoffrey I. Webb. Boston, MA: Springer US, 2010, pp. 263–267. ISBN: 978-0-387-30164-8. DOI: 10.1007/978-0-387-30164-8_204. URL: https://doi.org/10.1007/978-0-387-30164-8_204.

[24] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[25] S.J. Pan and Q. Yang. "A Survey on Transfer Learning". In: *IEEE Transactions on Knowledge and Data Engineering* 22.10 (2010), pp. 1345–1359.

[26] Steven C. H. Hoi et al. *Online Learning: A Comprehensive Survey*. 2018. arXiv: 1802.02871 [cs.LG].

[27] Xu Ying. "Ensemble Learning". In: (May 2014).

[28] Pavel Brazdil et al. "Metalearning". In: Jan. 2017. DOI: 10.1007/978-1-4899-7687-1_543.

[29] Ricardo Vilalta and Youssef Drissi. "A Perspective View And Survey Of Meta-Learning". In: *Artificial Intelligence Review* 18 (Sept. 2001). DOI: 10.1023/A:1019956318069.

[30] Donald Michie et al., eds. *Machine Learning, Neural and Statistical Classification*. USA: Ellis Horwood, 1995. ISBN: 013106360X.

[31] Pavel Brazdil et al. *Metalearning - Applications to Data Mining*. Jan. 2009. ISBN: 978-3-540-73262-4. DOI: 10.1007/978-3-540-73263-1.

[32] Ricardo Prudencio and Teresa Ludermir. "Using Machine Learning Techniques to Combine Forecasting Methods". In: *In Lecture Notes in Artificial Intelligence*. 2004, pp. 1122–1127.

[33] Christiane Lemke and Bogdan Gabrys. "Meta-Learning for Time Series Forecasting and Forecast Combination". In: *Neurocomput.* 73.10–12 (June 2010), pp. 2006–2016. ISSN: 0925-2312. DOI: 10.1016/j.neucom.2009.09.020. URL: https://doi.org/10.1016/j.neucom.2009.09.020.

[34] Fred Collopy and JS Armstrong. *Rule-Based Forecasting: Development and Validation of an Expert Systems Approach to Combining Time Series Extrapolations*. General Economics and Teaching 0412004. University Library of Munich, Germany, Dec. 2004. URL: https://ideas.repec.org/p/wpa/wuwpgt/0412004.html.

[35] Monica Adya et al. "Automatic Identification of Time-Series Features for Rule-based Forecasting". In: *International Journal of Forecasting* 17 (Feb. 2001), pp. 143–157. DOI: 10.1016/S0169-2070(01)00079-6.

[36] Robert J Vokurka, Benito E Flores, and Stephen L Pearce. "Automatic feature identification and graphical support in rule-based forecasting: a comparison". In: *International Journal of Forecasting* 12.4 (1996), pp. 495–512. ISSN: 0169-2070. DOI: https://doi.org/10.1016/S0169-2070(96)00682-6. URL: https://www.sciencedirect.com/science/article/pii/S0169207096006826.

[37] Bay Arinze, Seung-Lae Kim, and Murugan Anandarajan. "Combining and selecting forecasting models using rule based induction". In: *Computers Operations Research* 24.5 (1997), pp. 423–433. ISSN: 0305-0548. DOI: https://doi.org/10.1016/S0305-0548(96)00062-7. URL: https://www.sciencedirect.com/science/article/pii/S0305054896000627.

[38] Omid Gheibi, Danny Weyns, and Federico Quin. *Applying Machine Learning in Self-Adaptive Systems: A Systematic Literature Review*. 2021. arXiv: 2103.04112 [cs.NE].

[39] H. Van Dyke Parunak and Sven A. Brueckner. "Software engineering for self-organizing systems". In: *The Knowledge Engineering Review* 30.4 (2015), pp. 419–434. DOI: 10.1017/S0269888915000089.

[40] Stepan Shevtsov et al. "Control-Theoretical Software Adaptation: A Systematic Literature Review". In: *IEEE Transactions on Software Engineering* 44.8 (2018), pp. 784–810. DOI: 10.1109/TSE.2017.2704579.

[41] Gordon Blair, Nelly Bencomo, and Robert B. France. "Models@ run.time". In: *Computer* 42.10 (2009), pp. 22–27. DOI: 10.1109/MC.2009.326.

[42] David Garlan et al. "Rainbow: Architecture-Based Self-Adaptation with Reusable Infrastructure". In: *Computer* 37.10 (Oct. 2004), pp. 46–54. ISSN: 0018-9162. DOI: 10.1109/MC.2004.175. URL: https://doi.org/10.1109/MC.2004.175.

[43] Leo Breiman. "Random Forests". In: *Mach. Learn.* 45.1 (Oct. 2001), pp. 5–32. ISSN: 0885-6125. DOI: 10.1023/A:1010933404324. URL: https://doi.org/10.1023/A:1010933404324.

[44] Vili Podgorelec et al. "Decision Trees: An Overview and Their Use in Medicine". In: *J. Med. Syst.* 26.5 (Oct. 2002), pp. 445–463. ISSN: 0148-5598. DOI: 10.1023/A:1016409317640. URL: https://doi.org/10.1023/A:1016409317640.

[45] Guolin Ke et al. "LightGBM: A Highly Efficient Gradient Boosting Decision Tree". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Long Beach, California, USA: Curran Associates Inc., 2017, pp. 3149–3157. ISBN: 9781510860964.

[46] Clinton Gormley and Zachary Tong. *Elasticsearch: The Definitive Guide*. 1st. O'Reilly Media, Inc., 2015. ISBN: 1449358543.

[47] Adam Paszke et al. "PyTorch: An Imperative Style, High-Performance Deep Learning Library". In: *Advances in Neural Information Processing Systems 32*. Ed. by H. Wallach et al. Curran Associates, Inc., 2019, pp. 8024–8035. URL: http://papers.neurips.cc/paper/9015-pytorch-an-imperative-style-high-performance-deep-learning-library.pdf.

[48] F. Pedregosa et al. "Scikit-learn: Machine Learning in Python". In: *Journal of Machine Learning Research* 12 (2011), pp. 2825–2830.

[49] Jacob Montiel et al. "Scikit-Multiflow: A Multi-output Streaming Framework". In: *Journal of Machine Learning Research* 19.72 (2018), pp. 1–5. URL: http://jmlr.org/papers/v19/18-251.html.

[50] Jesse Daniel, ed. *Data Science with Python and Dask*. USA: Manning Publications, 2019. ISBN: 9781617295607.

[51] Holger Krekel et al. *pytest x.y*. 2004. URL: https://github.com/pytest-dev/pytest.

[52] Alan Julian Izenman. "Linear Discriminant Analysis". In: *Modern Multivariate Statistical Techniques: Regression, Classification, and Manifold Learning*. New York, NY: Springer New York, 2008, pp. 237–280. ISBN: 978-0-387-78189-1. DOI: 10.1007/978-0-387-78189-1_8. URL: https://doi.org/10.1007/978-0-387-78189-1_8.

[53] Bernhard Pfahringer, Geoffrey Holmes, and Richard Kirkby. "New Options for Hoeffding Trees". In: *AI 2007: Advances in Artificial Intelligence*. Ed. by Mehmet A. Orgun and John Thornton. Berlin, Heidelberg: Springer Berlin Heidelberg, 2007, pp. 90–99. ISBN: 978-3-540-76928-6.

[54] M.A. Hearst et al. "Support vector machines". In: *IEEE Intelligent Systems and their Applications* 13.4 (1998), pp. 18–28. DOI: 10.1109/5254.708428.

[55] R. A. Fisher. "The Use of Multiple Measurements in Taxonomic Problems". In: *Annals of Eugenics* 7.7 (1936), pp. 179–188.

[56] Alistair Johnson et al. "MIMIC-III, a freely accessible critical care database". In: *Scientific Data* 3 (May 2016), p. 160035. DOI: 10.1038/sdata.2016.35.

# Appendices

# A Architecture integration

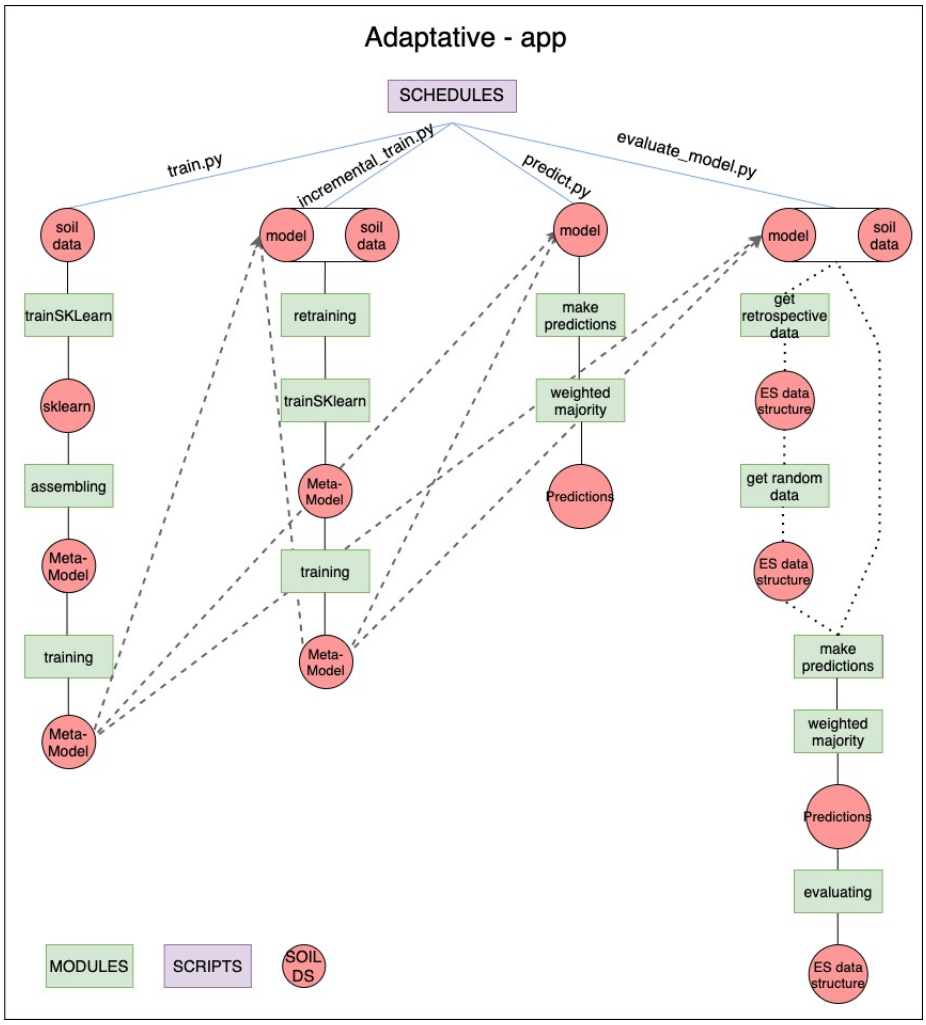


Figure 15: Architecture of the application, relation of all the Meta-Model components.

# B Synthetic Data

For the most part, data produced by a computer simulation may be considered synthetic data. This includes the vast majority of physical simulation systems, such as music synthesizers and flight simulators. The production of such systems is close to the real thing, but it is fully algorithmically generated.

In this project, we have created a package that is made up of various types of cyclic data generators to test the proposed Meta-Model. This module consists of basically two type of scripts: generators and test.

## B.1 Generators

Generators is a set of classes to generate fake data. There are different methods to create data depending on the desired output (numeric, classes...) and the used algorithms. As a result, five distinct generators have been developed. Four have been created randomly, the output variable does not follow any concrete pattern, and the missing one has been created to simulate the ED influx from data from an undisclosed hospital. Before using the data, we have ensured that all the data coming from this hospital could be used following the GDPR law as demonstrated in Annex E.

### B.1.1 Cyclic random generators

We have created four data generator classes randomly where the output variable does not follow any concrete pattern, it just changes in a cyclic manner. Some of these generators create binary target variables while others create numerical or categorical variables.

All of them have a method: `generate_data` which is the responsible of generating the data itself given a number of batches, $n_{batches}$; the approximated mean number of instances created in each batch, $n_{mean}$; and the number of explanatory variables that the user wants to use to generate the target variable, $d$.

This function returns:

- $X$: Matrix containing the attributes' data for all batches. Numpy ND array.
- $Y$: Array containing the values of each instance for all batches. Numpy 1D array.

- $t$: Array containing the batch id for each instance. Numpy 1D array.

The target variable, $Y$, can be discretized given some discretization method: threshold or k-means and given the number of desired classes obtaining a $c$ 1D Numpy array containing the labels of each instance.

**2 classes generator.** This is a basic data generator where $Y$ is a binary variable generated so that $x_1 + a * x_2 \geq b$ and changes over time in a cyclic manner. The main arguments of this class are:

- $p_a$: rate of change of $a$. Numerical type.
- $b$: constant. Numerical type.
- $pi$: probability of change. Numerical type.
- $noise$: maximum noise allowed. Numerical type.

Parameter $a$ changes following a sinus depending on the number of batches done such that $a = sin(p_a \times batch) + i$.

**Markov generator.** This is a basic data generator where data is generated following a two states Hidden Markov model. The main arguments of this class are:

- $init\_state$: state where the distribution starts. Integer.
- $p\_transition$: transition probability from one state to the other. Numpy 1D array.
- $Dices$: distribution of each state. Numpy 2D array

**Regression generator.** This is a basic data generator where $Y$ is generated so that $Y = a \times x_1 + b \times x_2$ and both $x_1$ and $x_2$ change in a cyclic manner. The main arguments of this class are:

- $p_{x1}$: rate of change of $x_1$. Numerical type.
- $p_{x2}$: rate of change of $x_2$. Numerical type.
- $a$: constant. Numerical type.
- $b$: constant. Numerical type.

- *noise*: constant. Numerical type.
- *method*: type of generation. 'threshold', 'kmeans' or 'regression'. String type.
- *clusters*: number of clusters. Numerical type.

The mean of $x_1$ changes following a sinus and the mean of $x_2$ changes following a cosine both depending on the number of batches done and $C$ is generated discretizing $Y$ with a threshold or a K-means algorithm.

**Basic regression generator.** This is a basic data generator where $Y$ is generated so that $Y = a \times x_1 + b \times x_2$ where both $x_1$ and $x_2$ are created randomly following a uniform distribution and $C$ is generated discretizing Y with a threshold or a K-means algorithm. The main arguments of this class are:

- $max_x$: maximum of the uniform. Numerical type.
- $min_x$: minimum of the uniform. Numerical type.
- $a$: constant. Numerical type.
- $b$: constant. Numerical type.
- *noise*: constant. Numerical type.
- *method*: type of generation. 'threshold', 'kmeans' or 'regression'. String type.
- *clusters*: number of clusters. Numerical type.

### B.1.2 Reaction model generator

The Reaction model generator is a synthetic data generator that simulates the Emergency Department occupation obtaining a realistic simulation of the influx of this department which, following the Pareto Principle [17], determines the overall stability of the hospital. This model is done on a minute scale simulating the CMBD (conjunt mínim bàsic de dades), a population registry that gathers information on pathology treated in Catalonia's health centers. The data used to produce this baseline comes from the the Sant Pau's hospital and we have demonstrated in Annex E that we can use it without breaking the GDPR law.

In Figure 16 the flux of the ED is depicted. People arrive to the hospital, to the admin part with a $\lambda$ rate. There they present their documentation and go

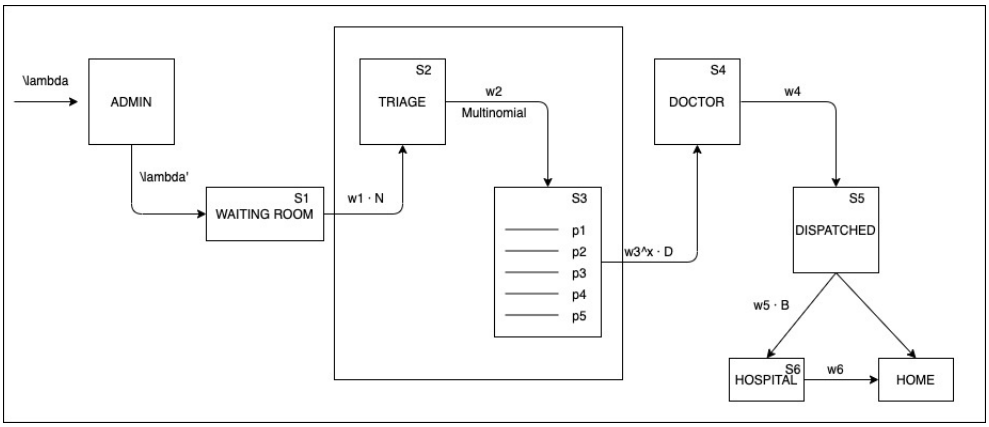


Figure 16: Flux of the ED.

to the first state **S1: waiting room**. Here they wait until they are visited by a nurse in triage who will determine the acuity of their illness or issue. The time people waits in the waiting room depends on the number of nurses who are in charge of triage, $N$, and the number of people in the waiting room, $w_1$. The time people is attended in **S2: triage** depends on the nurse speed encoded in $w_2$ and once they leave this state (S2), they are labeled from 1 to 5 according to their acuity. These labels are assigned with a multinomial distribution where label 1 indicates that the patient is dying and must be attended as soon as possible and label 5 indicates that the patient shouldn't have gone to the ER and can wait before being attended as its life is not in danger.

After that, patients are in **S3: waiting room** and the time they stay there depends on the number of people who is in the state $w_3$, the triage label assigned $x$ and the number of doctors who are attending patients, $D$. When they are being attended by a doctor they change to **S4: Doctor**. The length of the stay in S4 will depend on the severity of the patient and its diagnoses, information stored in $w_4$.

Then, patients are in the **S5: Dispatched** state and from here they can go home or they can be hospitalized depending on their diagnoses, $w_5$ and a binomial probability, $B$. If the patient is hospitalized he will go to **S6: Hospital** state where he or she will spend a different amount of time depending also on their diagnoses $w_6$. And finally, the patient will go home.

This synthetic data generator considers the following factors:

- The rate at which patients appear at the ER.
- The amount of professionals allocated to triage and boxes and the waiting period time before triage as well as the time spent being visited.
- The time it takes to attend the patient as well as the time it takes to find a bed for the patient if the patient has to be admitted.

**Emergency room arrival rate.**

In order to approximate the room arrival rate, we measured some aggregated averages at various scales: month, week of day, and hours. We literally take the average number of people who visit the ED per hour and also calculate the mean per day of the week and month and divide both measures by their mean to obtain a factor close to one which will be used to scale the mean per hours.

Once we got it, and for the sake of generating the arrival rate, we combine the three measures scaling the probability of arriving a user at a given time by the day of week and the month:

$$Pr(\text{arrival of patients})_{h,d,m} = Pr(h) \times d \times m \tag{2}$$

where $d$ and $m$ are the factors obtained from day of the week and month respectively.

**Triage.**

To provide an approach of the waiting time in the emergency room before going to S2, triage, we take the mean of the observed waiting time per hour and scale it by the number of nurses working in this sector.

In this state, we assign a label according to the acuity of the patient and the observed data following a multinomial distribution. We have seen that the severity states are the ones with less patients. Therefore, only a 10% of the patients have label 1, the proportion of patients who belong to label 2 is the same as the ones who belong to label 3, 15%, and the ones who belong to label 4 and 5 are the more frequent, 30% each group.

**Waiting room.**

The time spend here is modeled taking into account the average time that people of each group spends to the waiting room before being attended by a doctor.

From the empirical data of the undisclosed hospital, we have observed that the average waiting times are 11, 91, 102, 73 and 32 minutes respectively to each label from the most severe one (label 1) to the less one (label 5).

**Doctor.**

The time spend here is very variant as it depends on if the patient needs to get urgent surgery, if it was the case the patient would be considered to be hospitalized, or if the patient just needs a prescription before going home. After making some experiments and getting some metrics from data, we assumed the average time spend visited by the doctor to be 10 minutes with a high deviation. Therefore, people are being visited according to the parameter $w4$.

**Dispatched.**

Once the patient has been visited, he o she goes to the dispatched state where he or she is sent home or stays in the hospital following a binomial distribution encoded in $w5$. If the patient needs to stay in the hospital, we also need to take into account the number of available beds $B$. It is important to note that if the number of beds is scarce or if there is no available bed, the patient will stay for a longer period of time to the dispatched state.

**Hospital.**

This state simulates the length of the stay of a patient in the hospital. The time spent in this state is not straightforward estimated but using the Meta-Model as in the Transfer Learning experiment explained in Section 5.6. Therefore, $w6$ stores the features used by the Meta-Model to make the predictions.

**Home.**

In order to simplify the problem we assume that once the patient has been dispatched and sent home or sent home after being hospitalized, he or she stays there forever. We do not consider, in this case, that the patient can go to the ED again.

It is important to note that the latter model, the Reaction model, has been generated and tested but it has not been used during the experiments as during the course of the project we received extra data from St. Pau's hospital. The latter data was enough to train, test and validate or Meta-Models and so we decided to use it instead of the synthetic data.

## B.2 Tests

In order to validate the generated data created by the cyclic random generators a test method composed by a set of scripts has been generated. Therefore, we have checked that all the generators work properly. Hence, in order to test new functionalities we just need to check whether the new generated data is the same as the one we have generated with some fixed random seed and parameters and it can be automatically done by using Pytest [51] running: `pytest` in the generated folder.

## B.3 Installation and use

This package can be installed to any computer by downloading the following GitHub repository with the clone command:

```
git clone https://gitlab.com/amalfianalytics/
    research/fake-data-generators.git.
```

and running `pip install -e` in the generated folder. To use this package in a Python environment, we just need to import it as:

```
from generators import GeneratorName
```

and generate data with the desired generator calling it as:

```
data = GeneratorName()
```

# C Project planning

This project has been developed from the January 25th until around the July 16th of 2021. Therefore, the project duration has been approximately 24 weeks.

The work have had a load of 20 hours per week divided into 4 daily hours from Monday to Friday. Hence, I have performed a total of 480 hours in Amalfi Analytics dedicated to the project.

## C.1 Description of tasks

Here, we proceed to explain how the project has been scheduled as well as the primary project tasks, which may be classified into five phases:

- **T1:** Project management. (During all the project, especially at the beginning)
- **T2:** Study-related work and company familiarization. (January 25th – February 15th)
- **T3:** Meta-Model design as well as the generation of synthetic data using different methods design. (February 15th – April 5th)
- **T4:** Implementation and evaluation of the Meta-Model as well as the generation of synthetic data using different methods. (April 5th – May 31st)
- **T5:** Project documentation. (During all the project, especially from June 1st to June 18th)

### C.1.1 Project management

Project management is potentially one of the most critical groups of tasks of the project. It consists of defining the scope and the tasks and planning their distribution. The different tasks included in this section are listed below.

- **Contextualization and project scope.** Define the general objectives of the project, identify the problem to be resolved and contextualize it. Estimated time: 20 hours.
- **Temporal planning.** Planning the entire execution of the project, including a description of the tasks, resources and possible risks. It is essential to meet the project schedule and allows us to realize how much time

has to be invested in each part to meet the deadlines. Estimated time: 5 hours.

- **Economic management.** Analysis of the economic component of the project. Estimated time: 3 hours.
- **Meetings.** Meetings with the director and co-director will be scheduled every two or three days with the purpose of ensuring the correct evolution of the project. Team meetings are held every week to check the tasks. This implies an estimated time of 3 hours per week and a total of 24 weeks * 3 hours = 72 hours approximately.

**Study-related work and company familiarization**

This project has a big research component. Before starting the design and implementation of the new system, it is necessary to investigate the existing alternatives and the latest advances in the field. This part also includes all the necessary learning before starting the project as well as company familiarization.

- **Understand the current Amalfi Analytics' infrastructure.** Understand Amalfi Analytics' present architecture where the model must be incorporated. Estimated time: 25 hours.
- **Study Meta-Learning frameworks.** Study and investigate the existing alternatives and the latest advances in the field. Estimated time: 20 hours.

**Meta-Model design as well as the generation of synthetic data using different methods design**

Before starting the implementation it is necessary to design the main components of the final system. This step allows early error detention that can save a lot of time in the implementation.

- **Design architecture.** Design the architecture that is behind the Meta-Model as well as how it is going to be incorporated in SOIL. Estimated time: 40 hours.
- **Design synthetic data.** Design synthetic data format and specifications. Estimated time: 20 hours.

**Implementation and evaluation of the Meta-Model as well as the generation of synthetic data using different methods**

The main chunk of the work is in this section. We have to implement all the Meta-Model architecture in SOIL. The different tasks are the following:

- **Implement architecture.** Create all the architecture that will support the Meta-Model. Estimated time: 50 hours.
- **Methods.** Training, incremental-training, predicting and evaluating with the Meta-Model integrated to SOIL. Estimated time: 50 hours.
- **ML model.** Adapt some of the current machine learning algorithms and techniques from different libraries to work in the system. Estimated time: 15 hours.
- **Evaluation.** To assure correct operation in production, evaluate the Meta-performance Model's in various tasks, scopes and horizons. Estimated time: 40 hours.
- **Generation of four synthetic data generators.** Estimated time: 20 hours.
- **Code documentation and testing.** The implementation of each of the parts has to be accompanied by appropriate documentation and testing that ensures the correct functioning of the code. Estimated time: 30 hours.

**Project documentation**

- **Write paper.** By the end of May we had enough advancements and we submitted a paper to the 23rd International Conference of the Catalan Association for Artificial Intelligence. This task was optional and depended on the progress we had been able to make. Estimated time: 40 hours.
- **Write documentation.** The documentation has to be done during the entire course of the project. Estimated time: 50 hours.

Note that one of the milestones was the delivering of a Project Critical Review; in this semester, and it was presented on April 20th 2021. In it, the project evolution was discussed.

## C.2 Meetings and communication plan

### C.2.1 Meetings

Meetings with the director and co-director have been scheduled every two or three days with the purpose of ensuring the correct evolving of the project. Team meetings were held every week to check the tasks. They were remotely due to the current Covid-19 situation at the beginning of the project but since May they became face-to-face.

### C.2.2 Communication plan

Laura Aviñó, the project co-director, has been my go-to contact for needs, questions, deliverables... I have contacted her daily via Google Meet, Slack or in person at the office during all the project.

I have met all the tech team: Ricard Gavaldà (CEO), Jose Munuera (project director), Laura Aviñó (project co-director), Ton Rodriguez, Martí Zamora and Alba Gutiérrez in a weekly meeting on Friday or Thursday.

I have contacted Ton Rodriguez, Martí Zamora via Google Meet, Slack or in person at the office for problems related to libraries or servers respectively. And I have contacted Alba Gutiérrez also as needed via Google Meet, Slack or in person at the office for possible collaborations and support.

In addition, I have contacted Ricard Gavaldà for some technical questions and final milestones approval.

## C.3 Resources

This section lists the primary resources that have been required during the project.

### C.3.1 Human resources

I, the data scientist and researcher, was the primary human resource because I was the one working on it. However, both directors, Laura Avió and Jose Munuera, have been critical human resources in this project since they have coached and supported me.

### C.3.2 Material resources

Because research efforts are based on past studies, there has been a requirement for material resources such as articles or journals. In addition, I have required

some software and hardware resources such the ones listed below:

- **Gitlab**: It is the version control system used by Amalfi Analytics to collaborate in the same code-base.
- **Overleaf**: An online LaTeX editor with real-time collaboration, version control and templates. It has been used to write the documentation.
- **Computer** The project has been carried out using a MacBook Pro 2016.
- **Server** The project has been developed in a virtual private server (VPS) with 1CPU and 2GB of RAM.
- **Office space** Despite the fact that most of the project have been carried out online due to the Covid, two days per week we have worked from the office to boost collaboration and communication between coworkers. This is a material resource that we have to take into account.
- **Internet connection** It has been necessary to have a reliable internet connection.

## C.4 Risk management

As in any project, there have been obstacles and challenges that have had to be faced. In this part, I will discuss potential remedies and alternate approaches for dealing with these dangers.

### C.4.1 Tight schedules

The project's timelines are constrained. A little modification in the plan might create significant delays, putting the project's completion at danger.

- **Impact:** Very high.
- **Solution:** If an unforeseen incident causes a delay in the project, we will adjust the plan by deleting some non-essential capabilities or increasing the number of hours committed to the project if we do not want to limit the project's scope.

### C.4.2 Inaccurate estimations

We may have made faulty estimates because we did the estimations at a very early stage of the project. Furthermore, because this project has a research

component, it is impossible to predict how much time we will spend on the design and execution of certain components, as well as the implementation in Amalfi Analytics' architecture. This, along with tight deadlines, can have a substantial impact on the project.

- **Impact:** High.
- **Solution:** As previously said, the first alternative is to modify the plan or extend working hours; but, if no other options are available, we may have to modify the scope of the project. To prevent false estimations from harming the project, it is critical to regularly monitor the project's progress in relation to the initial plan.

### C.4.3 Familiarization with the company's architecture

Given that Amalfi Analytics has its own architecture and the proposed solution must be integrated on it as well as that multiple of the technologies at the company are new for me, getting started with the project and implementing it can take longer than expected.

- **Impact:** Medium.
- **Solution:** Avoid becoming weighed down by a lack of familiarity with the technologies utilized in the company's software development. This may be accomplished by asking coworkers questions and catching up as quickly as possible.

### C.4.4 Implementation errors

Errors in implementation can arise in every software project. Errors might cause task delays or the addition of additional tasks to resolve these faults. Our top objective is to identify these issues as soon as possible before they have a significant influence on the strategy.

- **Impact:** Medium.
- **Solution:** To reduce the chance of implementation problems, we will place a strong focus on designing the code ahead of time. We will be able to spot mistakes before we have spent hours writing code.

# D Cost analysis and economic viability

In this part, we analyze the expenditures that are only essential for this project's duties; to simplify the analysis, general expenditures such as office furniture, WiFi, and so on that should be anticipated for the project's proper development are not included.

We will divide these costs by the same work packages that were discussed at Section C.1.

- T1, T2, T3 and T4 need a pro Github account to manage the project (4€/month) This makes 20€.
- T4 needs, in addition to the previous needs, an online server in order to use SOIL and deal with different data and methods. We have used the Digital Ocean Server provided by Amalfi Analytics which costs 12€ per month. This makes a total of 60€.
- All tasks. Computers are essential for the project development and every employee have its own. It has to be powerful enough to support all the programming tools and software that is used to process all data and models. We estimate this cost to 1500€ per computer and person, which makes a total of 4,500€.
- All tasks. This project has been made by one data scientist and two advisors. The cost of having them working on this project is approximately of 9€/hour for the employee and 15€/hour for the advisors and coaches who have more responsibilities to attend. The estimated hours per person to be working on this project are approximately 4h per non week-end day for the data scientist and 1h per non week-end day for the advisors. Knowing the gross salaries to be 720€/month for the data scientist and 300€/month for the advisors, this makes a total of 6,600€ on gross salaries (this cost is already taking into account the 12.34% of IRPF). An extra cost of 7% on the gross salaries needs to be applied monthly in order to cover the payment to the Social Security (SS). The estimated cost of SS is then 462€. Thus means, we do not divide the employee salaries per tasks but per hours worked.

| | Type of cost | Task | Cost per unit | Total cost |
|---|---|---|---|---|
| Salaries | Direct and fix | All Tasks | 9€/hour Data Scientist, 15€/hour advisors | 6,600€ |
| Social Security | Indirect and fix | All Tasks | 7% of monthly gross salaries per employee | 462€ |
| Computers | Direct and fix | All Tasks | 1,500€/person | 4,500€ |
| Server | Direct and fix | T4 | 12€/month | 60€ |
| GitHub Pro | Direct and fix | T1, T2, T3, T4 | 4€/month | 20€ |
| **Total cost** | | | | **11,642€** |

Table 3: Total cost

# E Laws and regulations

To ensure compliance with Regulation 2016/679 of the European Parliament and of the Council of 27 April 2016, on the privacy of people with respect to the collection and free flow of personal data: GDPR[8], all Amalfi Analytics' products have been checked by expert jurists and the Data Protection Delegates of the collaborating health institutions.

However, in this section, we will examine the key points that support the fact that the application in the hospital ED where Amalfi Analytics could use my model is consistent with the GDPR[9]. We will focus specifically on: lawfulness of the treatment, pseudo-anonymized data and the impact assessment by the principle of proactive responsibility.

## E.1 Lawfulness of the treatment

According to the sixth article of the GDPR, which deals with the legality of procedure, our practice is legal because it is in the public interest. We use data from healthcare facilities to increase assistance efficiency and clinical management in emergency departments, as well as in the emergency departments themselves, which can then be directly translated to the patient's focus and source management and planning. As a result, the use of these hospital data is in the best interests of the public, and we do not require patient consents.

Furthermore, since we are dealing with anonymized and aggregated data, we would like to emphasize that the aim of this project is to assist in making management decisions, i.e., management of resources and staff groups; we would never make any personal decisions.

[8] https://rgpd.es/

[9] https://www.boe.es/doue/2016/119/L00001-00088.pdf

## E.2 Key points of the impact assessment by the principle of proactive responsibility

For each center, an anonymization procedure is followed during the data extraction process. Amalfi Analitycs has various types of protection measures in place during the extraction process, specifically: 8 honesty measures, 4 availability measures, and 6 confidentiality measures. The scoring assessment leads one to the conclusion that the extraction process has marginal risks that are appropriate under the control steps.

Script 1 Extractor is used by each center's Information Technology to retrieve data in a pseudoanonymous manner. Depending on its needs, the customer, i.e. each client, health center, may determine the degree of pseudonymization of the data. For example, one may wish to keep each patient's municipality of residence if they want to do differential studies by municipality, while another may not, and therefore may claim that they do not need to obtain the municipality. In the other hand, one would like to go really in detail with the age of the patients if they want to conduct pediatric trials on children aged 0 to 10, while another may be satisfied with age ranges ranging from 18 to 40, from 40 to 65 and over 65 years.

After that, the data is automatically transferred to the cloud web service via Script 2 Upload over a safe connection. During the configuration process, Amalfi Analytics verifies data at the destination to ensure personal comparative dissociation as well as as well as practical specifications. As a result of this approach, data is unrecognizable to Amalfi Analytics or the cloud. The findings of this review are only accessible to the approved members of the center. Furthermore, the link from the server to the core is encrypted and secure.

In addition, Amalfi Analytics has a protocol in place to protect data from unintended deletion, degradation, or harm, ensuring data security and confidentiality.

As a result, the impact evaluation based on the concept of constructive accountability document created by Amalfi Analytics and specialist jurists in compliance with articles 35 and 37 of the GDPR concludes that the treatments:

1. installed on the business cloud provider's external servers,
2. to interpret systematic patient data and make forecasts and mathematical analyses to better enhance emergency response management

involve a tolerable risk with a negligible effect, since the steps used and the recommendations represent a control situation

There have been no reports of cases involving a danger to patients' rights and liberties, as well as economic damages and reputational harm as a result of noncompliance with personal data privacy laws.