Automation of predictive algorithms and application to medical data

Bachelor's degree project

Octavi Pascual Sardà

Facultat d'Informàtica de Barcelona
Universitat Politècnica de Catalunya

Director: Ricard Gavaldà Mestre
Codirector: Jaume Baixeries i Juvillà

Bachelor degree in informatics engineering
Specialization: Computing

October 2017

Acknowledgements

I would like to thank my thesis supervisors Ricard and Jaume. They have advised me throughout the whole process of development of this thesis. Their engagement in this project has been decisive in the completion of this work. Their guiding, supervision and corrections have been invaluable and have allowed me to produce a satisfying bachelor’s degree project.

Abstract

English

Automated Machine Learning (known as AutoML) is a topic that gained lot of interest recently. Indeed, the success of machine learning in the tech industry has created a demand for tools that can be used by non-experts in the field. To create such a tool, we need a system that automatically executes some of the operations that used to be performed by real data scientists. In this project we will automate some of those operations, in particular model selection, hyperparameter optimization and validation.

This project is part of a collaboration between LARCA, a research group of the UPC and the Medical Services of Catalonia. Indeed, as a result of this association we will use a clinical dataset that was provided to the UPC and that is about readmissions of patients who suffered a heart failure.

Additionally, this project aims to build a web application to allow medical doctors and managers, with little training in machine learning, to benefit from the assistance of this discipline. Through this application they will be able to upload a dataset and train a model, which will be used to make predictions on unseen data.

In order to automate classification machine learning algorithms, we have to search through a series of hyperparameters, which is a task that is usually left to the data scientist, hence it requires some knowledge and expertise. We will use two techniques to carry out that exploration: grid search and random search.

Finally, we will try to evaluate how well our whole application performs, with a focus on the predictive power that it reaches.

Catalan

L'automatització de Machine Learning (AutoML) és una disciplina que ha guanyat molt d'interès recentment. L'èxit del machine learning en la indústria tecnològica ha creat una demanda d'eines que puguin ser utilitzades per persones que no són expertes en el camp. Per a crear una eina com aquesta, necessitem un sistema que automàticament realitzi una sèrie d'operacions que fins ara realitzen els data scientists. En aquest projecte automatitzarem algunes d'aquestes operacions, concretament la selecció de model, optimització de hiperparàmetres i validació.

Aquest projecte forma part d'una col·laboració entre LARCA, un grup de recerca de la UPC i els Serveis Mèdics de Catalunya. Un resultat d'aquesta associació és un dataset de dades mèdiques que va ser proporcionat a la UPC i que tracta de reingressos de pacients que han sofert un atac de cor.

Addicionalment, aquest projecte pretén construir una aplicació web per permetre als metges i directius, que no tenen experiència en machine learning, beneficiar-se de l'assistència que aquesta disciplina pot aportar. A través d'aquesta aplicació podran carregar un dataset i entrenar un model, amb el qual es poden realitzar prediccions sobre dades noves.

Per automatitzar algoritmes de classificació de machine learning, hem de buscar entre una sèrie de hiperparàmetres. Això és una tasca que acostuma a fer un data scientist real i per tant requereix un cert coneixement i experiència. Utilitzarem dues tècniques per dur a terme aquest exploració: cerca en quadrícula i cerca aleatòria.

Finalment, intentarem avaluar el rendiment de l'aplicació, centrant-nos en el poder predictiu que aconsegueix.

Spanish

La automatización de Machine Learning (AutoML) es una disciplina que ha ganado mucho interés recientemente. El éxito del machine learning en la industria tecnológica ha creado una demanda de herraminetas que puedan ser usadas por personas que no son expertas en ese campo. Para crear tal herramienta, necesitamos un sistema que automáticamente realice una serie de operaciones que hasta ahora se encargaban de realizarlas los data scientists. En este proyecto automatizaremos algunas de estas operaciones, concretamente la selección del modelo, la optimización de hiperparámetros y la validación.

Este proyecto forma parte de una colaboración entre LARCA, un grupo de investigación de la UPC y los Servicios Médicos de Cataluña. Un resultado de esta asociación es un dataset de datos médicos que fue proporcionado a la UPC y que trata sobre reingresos de pacientes que han sido afectados por un infarto de corazón.

Adicionalmente, este proyecto pretende construir una aplicación web para permitir a los médicos y directivos, que no tienen experiencia en machine learning, beneficiarse de la asistencia que esta disciplina les puede aportar. A través de la aplicación podrán cargar un dataset y entrenar un modelo, el cual se podrá usar para realizar predicciones sobre datos nuevos.

Para automatizar algoritmos de clasificación de machine learning, debemos buscar entre una serie de hiperparámetros. Esta tarea suele ser llevada a cabo por un data scientist y por lo tanto requiere un cierto conocimiento y experiencia. Utilizaremos dos técnicas para implementar esta exploración: búsqueda en cuadrícula y búsqueda aleatoria.

Finalmente, intentaremos evaluar el rendimiento de la aplicación, manteniendo el foco en el poder predictivo que se consigue.

Contents

1 Introduction 1

1.1 Context . . . 1

1.1.1 Stakeholders . . . 1

1.2 Objectives . . . 2

1.2.1 Obstacles . . . 4

1.3 Methodology and rigor . . . 5

1.3.1 Development tools . . . 5

1.3.2 Follow up tools . . . 6

1.3.3 Validation of results . . . 6

2 State of the art 7

2.1 Medical data . . . 7

2.2 Automation of predictive algorithms . . . 8

2.2.1 DataRobot . . . 8

2.2.2 Skytree . . . 9

2.2.3 Auto-WEKA . . . 9

2.2.4 Auto-Sklearn . . . 9

3 Project Management 11

3.1 Temporal Planning . . . 11

3.1.1 Project stages . . . 11

3.1.2 Action plan . . . 14

3.1.3 Resources . . . 15

3.2 Economic management . . . 16

3.2.1 Direct costs . . . 17

3.2.2 Indirect costs . . . 18

3.2.3 Contingency costs . . . 19

3.2.4 Total budget . . . 19

3.2.5 Budget control . . . 20

3.3 Sustainability report . . . 20

3.3.1 Environmental impact . . . 20

3.3.2 Social impact . . . 21

3.3.3 Economic impact . . . 21

3.3.4 Sustainability matrix 22
3.4 Laws and regulations 22
3.5 Changes from the initial planning 23

4 Machine Learning 27
4.1 Basic concepts 27
4.1.1 Definition 27
4.1.2 Types of machine learning 27
4.1.3 Types of data 28
4.1.4 Preprocessing 29
4.1.5 One hot encoding 30
4.2 Evaluation metrics 30
4.2.1 Classification accuracy 31
4.2.2 Confusion Matrix 31
4.2.3 Area Under ROC Curve (AUC) 34
4.3 Model selection 35
4.3.1 Train/test split 35
4.3.2 Cross-validation 36
4.3.3 Overfitting 37
4.3.4 The bias and variance trade-off 38

5 Classifiers 41
5.1 Decision Trees 41
5.2 Random Forest 42
5.3 K-Nearest Neighbors 43
5.4 Naive Bayes 43
5.5 Logistic Regression 44

6 Hyperparameter optimization 47
6.1 Definition 47
6.2 Manual search 47
6.3 Grid search 48
6.4 Random search 49
6.5 Bayesian Optimization 49
6.6 Comparing grid and random search 50

7 AutoML design and development 53
7.1 AutoML classifier . . . 53
7.1.1 Technology . . . 53
7.1.2 UML diagram . . . 53
7.1.3 Hyperparameter tuning . . . 56
7.1.4 Pipeline . . . 58
7.1.5 Preprocessing . . . 60
7.2 Server . . . 60
7.2.1 Technology . . . 60
7.2.2 Endpoints . . . 61
7.3 Client . . . 65
7.3.1 Technology . . . 66
7.3.2 Datasets . . . 66
7.3.3 Predict . . . 66
7.3.4 Results . . . 66
7.4 Interaction . . . 68

8 Clinical data analysis 71
8.1 Preprocessing . . . 71
8.2 Descriptive analysis . . . 73

9 Results 79
9.1 AutoML performance . . . 79
9.1.1 Iris dataset . . . 79
9.1.2 Adult dataset . . . 80
9.1.3 Congressional Voting dataset . . . 80
9.2 Clinical data results . . . 81
9.2.1 Sant Pau dataset . . . 81
9.2.2 Ictus dataset . . . 82

10 Conclusions 85
10.1 Technical conclusions . . . 85
10.2 Personal conclusions . . . 86
10.3 Future work . . . 86

Bibliography 89

A Technology **93**

B List of ICD-9-CM codes **95**

List of Figures

1.1 Parameters and hyperparameters 5

3.1 Gantt chart 24
3.2 Final Gantt chart 25

4.1 ROC curves 35
4.2 Holdout data 36
4.3 Cross-validation 37
4.4 Overfitting 38
4.5 Bias and variance 39

5.1 Sigmoid function 45

6.1 Grid search vs random search 51

7.1 UML diagram 54
7.2 AutoML pipeline 59
7.3 Datasets tab 67
7.4 Settings window 67
7.5 Predict tab 68
7.6 Results tab 68
7.7 AutoML overview 69

8.1 Admissions and discharges 77

List of Tables

3.1 Task distribution of GEP 12
3.2 Task distribution of product development 13
3.3 Task distribution of final report 14
3.4 Human resources cost 17
3.5 Hardware resources cost 18
3.6 Direct costs 18
3.7 Indirect resources cost 18
3.8 Contingency resources costs 19
3.9 Total budget 19
3.10 Sustainability matrix 22

4.1 Confusion matrix 32

8.1 Initial Sant Pau dataset 72
8.2 Final Sant Pau dataset 74
8.3 Summary statistics 75
8.4 Summary statistics 75
8.5 Summary statistics 76
8.6 Summary statistics 76
8.7 Summary statistics 77

9.1 Iris dataset results 80
9.2 Adult dataset results 80
9.3 Congressional Voting dataset results 81
9.4 Sant Pau dataset results 82
9.5 Sant Pau confusion matrix 82
9.6 Sant Pau confusion matrix 82
9.7 Ictus dataset results 83
9.8 Ictus confusion matrix 83
9.9 Ictus confusion matrix 83

B.1 ICD-9-CM codes 95

1 Introduction

1.1 Context

Automation of processes is a trend that has been growing at the same rate as computational power. That is why we are trying to automate more and more tasks. Data science is a field in which currently only experts can be working at. It seems reasonable to wonder if all those processes, that usually are carried out by experts, could be automated. That would make people with less experience to be able to work on those areas.

Predictive algorithms can be applied in almost any field. Provided that you have some kind of labeled data, you can try to predict the labels of similar but previously unseen data. Since algorithms are generic, any kind of data can be used. However, that situation is idealistic. In fact, data has to fulfill some restrictions depending on the algorithm that we are using, and depending on the data we will obtain better or worse results. Because of this it is convenient to focus on a specific area, in this case medical data. Medical data is one of the most areas since, depending on the analysis performed, life quality of patients can be improved.

This project is part of a larger one being carried out of the LARCA research group on analysis of electronic healthcare records in collaboration with several healthcare institutions in Catalonia. The project involves, among others, construction of predictive models for various tasks. The end users at the institutions are normally medical doctors and managers with little training in data mining. Therefore, the tools provided for them must be usable without the need of knowing the technical details of predictor building. To accomplish that, this project must allow running and optimizing the models in an autonomous way.

On the one hand, the purpose of this project is to build a software that automates predictive algorithms and on the other hand to apply this software to medical data.

1.1.1 Stakeholders

When developing a project, it is important to analyze the impact that our product will have to different targets. Among them there are direct and indirect users. The most important ones will be listed and described below.

Target audience

The intended readership of this publication are machine learning practitioners but it also can be read by anyone interested in machine learning.

Director and codirector

This project has a director and a codirector. Their role is to supervise the project and to support and instruct the student during the development of the project. The student can resort to them if he has any doubt in the realisation of the project since they are both experts on the topics covered by this project.

Users

The goal of this project is to allow non technical users who are not familiar with machine learning and data science to be able to make predictions of their data. The better the product generalises, the more potential users it will get. A web application will be developed in order to help users to interact with the product that will be built.

Beneficiaries

Beneficiaries refer to the people who will benefit from our final product. The potential reach of this product is far-ranging and problematic to be evaluated beforehand. In fact, its ambition is to work on any kind of dataset from a classification problem. It is hard to predict the reach of this product as it will be determined by how well our product generalises, that is, how good the performance of the models constructed by the product are. Also, it will be determined by the purpose of the user. In fact, if the user expects getting an approximate idea of the results that could be slightly improved with a deeper analysis, he will be satisfied. Aversely, if the user expects better results than the ones obtained by a real data scientist, he will probably feel disappointed.

1.2 Objectives

The central goal of this project is to automate the construction of a predictive model of data of tabular form, where rows are instances and columns are features. Moreover, we will not automate preprocessing nor feature engineering. To achieve that, we will develop a software that takes as input a dataset and gives to the user information about the models that we have tried and how well they performed.

We will define a series of objectives to accomplish in order to successfully achieve the project:

- **Analyze a clinical dataset**
  Before automating a process, it is unavoidable to execute that process manually in order to understand what can be automated and how to manage it. The said process will be performed using a clinical dataset from the *Hospital de Sant Pau*. Moreover, we will use this dataset as a reference when automating the predictor building process. That is, we will be able to compare the automated execution against the manual one and identify possible errors or problems.

- **Automate predictive algorithms**
  We will automate a series of predictive algorithms in order to find the one that better predicts a given property from the dataset. Our program will try each of those and try to find the better parameters for them. At the end it will report to the user which model is the best one, and the user will be able to use that model to predict new data.

- **Explore hyperparameter tuning of predictive algorithms**
  The quality of the results obtained with predictive algorithms is highly dependant on its hyperparameters. Because of that, implementing algorithms that explore efficiently those hyperparameters appears to be decisive.

- **Create an application to interact with the system**
  In order to use the system a user interface must be developed. Indeed, a web application will be implemented to upload data, visualize the results and predict the class of new observations.

To validate that our software satisfies the objectives defined above, we will use a dataset which has been studied in a previous work [34]. We will compare the results given by our automated software and those obtained by the analysis of a data scientist. We expect to obtain similar results, even though they might differ. In fact, pre-processing a dataset is a decisive step that determines the quality of the results. Thus, if the pre-processing differs then the data will also differ and the results will change, so the comparison might not be completely fair.

1.2.1 Obstacles

In the course of a relatively long project complications may arise. Pretending that there will be no issues seems unrealistic, so instead we will take them into account. Next we will identify and describe the possible obstacles that may appear.

Data Quality

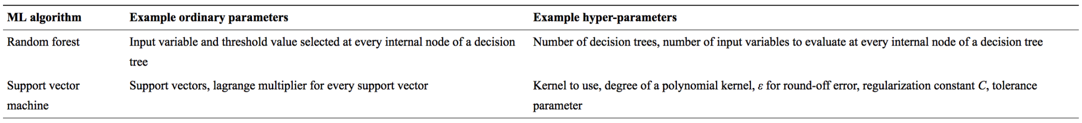
In data mining, the quality of the data determines the performance of a model. Because of this, if the quality of the data is poor, we will obtain poor results. This is known as the "garbage in, garbage out" principle [33]. In fact, it is known that pre-processing is a critical step and has a huge impact on the performance of a data mining project[30]. It is worth noting that this step is the harder one to automate, as it usually requires domain knowledge. If there is enough time, we will try to explore some basic pre-processing techniques.

Software errors

Software errors, most known as bugs, happen in any project. In data mining it might be hard to detect them as there is not an easy way of knowing what the correct output should be. In order to detect bugs, tests must be done. It is very important to detect bugs early in the development cycle as otherwise they can be harder to spot and will penalize our project. Knowing what to test and what not to test is also significant. For example, we will assume that the libraries we use are correct. It is reasonable to do that since we are using a trusted implementation. In our case integration tests are probably the most valuable ones.

Computational power

The majority of models have hyperparameters that must be tuned in order to be able to better predict the real label. In Figure 1.1 we can see some examples. The problem of hyperparameters is that they cannot be directly learned from the proper training process. To decide them we must train different models and choose the values that minimize the error of the model. If the dataset is huge, training a model several times with different hyperparameters can be very expensive in terms of computational time and resources. We will have to take into account those elements when implementing our project. If needed, the size of the dataset can be reduced or we can envisage using another computer than the personal one.



ML algorithm	Example ordinary parameters	Example hyper-parameters
Random forest	Input variable and threshold value selected at every internal node of a decision tree	Number of decision trees, number of input variables to evaluate at every internal node of a decision tree
Support vector machine	Support vectors, lagrange multiplier for every support vector	Kernel to use, degree of a polynomial kernel, ε for round-off error, regularization constant C, tolerance parameter

Figure 1.1: Some example machine learning (ML) algorithms, ordinary parameters and hyperparameters [31]

Schedule

This project has a final deadline. In real industry, many software projects are delayed. Here, we cannot afford doing that. For this reason, a plan has to be constructed and followed at all costs. Designing this plan has to be done carefully as we cannot be extremely optimistic. Taking into account unexpected incidentals is important.

1.3 Methodology and rigor

When developing a project, it is important to fix a methodology and ways of measuring and monitoring the work that has been done.

1.3.1 Development tools

Our project will be built using Python as it possesses some powerful machine learning libraries backed by the data science community. The main advantage is that it is open-source and multi-platform. Also, there is a huge community so getting help over the Internet is extremely easy. We will use Conda to manage library versions. In addition, the director of this project suggested this language since it is the one that is being used in the LARCA project. It is useful to maintain the same language among projects since it makes easier to integrate them or reusing some parts.

Nowadays, it is unthinkable to develop a project without a Version Control System (VCS), even when being a single developer. We will use Git since it is the most used one since 2014 [36]. This will allow us to track the work that has been done and to strengthen our fast programming cycles.

1.3.2 Follow up tools

During the execution of the project it is useful to know at which stage of the project we are. That way, we can realize if we are in advance or delayed. Moreover, follow up tools allow other people to check how the project is going.

We will try to follow the schedule defined in the Gantt chart. To do that we will communicate via email with the directors to inform them at which step we are. If we foresee deviations we can discuss why it is happening and adapt to those circumstances.

1.3.3 Validation of results

To validate our results it is necessary to be able to measure somehow the performance of our software. We have been provided with the final degree project of a student who worked with the same dataset that we will use as reference [34]. That means that we will know in advance which results our software should obtain for that given dataset. We will try to obtain the same results with our automated software. We must keep in mind that during pre-procesing the changes in the dataset may differ from the ones that were done in the previous work so results can diverge because of that.

Another way of validating our work is to provide more datasets which have already been analyzed so we know the expected results. Since our software is a general purpose one, it should work for any kind of dataset. However we must keep in mind that achieving that is idealistic and that some kind of restrictions will have to be made. Therefore, we will try to select datasets that adapt to those restrictions in order to be able to evaluate our software in a fairly way.

2 State of the art

Reviewing the literature about the topic we are working with is convenient as it can help to better understand it. In this project we are studying automation of predictive algorithms and its application in medical data, so it seems reasonable to research about those two fields.

2.1 Medical data

Data analysis on medical data is an application of computer science that focuses on analyzing electronic medical records. Since our reference dataset is about readmissions, we will focus on the work it has been done in this area. A readmission occurs when a patient is hospitalized shortly after he has been discharged. Usually hospitals are penalized by readmissions so it is interesting for them to avoid those. Also, readmissions have an impact on the quality of the hospital and on the society. In the clinical literature readmissions typically refer to hospital admissions within 30 days following the initial discharge.

First, we will focus on *Assessing the Predictability of Hospital Readmission Using Machine Learning* [17]. In this paper a study on readmissions has been made and they found that a substantial portion of readmissions is inherently hard to predict. To do that, they introduce an index named LACE which measures the length of stay, the acuity of the admission, the comorbidity of the patient and the emergency department use. This index can be deterministically computed if those four variables are known. They show that using LACE index is a poor tool to predict readmissions. That is why we must search other techniques to accomplish that task. For this reason using machine learning to predict readmissions seems appropriate. Our project is pertinent as it has been shown that easier models do not work to solve the problem we are dealing with. At the end of the paper they also suggest that feature reduction methods can be applied to the dataset in order to reduce the dimensionality. That also interests us as reducing the size of the dataset can speed up the algorithms that we will apply. Furthermore, by reducing the data we may also reduce the noise and improve the accuracy of our predictions.

Evaluating Preventive Measures for Heart Failure Readmissions using Machine Learning [23] studies the same dataset that we will use as reference thus reading it seems mandatory. It describes the characteristics of the dataset and the models that have been applied. The interesting part of this work is that it also tries to take into account the cost of readmissions to compute the savings that a hospital can make. To do that they use a ROC (receiver operating characteristic) curve which is a graphical plot that illustrates the performance of a binary classifier. In that curve we can graphically compare different classifiers and easily detect which is the better one. This technique is helpful as in our project we will also compare different classifiers.

2.2 Automation of predictive algorithms

This process has been mostly developed by private business who want to commercialize their product. Indeed, we can talk about machine learning as a service. They offer a service that is able to automate all the data mining process, not only the predictive model construction. Furthermore, they also offer a sophisticated interface which includes graphics and real-time information about the training process.

Meanwhile, in terms of literature, there is not a lot of information since the automation itself has not been much discussed. Literature is more concerned about the techniques themselves and not about automating them and offering them as a service. However, the techniques that are used to search efficiently in the hyperparameter space of the algorithms have captured the attention of researchers. That is why we can find some open-source software that focus on automating the predictive modelling construction. This software is more oriented to developers since they only provide a library to interact with and not a full service as the ones cited above.

2.2.1 DataRobot

DataRobot [4] provides a predictive analytics platform to rapidly build and deploy predictive models to the cloud. It is targeted for data scientists of all skill levels and uses massively parallel processing to train and evaluate models. One of the interesting features of this platform is that it tries hundreds of algorithms from different libraries and languages such as Python, R or Spark. A real data scientist usually performs this operation with the package he is comfortable with, not trying all those different languages (indeed, it is infeasible for a single person to master so many different languages). In addition, they also take care of preprocessing the dataset Finally, they take care of the storage of the model and, in order to interact with it, they provide an API. This project is in a very advanced state both in terms of infrastructure and interface. Our project does not intend to implement such a massive parallelization as we will develop the app in a local environment and we do not intend to use cloud computing. Moreover, our interface will not provide advanced visualitzations about the results.

2.2.2 Skytree

Skytree [6] develops machine learning software for enterprise use. One of the interesting features of this product is that algorithms parameters can either be automatic or the user can select their values. Moreover, it provides advanced visualization tools, especially in decision trees where the tree and the rules of each node can be displayed. In our project they will be automatic since the final user will not have any kind of knowledge related to machine learning.

2.2.3 Auto-WEKA

WEKA [15], which stands for Waikato Environment for Knowledge Analysis, is one of the most famous machine learning platforms. It is an open-source project, provides a graphical interface and a Java API to integrate it in your software. Auto-WEKA [39] is based on WEKA but incorporates model selection and hyperparameter optimization. It searches through hyperparameter settings to maximize the performance of the model and uses bayesian optmization to accomplish that. In our case we will not use such an advanced optimization method.

2.2.4 Auto-Sklearn

Auto-Sklearn [9] is an open-source project that provides an automated machine learning toolkit for a Sklearn estimator. It comes in the form of a Python module and shares the same API than other Sklearn estimators. In *Efficient and Robust Automated Machine Learning* [25] there is a technical explanation of the technology and the design of this project. It is also based on bayesian optimization but also uses meta-learning techniques by taking into account the performances of similar datasets.

3 Project Management

3.1 Temporal Planning

This section aims to define and describe the tasks that are going to be accomplished during the project. It is useful to be aware of the temporal planning of the project in order to culminate it in the specified time frame. Despite keeping that plan in mind, we have to accept that it is subject to modifications depending on the actual duration of each individual task.

The estimated duration of this project is about 4 months and a half. More precisely, this project starts on February 13th and ends on June 30th. Since this is an end of degree project that is compulsory for the degree and the author of this document is expected to graduate from university this summer, the deadline must be considered final. For this reason, extending the deadline is not feasible and it is imperative to appropriately plan the project.

The workload of this project is 18 ECTS (European Credit Transfer and Accumulation System). Each credit corresponds to 25 up to 30 hours of work, so the workload is between 450 and 540 hours. We will approximate our planning to fulfill those requirements.

3.1.1 Project stages

Project Management (GEP)

Project management is a course that covers the planning of this project. Every week a deliverable is submitted. We are working using a waterfall model which is a sequential model. In Table 3.1 we can see the task distribution of GEP. The time of each task has been established using the course syllabus.

Product development

This phase is the most significant one as it is the reason of doing all this work. All the other tasks of the project are done in order to develop successfully our product. To develop our product, we will use an incremental build model. That means that we will design, implement and test a minimal working product first. Then we will iteratively add functionalities until satisfying all the requirements. We will combine this incremental philosophy with an agile process model. The latter fits in such a

Task	Time (hours)
Context and scope of the project	20
Project planning	8
Budget and sustainability	9
First oral presentation	6
Review of bachelor’s thesis competences	15
Oral presentation and final document	18
Total hours	76

Table 3.1: Task distribution of GEP

short project as we need fast iteration cycles and continuous tracking. In Table 3.2 there is the task distribution of this phase. The time of each task is orientative as it is hard to predict, but since we are using an agile methodology we will be able to adjust it if necessary.

Initial set up

This stage consists on preparing the working environment that will be used to develop our project. In this case, this is mostly configuring libraries and frameworks that we set in the development tools section. We will have to check that there are no issues during and that everything is working and well-configured. If not, we may have to reconsider which tools we will use.

First prototype construction This step will give us a product that has all the core functions. Those are reading the dataset, splitting into training and testing data, implementing the simplest classifier, applying an evaluation method and trying the prototype with the reference dataset. This will generate a minimal product that is already operational and we will only add new features to it. Furthermore, we will obtain immediate feedback of the performance gain when doing those additions.

Implementation of the remaining classifiers

This step consists in implementing the rest of the classifiers that we will use. Since

Task	Time (hours)
Initial set up	1
First prototype construction	110
Implementation of the remaining classifiers	70
Implementation of additional methods	70
Automation of pre-processing	60
Testing with different datasets	50
Total hours	361

Table 3.2: Task distribution of product development

we will already have a first product that works, we will be able to easily integrate new classifiers and see how they perform.

Implementation of additional methods

In this phase we will incorporate additional methods that should improve our classifiers. Among those methods there is over and under sampling, cross-validation, ROC curve and confidence calibration.

Automation of pre-processing

Once we have a product that analyzes a dataset and tries different classifiers on it, we can improve their accuracy by adding some methods that modify the data in order to help the classifiers. For example we will treat outliers or select, remove or derive certain features.

Testing with different datasets

This last task is important as it will evaluate our product. The latter was intended for working with any kind of dataset, so this test will determine how well our product is performing.

Final report

This phase is the one that closes our project. After having implemented our product, we have to compose a report which exposes all the results that we obtained during

Task	Time (hours)
Writing of report	40
Final presentation	15
Total hours	55

Table 3.3: Task distribution of final report

the project. In Table 3.3 we can see the time commitment of each of those two tasks.

Writing of report

In this task we will document all the project, hence we will generate a report explaining in detail all our results.

Final presentation

In this task we will prepare our final presentation which consists on an oral defense of our project.

Gantt chart

Figure 3.1 presents the Gantt chart of the whole project. The tasks are the same as the ones listed in the above sections.

3.1.2 Action plan

According to BusinessDictionary [2], an action plan is a *"sequence of steps that must be taken, or activities that must be performed well, for a strategy to succeed"*. We have already defined our strategy and now we will explain how we will ensure that it is accomplished. Furthermore, we will exhibit alternative solutions to potential deviations.

The initial idea is to follow the sequence of tasks defined in the Gantt chart. However, as we mentioned before, difficulties that delay a task may happen and we have to establish some strategy to deal with that situation. We decided to give priorities to features that our product will provide. Some of them are mandatory whereas others are optional as they are methods or techniques that might improve the accuracy or the level of automation of our product. Therefore we will guarantee that the mandatory ones are implemented and in order to accomplish that we might abandon an optional feature.

In order to make sure that mandatory features are completed, we placed them in the beginning of the project development. If we need more time to finish the first prototype, we will have it. We will then proceed to reject an additional method or pre-processing automation. The latter is one of the most difficult tasks to automate as the input is unpredictable. For that reason we have a lot of flexibility to decide how much dedication we end up assigning on that task. A similar argument can be made for the other additional features that will improve the quality of our product. Dismissing one of those features is not severe as they are not indispensable for the final product. Hence, it is legitimate to take this action.

From the task distribution tables we can observe that the project will take around 500 hours, which meets the workload estimated in the first section. Moreover, this number is about the average of the lower and higher bounds so we can afford slight deviations in our planning without leaving that range.

3.1.3 Resources

This section depicts the resources that will be needed to accomplish the project. We will divide them in three resource categories: human, hardware and software.

Human Resources

- **Project manager**: The person who is in charge of leading the project. He is in charge of planning and monitoring the tasks that will be executed during the project.

- **Data scientist**: The person who has skills and possesses a strong knowledge in predictive algorithms. He will provide insights about the classifiers and methods that the software developer will implement.

- **Software developer**: The person who is in charge of developing the product following the directives from the project manager and the data scientist.

Hardware Resources

- Personal computer (Intel Core i5 2.6 GHz, 8GB RAM, 256GB SSD): used in all tasks of the project

Software Resources

- *macOS Sierra v.10.12.2*: used in all tasks
- *Python*: used in product development
- *Conda*: used to manage *Python* environments
- *JavaScript*: used in product development
- LaTeX: used to perform documentation tasks
- *TeamGantt*: used to produce Gantt chart
- *Draw.io*: used to generate flowcharts and UML

3.2 Economic management

In this section we will discuss the economic aspects of this project. Part of the planning of a project consists in evaluating its economic and financial viability. In this analysis we will divide the costs in the following categories: direct, indirect, contingencies and incidentals. At the end we will aggregate all those categories to obtain the total budget of the project.

We will make a series of assumptions:

- Escalation is not going to be taken into account since the project will last for a short period of time so sudden changes in prices are not likely to happen.
- The author of this document will perform the three different roles that have been specified in the resources section. Since he holds a junior profile, the remuneration of those roles will adjust to that characteristic.

Role	Pay (€/h)	Estimated time	Cost(€)
Project manager	15	96	1440
Data scientist	12	124	1488
Software developer	10	272	2720
Total	-	492	5648 €

Table 3.4: Human resources cost

3.2.1 Direct costs

Direct costs are those which are immediately related with the production of a product and includes items such as software, equipment, labor and raw materials [18]. We will split direct costs into human resources, hardware resources and software resources.

Human resources

The cost of human resources are listed in Table 3.4. As it has been explained, there are three different roles that will be taken by the same person, the author of this document. To determine the pay we took into account his junior profile.

The number of estimated hours matches with the task distribution that was defined in the Gantt chart. The project manager will work mostly on the project management and the final report, as well as monitoring the project. The data scientist will help the software developer to implement the machine learning techniques in our product and will also collaborate with the technical parts of the reports. Finally, the software developer will program the product in its development phase and eventually help with the documentation of his work.

Hardware resources

Table 3.5 contains the hardware that we will use in order to execute the project.

Software resources

The software that we will use is either free or came with the hardware. Because of that we will not list all the software resources and we will directly assign a total cost of 0 €to them.

Item	Units	Price per unit (€)	Useful life (years)	Amortization (€)
Personal computer	1	1529	6	25.48
Total	-	1529	-	25.48 €

Table 3.5: Hardware resources cost

Total direct costs

Once we have computed the budget of each category we can add them to derive the total budget of direct costs. It is presented in Table 3.6.

Category	Cost (€)
Human resources	5648
Hardware resources	25.48
Software resources	0
Total	5673.48 €

Table 3.6: Direct costs

3.2.2 Indirect costs

Indirect costs go beyond the costs associated with creating a particular product to include the price of maintaining the entire company [18]. For example cost of electricity or office equipment are indirect costs. In Table 3.7 we list them.

Product	Price (€)	Estimated time	Cost (€)
Electricity	0.116 €/KWh	492 hours	5.75
Internet	17.37 €/month	5 months	86.85
Office equipment	50	-	50
Total	-	-	142.60 €

Table 3.7: Indirect resources cost

3.2.3 Contingency costs

When planning our budget, due to incomplete information or oversights we can miscalculate a resource cost. For example, if there is an increment in the number of hours, more working hours will be required thus we will have to assign more resources to the human category. Usually, we calculate it as a percentage of the value of the budget. The latter depends on the risk and on the past experience. We will only consider human resources as we do not expect to need additional hardware nor software. Indeed, if we need an additional software resource we can assure that we will find an open-source resource. Finally, to establish the contingency percentage we will take into consideration that at most we should extend the total workload to 540 hours. In Table 3.8 we show the results.

Category	Cost	Contingency percentage	Contingency cost (€)
Human resources	5648	10	564.80
Total	-	-	564.80 €

Table 3.8: Contingency resources costs

3.2.4 Total budget

To terminate this section, we only have to aggregate all the results that we computed in the previous divisions. This is done in Table 3.9.

Category	Cost (€)
Direct resources	5673.48
Indirect resources	142.60
Contingencies resources	564.80
Total	6380.88 €

Table 3.9: Total budget

3.2.5 Budget control

As we have seen, the main part of the budget resides in human resources. Moreover, it is not expected to need more software nor hardware. We can say that the only budget category that we have to control is the one related to human resources.

The budget assigned to human resources could need a modification if the project schedule is compromised. That is, if we need more hours than we have predicted in the time management section. If the product development takes longer than expected, we will have to increase the amount of money that we initially reserved to that category.

3.3 Sustainability report

In this section we will evaluate the impact of our project in three different scopes: environmental, social and economic. Then we will rely on the appreciations we made to construct the sustainability matrix.

3.3.1 Environmental impact

Hardware resources used in this project anticipate a lengthy amortisation. In fact we only use a personal computer which is anyways used by almost any task due to the author's studies. In some way we could argue that this project does not require any additional hardware since the author would have a personal computer regardless of this project. To conclude, we could say that this project is more oriented to exploit knowledge in both data science and software than to exploit hardware resources.

We can now focus on the environmental impact of the personal computer. We can see in the specifications [16] that this computer consumes 60 Watts. We will consider that we will it for all the tasks of the project, that is for around 500 hours. Then the total amount of energy needed is 30 kWh. Computing how much CO_2 is generated to produce this energy is tangled since it varies substantially by form of generation. For example, coal or natural gas produce much more CO_2 than renewable energy such as wind. Nevertheless we will try to approximate that value. According to *Generalitat de Catalunya* [21] we emit 308 g of CO_2 per kWh. That means that we are generating 9.24 kg of CO_2 in total pursuing this project.

Finally we can think about which materials we will reuse. In our product we will use libraries to construct our predictive models. Implementing those models from scratch would be certainly time consuming and would require more human resources.

To close this environmental analysis we must grant a score to this dimension. As we said, the resources we mainly need are software and a personal computer, which will continue to be used once the project is completed. We decide to give a 9.

3.3.2 Social impact

Machine learning is a field that nowadays is present in most of the industries as they all want to extract value from their data. The author of this document is inclined to mathematics and computer science, two areas that are fundamental in machine learning. He wants to learn more about that field and performing this project will help him to decide what steps to take in his career.

Ideally, our product will be able to help hospitals by evaluating their medical data. Exploiting this kind of data is powerful since it saves money from the hospitals and improves the quality of life of the patients and their families. These days many hospitals might be skeptical about the benefits of medical data analysis. One of the main reasons is that they need a data scientist and that can lead into a considerable investment. Moreover, results are not immediate, they can only be used after the analysis has been completed. Our product can help the hospital to establish a first contact with data analysis. They will obtain some results, that almost certainly will not be as good as those achieved by a data scientist, but hopeful enough for them to consider pursuing further analytics. In brief, our product could make hospitals more interested and more willing to invest money in this area.

To conclude this part, we will award it with a 9. Our product has a lot of potential but at the same time we do not control entirely how it will impact the society.

3.3.3 Economic impact

In the previous section we already did a detailed analysis of the economical costs of this project. We estimated the budget that was needed and we took into account direct and indirect resources and contingencies. To determine whether the project is competitive or not we can wonder if we could have used fewer resources or less time. Considering that our budget consists mainly on human resources, that all the software used is open source and that indirect costs can not be avoided, we will focus on the first. To reduce human resources cost we can either try to reduce pay or try to reduce the estimated time. On the one hand, reducing pay seems unreasonable as the person is already a junior. On the other hand, to reduce the estimated time we would need a higher profile, but that would considerably increase the pay.

We will now see if we allocate our resources to the most important tasks. That is, if time spent on each task reflects its importance. As we explained in *Temporal Planning* chapter, by taking an incremental approach when building our product we ensure that essential tasks will be executed first. That means that we will optimize our resource distribution.

All in all, the economic resources are very tight and lowering them seems unworkable. Because of this we will give a 10 to this category.

3.3.4 Sustainability matrix

To complete this sustainability block we will proceed to construct the sustainability matrix. In Table 3.10 we present it.

	Economic	Social	Environmental	**Total**
PPP	Bill: 9/10	Personal impact: 8/10	Design consumption: 9/10	26/30
Useful life	Viability plan: 15/20	Social impact: 16/20	Ecological footprint: 17/20	48/60
Risks	Economical: -3/-20	Social: -5/-20	Environmental: -1/-20	-9/-60
	-	-	-	65/90

Table 3.10: Sustainability matrix of the project

3.4 Laws and regulations

This section identifies laws and regulations that may be related with the project. Since we own a medical dataset we must seek for information about how this personal data must be treated. In Spain, this kind of data is regulated by LOPD (Ley Orgánica de Protección de Datos) which is a series of laws that regulate personal information. It is based on the article 18.4 of the Spanish Constitution [1] and it makes reference to the right to family and personal privacy and the secret of communications.

3.5 Changes from the initial planning

During the development of this project, there has been a major change: the delivery of the project, which initially was planned to be on the June shift, was delayed to October. This decision was consensual between the directors of the project and the author and it was in favor of a better quality and exhaustive project. In fact, it was decided that a web application would be build to interact with the final user.

This decision forces to revise and reorganize the planning that was initially produced. In Figure 3.2 we see how the new planning has been restructured. To avoid delaying all the writing of the final report to the *Final Report* stage, we have interleaved implementation and documentation in *Project Development* stage. That is, part of the final report will be written during said stage since it is more convenient to explain concepts right after studying or implementing them. Moreover, its revision will be more manageable by supervisors as they will receive smaller pieces of documentation.

Since the temporal planning has been modified, costs must also be revised. In terms of costs, there are no alterations since the software developer will be able to create the web application, so it is not needed to hire an additional member. Moreover, the implementation of the automated model builder was easier than it was initially thought so part of the hours that were initially allocated on the builder were transferred to the implementation of the web application.

In terms of methodology, Sklearn library has proven to be an excellent library to work with machine learning in Python. In fact, it covers many of the common operations and allows to iterate in a fast pace. It even provides parallelization. For that reason changes were not needed.

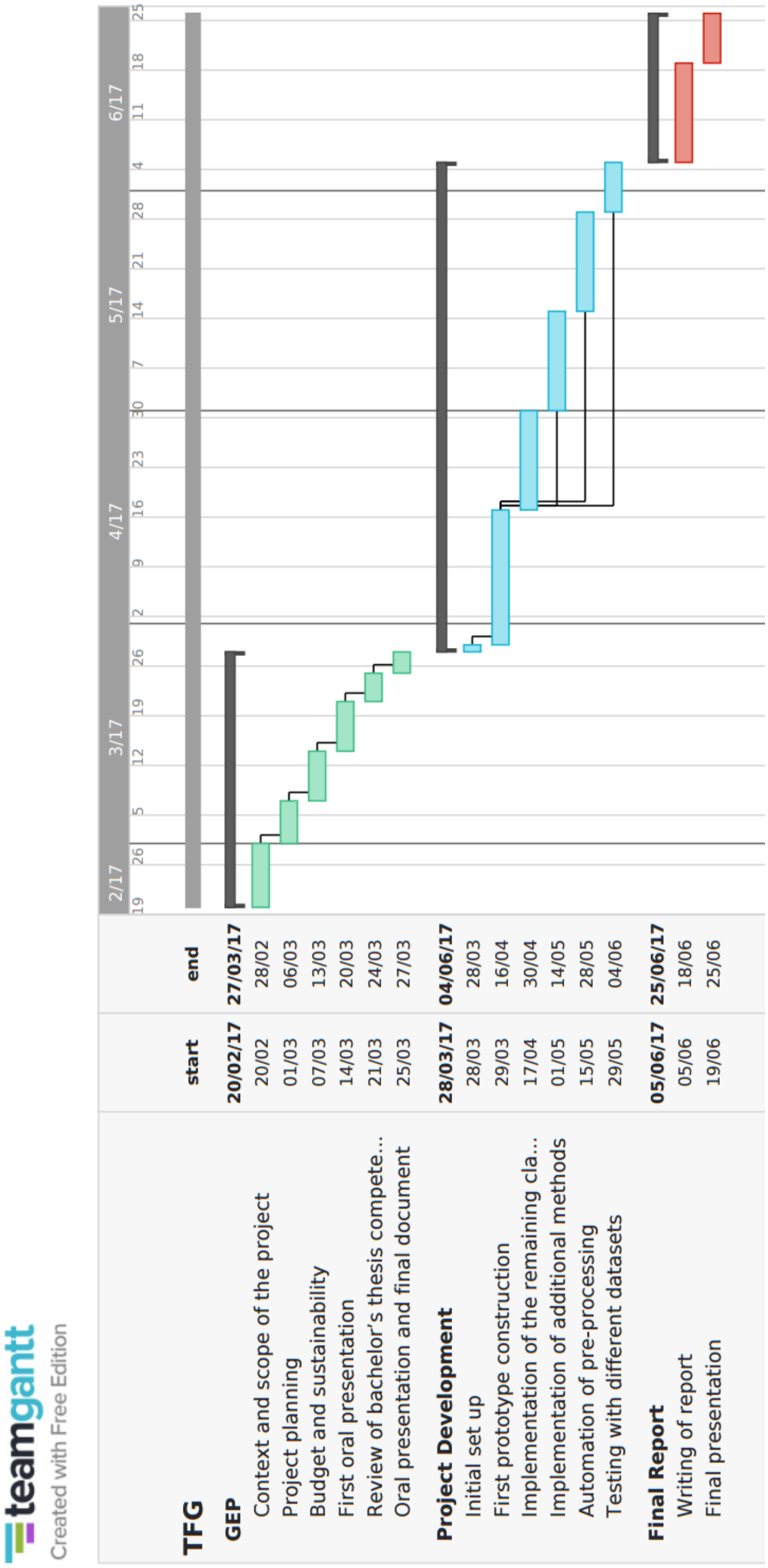


Figure 3.1: Gantt chart of the whole project (generated with *TeamGantt* [3])

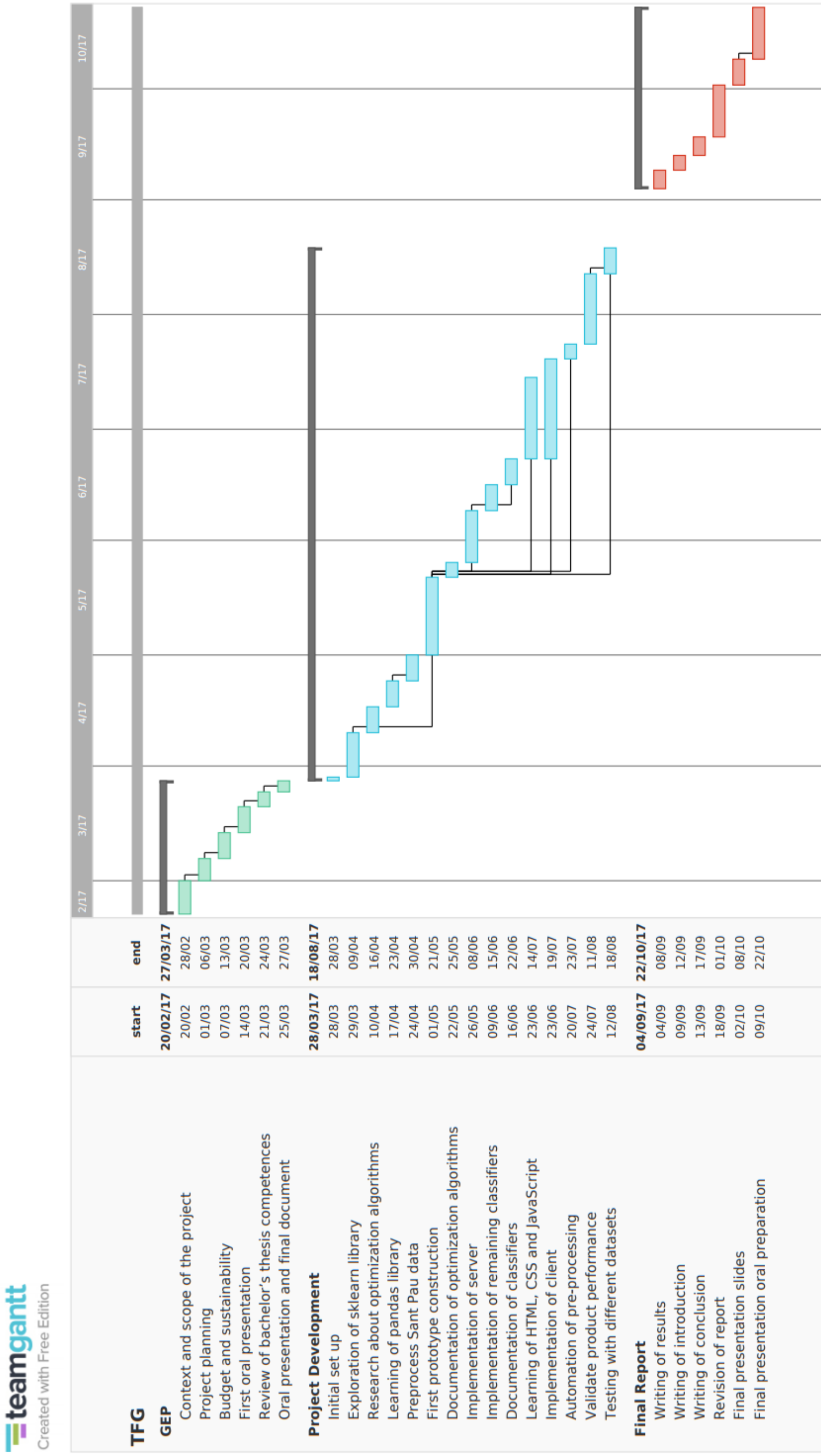


Figure 3.2: Final Gantt chart of the whole project (generated with *TeamGantt* [3])

4 Machine Learning

4.1 Basic concepts

4.1.1 Definition

Machine learning has been defined in many ways; we take the definition of [32], for example: *machine learning is a set of methods that can automatically detect patterns in data, and then use the uncovered patterns to predict future data or to perform other kinds of decision making under uncertainty* [32]. We can see that, in order to use machine learning, we need data and a set of techniques to process it. Moreover, not only do we want to analyze that data, we also want to predict future data. In other words, we need to extract knowledge from data using techniques that allow to do that in a semi-automated way. That means that a machine will learn the insights of the data and produce a model instead of a human doing it manually. However, it is semi-automated because that requires some decisions by a human in order for the process to be successful. For example, the technique that will be used or how the performance of the model is measured.

4.1.2 Types of machine learning

Machine learning can be divided into two main types: supervised and unsupervised learning.

Supervised Learning

In the predictive or supervised learning approach, we try to make predictions using data. The data consists of a series of training inputs. Each input is a dimensional vector of numbers which are called features or attributes. Furthermore, each training example has also associated a label or response variable. A classical example is to determine whether an email is spam or not. There is a specific outcome that we want to predict and in this case the features could be the content of the email. Finally, classification can be broken into two categories: classification and regression. In classification the label is discrete and we talk about classes. In regression the label is continuous (a real number). An example could be trying to predict the weight of a person given his height. In this thesis we will focus on classification.

Unsupervised Learning

In the unsupervised learning approach setting we want to extract interesting patterns from unlabeled data. That means that there is not a response variable associated with any training example thus there is no right or wrong answer. An example could be customer segmentation in a supermarket, that is finding groups or clusters of customers who exhibit similar behaviours. For that reason, unsupervised learning is often harder to understand and to evaluate, hence it is more difficult to automate.

4.1.3 Types of data

A dataset is composed of features or attributes that describe each sample. In general, features can be any kind of information that characterizes each sample: it can be a number, a string, a symbol or a code. Features can be divided among two main categories: numerical features and categorical features.

Numerical features

Numerical features can either be continuous or discrete:

- **Continuous**
  Continuous attributes are measured on a continuum scale. For example the length and width in centimetres of a box could take any value.

- **Discrete**
  Discrete attributes can be categorized into a classification and consists in a finite number of values that cannot be subdivided. For example the result of rolling a die can only take 6 values.

Categorical features

Categorical features can either be nominal or ordinal:

- **Nominal**
  Nominal features do not imply order. For example the color of a car can be red, blue or green and it clearly makes no sense to say that red < blue < green.

- **Ordinal**
  Contrary to nominal features, ordinal features can be ordered in a meaningful

way. For example T-shirt sizes are ordinal as we can state that XL > L > M > S.

4.1.4 Preprocessing

Preprocessing is one of the most important steps of data analysis. After having loaded a dataset, we must explore and detect . In *Data Preprocessing for Supervised Learning* [30] the different steps of preprocessing are described:

- **Instance selection**
  The first step is to decide which instances we will select from the dataset. Maybe there are too many and we want to select a subset of them or there is an under-represented class that we want to balance with the others.

- **Outlier detection**
  Outliers are data points that are very distant from the rest. An error in the measurement or a human error in the imputation can be responsible of outliers. They should be removed if possible as they might pollute the data and trick the classification algorithm.

- **Missing feature values**
  Real world data is usually incomplete, which means that for some samples the value of a feature might be unknown. There are many techniques to deal with missing values such as ignoring the sample, imputing the value or creating a new "unknown" value.

- **Discretization**
  The idea is to reduce the number of possible values of a continuous variable. For instance, assume that a feature is age in years. A possible discretization of this feature could be defining the categories: young, adult, old. That way we would end up with only three different possible values.

- **Normalization**
  Normalization is a transformation that scales data into a range of values. This is important as some algorithms might be influenced by the order of magnitude of a measure.

- **Feature selection**
  Feature selection is the process of selecting which features will be finally taken into account. Sometimes it is useful to remove redundant or irrelevant features as they might damage the performance of a classifier.

- **Feature construction**
  Feature construction is the process of creating new features from the ones that already exist in the dataset. For example, if we have the start and end time of an activity, we can derive its duration.

4.1.5 One hot encoding

Some machine learning algorithms cannot work with categorical features directly, they only accept numerical ones. For this reason we must convert categorical features into numerical features. One hot encoding is a technique that allows to perform this transformation. It represents categorical features as binary vectors, where each possible value of the categorical feature is mapped to an element of the vector. For each observation, this vector will have all zero values except the element of the categorical feature of that observation, which will be marked as one.

To fully understand this process, we will provide an example. Assume that we have an observation with a feature named color. Color can either be red, green or blue. Note that this feature is nominal, hence it would not be appropriate to assign integers to each color. A classifier would then assume that some colors are more similar than others. The one hot encoding technique discards the color feature and generates three new features: red, green and blue. Then, an observation with color equal to red would have a 1 in the red feature and 0 on the others.

One of the limitations of this technique is that it generates as many features as different values the categorical feature can take. That might generate many features and the classifier's performance might decrease. This is called the curse of dimensionality.

4.2 Evaluation metrics

If we want to compare the performance of multiple classifiers and decide which is the best one, we need some kind of metric to measure qualitatively how good a classifier is. Note that even if we have a single classifier, we might also want to measure its goodness in order to tune its hyperparameters. Moreover, the choice of the metric will influence the final choice of the algorithm.

4.2.1 Classification accuracy

This metric measures the number of correct predictions out of all predictions made. This metric is also called error rate since it measures the proportion of mistakes that the model makes. Accuracy can be computed using the following formula where $\hat{y}_i$ is the predicted class label for the *i*th observation and y_i is its real class. Moreover, I is an indicator variable that equals 1 when $y_i \neq \hat{y}_i$ and is 0 otherwise.

$$\textit{accuracy} = \sum_{i=1}^{n} I(y_i \neq \hat{y}_i)$$

Accuracy is not a good measure when the dataset is unbalanced, which means that there are many more samples from one class than from the other. Let's suppose that the 90% of the samples are from one class and the 10% left are from another. Then, a dummy classifier that always predicts the majority class will score an accuracy of 90% but it is easy to realize that in fact it is not useful. Another issue is that accuracy assumes equal cost for all prediction errors. However, the cost of different errors may differ a lot. For example, if we are predicting whether or not a patient has cancer, it is much worse missing a patient who actually has cancer than giving a false alarm to a patient who is healthy.

4.2.2 Confusion Matrix

Confusion matrix is not a metric per se but it is probably the most visual way of examining the results of a classifier. Instead of a single number summarizing its performance, the confusion matrix is a table that allows to visualize the predictions of the classifier in each class. Certainly that provides more information than a single number: with this table it is easy to identify what kind of errors are being made such as two classes being confused.

For a more concrete example we will focus on the following binary classification problem: a bank wants to know if a client will default or not on his credit. The positive category will be clients who default and the negative one the clients who do

		Predicted Class		
		Positive	Negative	Total
Actual Class	Positive	TP = 100	FN = 5	P = TP+FN = 105
	Negative	FP = 10	TN = 50	N = FP+TN = 60
	Total	P’ = TP+FP = 110	N’ = FN+TN = 55	165

Table 4.1: Confusion matrix of a binary classifier

not. This example can be found in [29], and the two kinds of errors that may happen are described in the following way:

> *In practice, a binary classifier such as this one can make two types of errors: it can incorrectly assign an individual who defaults to the no default category, or it can incorrectly assign an individual who does not default to the default category. It is often of interest to determine which of these two types of errors are being made. A confusion matrix [. . .] is a convenient way to display this information.*

In Table 4.1 we can see the confusion matrix resulting from this example. As we can see 165 predictions have been made which means that 165 clients have been tested. Out of those 165 clients, 105 defaulted and 60 did not. The classifier predicted that 110 defaulted and 55 did not thus the classifier made some mistakes. In order to refer to those mistakes we will introduce the following terminology (note that we refer to whole numbers and not to rates):

- **True Positives (TP)**: Number of correct predictions on the positive class.
- **False Negative (FN)**: Number of predictions where the classifier predicted that an instance was negative when it was positive in reality. It can be seen as a miss. Those errors are also known as Type II error.
- **False Positive (FP)**: Number of predictions where the classifier predicted that an instance was positive when in reality it was negative. It can be seen as a false alarm. Those errors are also known as Type I error.
- **True Negative (TN)**: Number of correct predictions on the negative class.

Many metrics can be derived from those four values. For example, the precision and the recall.

- **Accuracy**

  Accuracy measures how often the classifier is correct. A more detailed explanation can be found in the previous section.

$$\mathit{accuracy} = \frac{TP+TN}{TP+FP+FN+TN} = \frac{TP+TN}{P+N}$$

- **Recall, True Positive Rate (TPR), Sensitivity**

  The specificity is the proportion of real positive examples that are correctly classified by the classifier.

$$\mathit{recall} = \frac{TP}{TP+FN} = \frac{TP}{P}$$

- **Precision, Positive Predictive Value (PPV)**

  The precision measures how much of the predicted positive examples were actually positive.

$$\mathit{precision} = \frac{TP}{TP+FP} = \frac{TP}{P'}$$

- **Specificity, True Negative Rate (TNR)**

  The specificity is the proportion of real negative examples that are correctly classified by the classifier.

$$\mathit{specificity} = \frac{TN}{FP+TN} = \frac{TN}{N}$$

- **Fall-out, False Positive Rate (FPR)**

  The fall-out is the proportion of real negative examples that are correctly classified by the classifier.

$$\mathit{specificity} = \frac{FP}{FP+TN} = \frac{FP}{N}$$

4.2.3 Area Under ROC Curve (AUC)

In order to understand what is the AUC metric, we must first explain what ROC is. ROC stands for Receiver Operating Characteristic curve and it is commonly used to visualize the performance of a binary classifier, which is a classifier with two possible output classes.

The ROC curve is a plot of the True Positive Rate (TPR) against the False Positive Rate (FPR) for every possible classification threshold. Remember that the TPR answers the question: "When the actual classification is positive, how often does the classifier predict positive?" (ratio between true positives and all positives). Meanwhile, the FPR answers the question: "When the actual classification is negative, how often does the classifier incorrectly predict positive?" (ratio between false positives and all negatives). Those two values range from 0 to 1 [24].

The main advantage of this method over others is that the ROC curve visualizes all possible classification thresholds. If for example we look at misclassification rate, we are only considering a single threshold.

A classifier that does a very good job separating the classes will have an ROC that hugs the upper left corner of the plot. On the contrary, a very poor job separating the classes will have an ROC that is close to the diagonal line, which represents a dummy classifier that predicts randomly. In order to quantify the performance of a classifier, we cannot just compare two curves since usually it is not obvious which one is better. In fact, we need a numerical value (a score) in order to compare rigorously two curves. The score that is used is the AUC which stands for Area Under the Curve and it is the percentage of the unit box that is under the curve. In short, the AUC measures the ability of the model to discriminate between positive and negative classes.

To conclude this explanation we will now show an example of ROC curve. In Figure 4.1 we can see three different ROC curves in the same graphic. We can see that the one that is qualified as excellent is the one which has the bigger area under the curve since the area that is covered under the curve is the biggest one. The worthless curve is the diagonal one.

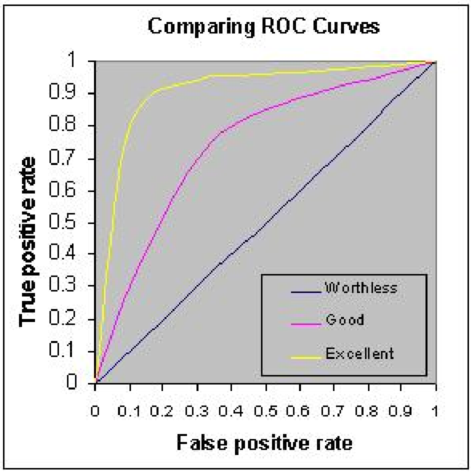


Figure 4.1: Comparison of three ROC curves [38]

4.3 Model selection

After having tried a series of candidate models for a given data we must decide which model is the best one. Remember that the goal of machine learning is to estimate the performance on future data so we are looking for a model that maximizes the performance on that future data, not the one that was used during the construction of the model. The main problem is that maximizing the score on training data rewards overly complex models that do not generalize well. This behaviour is called overfitting, a concept that will be explained in this section. Apart from this we will also introduce some evaluation procedures which are used to evaluate the goodness of a model such as the train/test split or cross-validation. Finally, we will introduce the bias-variance trade-off, which is a very relevant concept in model evaluation.

4.3.1 Train/test split

One of the easiest ways to compare models is to split a dataset into two pieces: a training set and a testing set. Then we train the model on the training set and we test the model on the testing set, also known as hold-out set. This way, the model is tested on unseen data, also called out-of-sample data. Indeed, we are simulating more accurately how well a model is likely to perform on future data. In Figure 4.2 we can see how this procedure works. We call training score to the score that the model has in the training data whereas the test score is the score obtained in the test data. The cost of this method is that the hold-out data is removed from the training process which means that the model has less examples to learn from. Moreover, the testing score is a high variance estimate of the out-of-sample performance which means that it depends a lot on which observations happen to be in the testing set. Finally, it is also important to avoid adjusting the model using the testing set since by doing that the testing data would no longer be unseen.

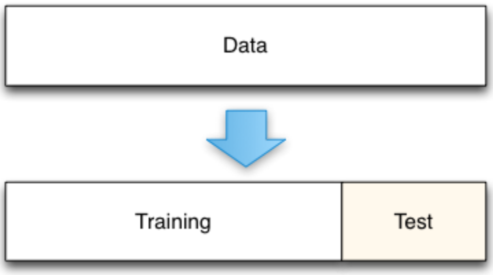


Figure 4.2: Holdout data split [26]

4.3.2 Cross-validation

Cross-validation is a technique that tries to handle the issues of the train/test split method. The idea is to randomly divide the data into K subsets of equal size (known as folds), holding out each one while training on the rest, testing each learned classifier on the examples it did not see and finally averaging the results to know how well the particular model does [22]. In Figure 4.3 we can see an example of this method with $K = 5$. Note that for each iteration, every observation is either in the training set or the testing set, but not both. Also, every observation is in the testing set exactly set. The advantages of this method over the previous one is that it provides a more accurate estimate of out-of-sample score and a more efficient use of data. However, it is K time more slower than train/test split. In fact, we repeat exactly K times the train/test split procedure. This is especially important when datasets are huge and training a model can take a long time. Another condition that has to be taken into account is to use stratified sampling when creating the folds for classification problems. If we are dealing with a binary classification dataset and there are only 20% examples of one class, then we must ensure that the folds maintain that proportion. Finally, the typical values that are used for K are 5 or 10 [29].

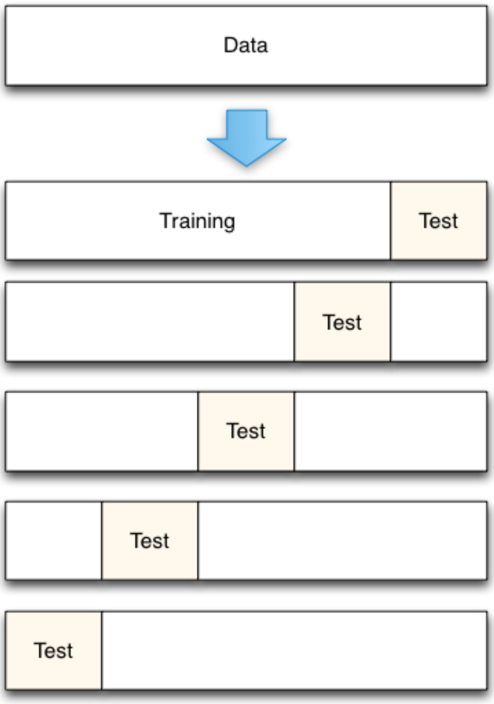


Figure 4.3: 5-fold cross-validation [26]

4.3.3 Overfitting

Overfitting occurs when we generate a model that is too complex and hence does not generalize well. Indeed, the model has learned all the noise in the data. Usually, the training results will be very high but the testing results will be very poor. When that occurs, it is usually because of overfitting. In Figure 4.4 we can see an example of overfitting. There are two models that try to predict the classes of the points and we can see their decision boundaries on the diagram: the black curve misses some of the training observations but it captures the overall tendency. Meanwhile, the green curve does a perfect job on the training data since it does not miss a single observation, but it is likely that it will misclassify many future examples. In two dimensions it is easy to detect that a model is overfitting by looking at the complex shape of the decision boundary. Nonetheless, when the number of features hence dimensions is higher, it is impossible to visualize what the classifier is doing. In fact, it is said that if people could see in high dimensions machine learning would not be necessary [22], as it would be obvious to visualize what the model is doing.

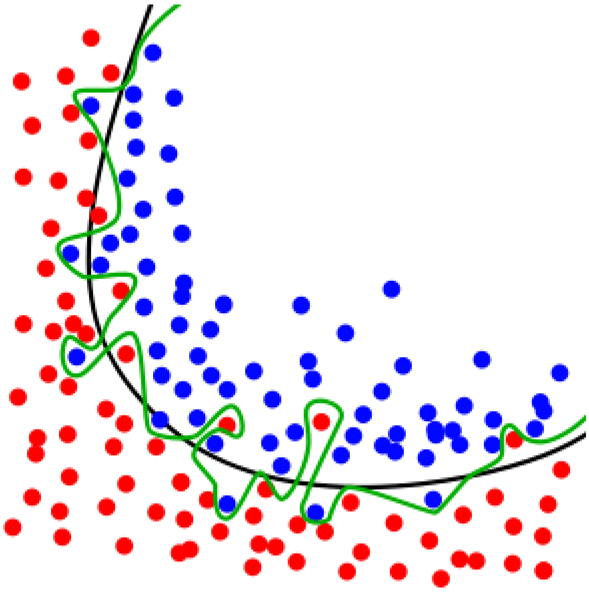


Figure 4.4: Diagram where each point represents an observation and the color its class. The black and green curves are two different decision boundaries that try to classify the points [10]

4.3.4 The bias and variance trade-off

To better understand overfitting it is useful to introduce the concepts of bias and variance. Bias is the tendency to consistently learn the same wrong thing. It happens when we are being to restrictive with the model that we are selecting, we make too strong assumptions over the data. On the other side, variance is the tendency to learn the noise of the data instead of the real signal. In Figure 4.5 we can see a graphic representation of this trade-off. On the left, the model complexity is low and there is high bias and low variance: our model is underfitting. On the right, the model complexity is high and there is a low bias a high variance: our model is overfitting. In the middle, we can find the sweet spot which is the optimum model complexity.

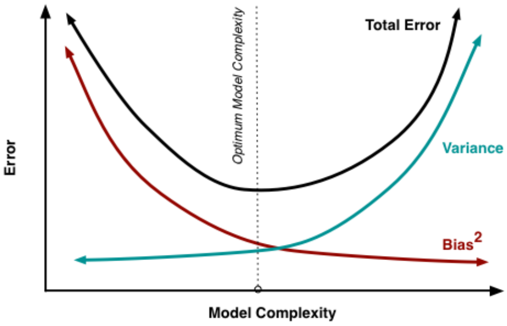


Figure 4.5: Bias and variance contributing to total error [26]

5 Classifiers

This chapter aims to give a background of the classifiers that will be used in this project. Classifiers are algorithms that compute a learning function that maps input variables to output variables. In machine learning, input variables are called features and the output variable is the class or label. In order to learn that function, classifiers use data. The goal of a classifier is to approximate as best as possible the functional form of the underlying function, that is the real function that shapes the data. Different algorithms make different assumptions or biases about the form of the function and how that function is learned. Next we will define the two main categories of machine learning algorithms: parametric and non-parametric models.

A parametric model simplifies the learning process at the cost of limiting what can be learned. Indeed, it summarizes data with a set of parameters of fixed size, independent of the number of training examples [35]. Examples of parametric models are Naive Bayes or Logistic Regression. The benefits of those algorithms is that they are easy to understand, fast to train and do work with less training data. The limitations are the constraints of the form of the functions, usually it will end up with a model with high bias.

Non-parametric algorithms do not make strong assumptions about the form of the underlying function. Because of that they are free to learn any functional form from the training data. In fact, they are able to fit a wide number of functional forms. Examples are non-parametric algorithms are k-Nearest Neighbors or Decision Trees. The benefits are they can build a more complex model hence improve the performance of the classification. The limitations are their interpretation, their slower training time and that they are more prone to overfitting the training data.

In the next sections we will describe the classification algorithms that have been used in this project.

5.1 Decision Trees

Decision trees are widely used in classification problems. The goal of a decision tree is to learn rules from the features of the data and classify new examples using those rules. As its name suggests, decision trees can be represented in a tree form. They encode a series of binary questions about the value of a feature and make decisions based on the answer. The usefulness of each question depends on how well it splits the data in labels. To understand that, we must introduce the concept of entropy.

Entropy is a real number between 0 and 1 that measures the impurity of a set of examples. If the entropy of a set is 0 it means that all the examples are from the same class. On the other hand, if the entropy is 1 it means that the examples are evenly split among classes. The best question is the one that splits in a better way all the examples. For that we define the concept of information gain, which derives from entropy.

$$\text{Information Gain} = entropy(\text{parent}) - [\text{weighted sum}]entropy(\text{children})$$

Most of the hyperparameters of trees control its size to avoid overfitting. In fact, without constraints a tree could reach a 100% accuracy by making a leaf for each observation. Obviously this model would not generalize well thus it would overfit the data.

5.2 Random Forest

In the previous section we have introduced decision trees. A limitation that decision trees have is that they can suffer from high variance with the training data if they are not properly pruned. A possible solution is to build multiple decision trees instead of a single one, which is called bagging. The main idea of bagging is to combine multiple classifiers of the same kind trained on different samples from the same dataset. The samples, called "bagging replicas", are obtained by choosing with replacement a desired number of instances from the dataset.

A problem of decision tree algorithms is that they make a greedy choice at each split. That means that, even if we train a lot of them, they will end up being similar hence the predictions will also be similar. To solve that issue we can limit the features that the greedy algorithms can use. More precisely, at each node to be split, the algorithm selects randomly a number of candidate attributes (usually much smaller than the total number of attributes), and the "best" attribute among this set is chosen, not the overall "best". With that, we obtain more diverse trees which are uncorrelated to each other [20].

5.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) algorithm has the particularity that the model its representation is the entire training dataset. Indeed, there is no learning required. Nevertheless, evaluating the performance of the algorithm is expensive since for each example we must compute its neighbors. Efficient implementations can help to store the data in complex structures such as k-d-trees [27].

To make predictions on a new example, KNN searches the k most similar instances (the neighbors) and selects the class of the majority of those K instances. In order to avoid ties, k is usually an odd number. The key of this algorithm is how to measure the similarity or dissimilarity between two examples. To summarize the value, we need a function that given two examples returns a number that tells how similar they are. For instance, Euclidean, Manhattan or Hamming distances can be used. Next we will show the euclidean distance between two examples x and y across all input attributes j (x_j is the value of the attribute j of x).

$$dist(x, y) = \sqrt{\sum_{j=1}^{n} (x_j - y_j)^2}$$

KNN works well when the number of features is small but struggles when this number increases. In high dimensions, points that are similar are still separated by a long distance, so it gets harder to distinguish them, particularly if there are attributes that are irrelevant for deciding the class. In order to avoid that, it is recommended to lower the dimensionality by performing some kind of feature selection. It is also advisable to rescale the data to avoid the dominance of one feature respect the others.

5.4 Naive Bayes

Naive Bayes is an algorithm based on the following principle: select the most probable class of a new example given the data that we have from the past ones. In order to calculate the probability of a hypothesis given our prior knowledge, we can use the Bayes' theorem. We have a probability model and we update our beliefs with evidences.

Let $F = \{f_1, f_2, \ldots, f_n\}$ be the set of features of an example and c its predicted class.

$$\mathcal{P}(c \mid F) = \frac{\mathcal{P}(F \mid c) \cdot \mathcal{P}(c)}{\mathcal{P}(F)}$$

- $\mathcal{P}(c \mid F)$ is the posterior probability of an example belonging to class c given that he has features F.
- $\mathcal{P}(F \mid c)$ is the probability of an example having features F given that he belongs to class c.
- $\mathcal{P}(c)$ is the prior probability of class c being true.
- $\mathcal{P}(F)$ is the probability of features F.

This formula allows to calculate the posterior probability $\mathcal{P}(c \mid F)$ from the prior probability $\mathcal{P}(c)$.

In order to compute the $\mathcal{P}(F \mid c)$ we make a strong assumption: the features are assumed to be conditionally independent given the class. Even if this assumption is unlikely in real data, the algorithm still performs well. We compute its value the following way:

$$\mathcal{P}(F \mid c) = \prod_{j=1}^{n} \mathcal{P}(f_j \mid c)$$

Finally, to use this model we just have to compute the posterior probability of a new example for each possible class and select the class that has the maximum probable hypothesis. This is called the maximum a posteriori estimation (MAP).

$$\text{MAP}(c) = max(\mathcal{P}(c \mid F))$$

5.5 Logistic Regression

Even if regression appears in the name of this algorithm, logistic regression is a classification algorithm. It is based on the logistic function, also called sigmoid function f. In Figure 5.1 we can see its shape. We note that all the inputs are transformed into the range of [0, 1].

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

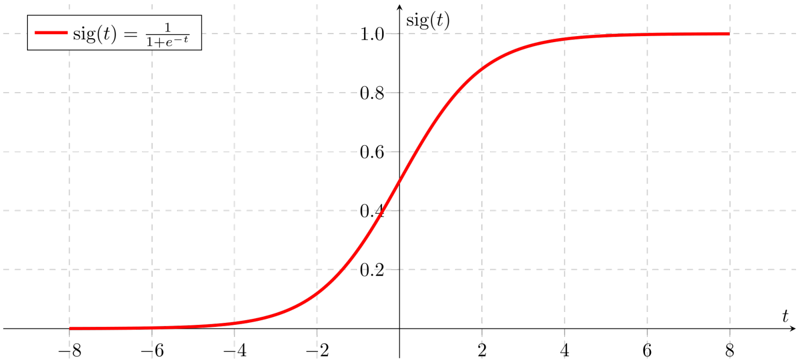


Figure 5.1: Sigmoid function [11]

We will explain how this algorithm works in a binary classification problem where there are two classes: class 0 and class 1. In order to use it in a multi-classification one, we could use a one versus all approach. The logistic regression model takes a vector of real numbers as input and outputs the probability of belonging to a class. Let x be an observation and x_j be the value of the feature j. We can compute the value y the following way:

$$y = \theta_0 + \sum_{j=1}^{m} \theta_j * x_j$$

Then, the probability of observation x belonging to class 1 is:

$$\mathcal{P}(\text{class } 1|x) = f(y) = \frac{1}{1 + e^{-y}}$$

Remember that the logistic function returns values in the range of [0, 1] so it will always return a valid probability.

6 Hyperparameter optimization

This chapter covers the topic of hyperparameter optimization in learning algorithms. This optimization deals with the problem of choosing a hyperparameter setting that optimizes the performance of a given algorithm. In the following sections a detailed definition of what hyperparameters are will be provided and then four algorithms for hyperparameter optimization will be described. Finally, a comparison between two of those algorithms will be performed.

6.1 Definition

In machine learning, we need to distinguish between two kind of parameters: model parameters and hyperparameters. Model parameters are the ones that are learned from the training set when training an estimator. In contrast, hyperparameters cannot be learned within the estimator directly. They are set when creating the estimator and define higher level concepts about the model such as complexity, or capacity to learn. Usually, they control the trade-off between model complexity and model generalization. In general, the more complex a model is, the less it generalizes since it will overfit.

A classical example of hyperparameter is found in KNN, which is very popular in classification analysis. In Section 5.3 we have explained how it works. To sum up, it tries to find the k most similar observations to the one that we are trying to predict and assign the majority class of those k observations to it. In this case, the value of k must be provided to the algorithm, that is, the algorithm is not able to find by himself which is the appropriate value of k. For that reason, a user or another algorithm must take care of trying different values of k and selecting the one that maximizes a certain scoring function.

6.2 Manual search

Traditionally this is one of the most used techniques and this tuning is often seen as a "black art" requiring expert experience and rules of thumb [37]. A major drawback of manual search is that is usually inefficient, time consuming and difficult to reproduce results. Moreover, a deep knowledge of the internals of the algorithm that is being used is required in order to be able to successfully tune its hyperparameters. One must know exactly what controls each hyperparameter and how they interact with each other.

By definition, manual search cannot be automated hence its implementation will not be considered in the scope of this project (remember that we aim to avoid interaction with the user). Nevertheless, this technique will still be useful to validate the results produced by our program.

6.3 Grid search

The most trivial way of performing hyperparameter optimization is to take a brute force approach with respect to the search space. Grid search is based on this idea and consists of an exhaustive searching through a defined set of parameter values.

For example, let's suppose that we have a function $f : \mathbb{N} \times \mathbb{N} \to \mathbb{N}$ and that we want to optimize using grid search. Let x and y be the parameters of f. First of all we will define a search space, that is select a finite set of values for each parameter, say $x \in \{1, 2, 3\}$ and $y \in \{4, 5\}$. We will then proceed to generate all the possible combinations of pairs (x, y) and evaluate f for each one. In this case, we will evaluate $f(1, 4)$, $f(1, 5)$, $f(2, 4)$, $f(2, 5)$, $f(3, 5)$ and $f(4, 5)$. Finally we will select the pair that optimizes the value of f. Note that the number of generated pairs follows the formula $|x| \times |y|$, in this case $|x| = 3$ and $|y| = 2$ so six pairs were generated. Therefore, it is important to keep the search space as small as possible because otherwise too many combinations are generated. This is a problem if the space is high dimensional or the set of values of a parameter to test is high.

To conclude this section, we will discuss about advantages and disadvantages of grid search. On the one hand, grid search is easy to implement and reliable in low dimensional spaces. On the other hand, for most data sets only a few of the hyperparameters really matter, but different hyper-parameters are important on different data sets. Grid search does not take advantage of this fact as it gives equal importance to each parameter. In fact, *the number of wasted grid search trials is exponential in the number of search dimensions that turn out to be irrelevant for a particular data set* [19]. The Sklearn library provides a *GridSearchCV* class implementing and parallelizing grid search with cross-validation, so it is ready to use.

6.4 Random search

As we have seen in the previous section, grid search is a potentially expensive method. An alternative to this method is random search which performs a randomized search over the parameters. In this technique each setting is sampled from a distribution over possible parameter values for a fixed number of times.

The key point to understand why this technique works well is the concept of *low effective dimensionality*. Usually, in high dimensional spaces there are dimensions that have more impact than others in the function that we want to optimize. For example, if we have a two dimensional function whose value can be approximated by only one of those dimensions ($f(x, y) \approx g(x)$) then we can say that f has a low effective dimension.

Compared to grid search, random search has two main benefits that are described in [7]:

- A computation budget can be chosen independently of the number of parameters. For example, we can set a number of sampling iterations or a time limit. Doing that in grid search is much more dangerous since the space that has been explored is not random and we might miss a huge area of the search space.
- Adding parameters that do not influence the performance does not decrease efficiency. Indeed, sampling a fruitless parameter is cheap whereas the number of combinations that will create that given parameter is exponential.

This algorithm has a parameter which indicates how many times we will sample the hyperparameters and is called number of iterations. Finally, implementing this algorithm is feasible with *sklearn* library since it provides a *RandomizedSearchCV* class that implements and optimizes randomized search with cross-validation. Moreover, it supports assigning a continuous distribution to sample each parameter such as uniform, normal or exponential distributions.

6.5 Bayesian Optimization

Both grid search and randomized search are said to be *non-adaptive* [19]: they do not vary the development of the experiment by taking into account any results that are already available. That has the advantage of making those algorithms embarrassingly parallel but the disadvantage of testing some settings that could have been discarded after seeing some results. Bayesian optimization is an algorithm that takes into account previous results to optimize an unknown black-box function.

In [37] Bayesian optimization of machine learning algorithms is studied and it is shown that this algorithm can even improve expert-level optimization. The idea of this algorithm is to minimize a function on a finite space. To achieve that, it constructs a probabilistic model of that function which is used to decide which point will be evaluated next and this model is updated after each evaluation. That is, after each call to the function, the result is going to update the beliefs about the model that it consider the function has.

In order to use this algorithm in this project, a search has been carried out and a library named *hyperopt* [5] has been found. This library implements Bayesian optimization but the problem is that it is in a very low level way. Indeed, only a function and a search space must be provided and the library minimizes that function within that search space. *sklearn* is being used in our project and we are limited and coupled with that library. The problem with *sklearn* is that it is a high level library while *hyperopt* is a low level one, hence the integration between them is not straightforward. For this reason it has been decided that, even if this algorithm is considered to be the best one, it will not be used in the scope of this project.

6.6 Comparing grid and random search

One of the main problems of hyperparameter optimization is that any dataset has a different optimal setting of hyperparameters. Hence a magic setting that works with any dataset will never exist. Because of that we are forced to explore that space as we cannot know in advance which setting will be the optimal for a given dataset. In [19] it is empirically and theoretically shown that random search is more efficient for hyperparameter optimization than grid search.

In order to understand why they came to that conclusion we can look at Figure 6.1. We see the layout of a random search and a grid search over the same function that we want to optimize. Note that this function has a low effective dimensionality as one parameter is more important than the other one. Grid search points give an even coverage of the space but they cover poorly the projections onto either x or y axes. In fact, they only cover 3 distinct points. Meanwhile, random search points are less evenly distributed over the space but they greatly cover each of both axes. In fact, they cover 9 distinct points of each axis. Because of that we can better explore the most important feature and do not waste trials exploring irrelevant features.

This project will implement those two algorithms and we will study which one provides the better results. Something that has to be considered when comparing those two algorithms is to execute them under the same conditions. In this case it seems reasonable to set the number of iterations of randomized search equal to the number of combinations that grid search explores.

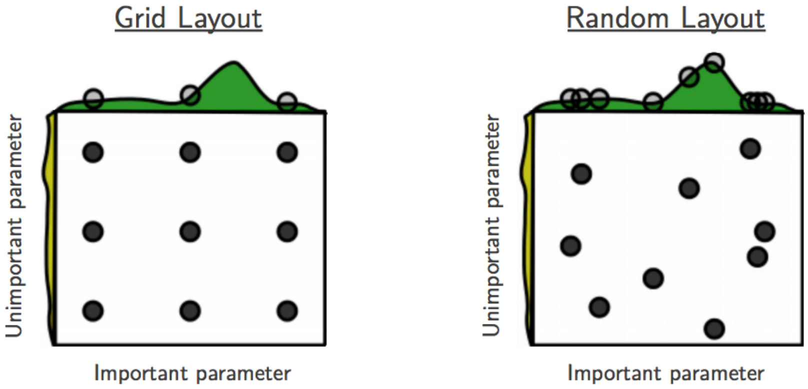


Figure 6.1: Grid and random search of nine trials for maxmimizing a function $f(x, y) = g(x) + h(y)$. The value of $g(x)$ is shown in green and the value of $h(y)$ in yellow [19].

7 AutoML design and development

This chapter describes the design of the software built in this project. This project can be divided into three main components: the classifier, the server and the client. We will proceed to explain how each component works, which technologies it uses and what they do.

7.1 AutoML classifier

This is the most important module of the project as is the one which performs the training of the dataset. This module receives a dataset as input and outputs a model that predicts new examples of the dataset.

7.1.1 Technology

The classifier has been build in Python. This language is suited for building fast prototyping, especially in machine learning environments. The libraries that have been used are pandas, numpy, sklearn and joblib. Pandas provides a powerful interface to read a csv file and perform transformations on a dataset. Once the preprocessing has been completed, data is translated into numpy arrays since that is the input format of sklearn. Finally, joblib is a library designed to persist Python data. The particularity of this library is that it is optimized for large numpy arrays hence it is suited for this project.

7.1.2 UML diagram

In Figure 7.1 we can see the UML diagram of the most important classes of the module. Those classes are the ones that are related with the classifier representations. As we can see, there are three classes. The first one is *AutoClassifier* which is composed of *ClassificationAlgorithm*. The latter is an abstract class that defines the contract that any classification algorithm must provide. Finally, *AutoNaiveBayesClassifier* is an example of a possible implementation of *ClassificationAlgorithm*.Each class has its own attributes and methods, which will be described below.

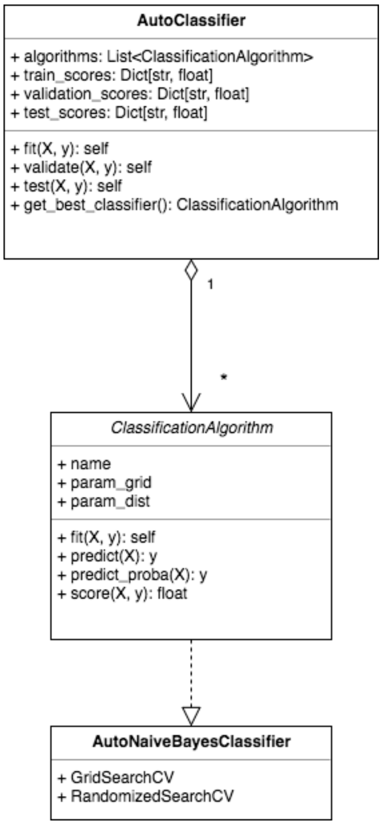


Figure 7.1: UML diagram of the module

AutoClassifier

First of all, the *AutoClassifier* class is the one that is responsible of storing all the different classifiers and selecting the best one. Moreover, it also records the scores for each classifier, in order to recognize the differences among them.

We will start by enumerating the different attributes that has this class.

- *algorithms*: List of classification algorithms that the will be tested.

- *train_scores*: Score of the training set. The key is the name that identifies each classifier and the value is the score obtained after fitting the training set.

- *validation_scores*: Score of the validation set. The key is the name that identifies each classifier and the value is the score of the validation set.

- *test_scores*: Score of the test set. The key is the name that identifies each classifier and the value is the score of the test set.

Next we will explain which operations perform each method.

- *fit(X, y)*: Method to process training set. It trains each classifier with the input dataset *X* and *y*, where *X* are the features and *y* is the class.

- *validate(X, y)*: Method to process validation set. It computes the score of the input dataset for each classifier from *algorithms* list and selects the best one.

- *test(X, y)*: Method to process the test set. It computes the score of the input dataset for each classifier.

- *get_best_classifier()*: Returns an instance of the best classifier.

ClassificationAlgorithm

ClassificationAlgorithm is an abstract class that defines the contract that any algorithm must implement. This contract is inspired in the API of *SKLearn* objects [8]. We can see in Figure 7.1 that this abstract class forces to implement some attributes and methods.

We will begin by describing what the attributes consist of. Any classifier must provide the following attributes in order to be valid:

- *name*: Name that identifies the classifier

- *param_grid*: A grid of parameters that will be used by grid search. It is a dictionary where each key is a hyperparameter and the value is a list of possible values that will be explored.

- *param_dist*: A distribution of parameters that will be used by random search. It is a dictionary where each key is a hyperparameter and the value is a sampling distribution. For each iteration, a value will be sampled according to that distribution.

Similarly, any classifier must implement the following methods in order to be executable:

- *fit(X, y)*: Method to learn from data. *X* are the features and *y* is the class. All those values must be numeric.
- *predict(X)*: Method to predict the class of a series of samples.
- *predict_proba(X)*: Method to predict the probability of each class. It returns the probability of the sample for each class in the model.
- *score(X, y)*: Method to compute the score of the classifier on the dataset defined by *X* and *y*. The score metric that is used is the same that the one which was defined when instantiating the classifier.

7.1.3 Hyperparameter tuning

For each classification algorithm, we will list the hyperparameters that we explore and tune.

Decision Tree

- **criterion**
  This parameter controls the function that measures the quality of a split. It can be either *gini* or *entropy*.
- **min_samples_leaf**
  This parameter sets the minimum number of samples required to be a leaf node. If it is equal to 1, we allow the tree to create a leaf for each sample, hence it would have a perfect accuracy on training data. However, it is easy to see that the corresponding model would suffer from overfitting.

Random Forest

Random forests share the same hyperparameters than decision trees except that in this case we train many decision trees, so there is an extra hyperparameter that controls how many trees we end up training.

- **n_estimators**
  This parameter selects the number of trees in the forest.

Logistic Regression

- **C**
  This parameter controls the regularization.

- **solver**
  This parameter selects which solver will be used. There are different implementations that may converge faster or that are more appropriate for multi-class problems.

Naive Bayes

- **alpha**
  This parameter controls the Laplace smoothing. This solves the problem of assigning a zero probability to a sample that was not observed in training.

- **fit_prior**
  This parameter is a boolean value that controls whether or not the classifier will learn class prior probabilities.

K-Neighbors Classifier

This algorithm has the particularity that, in order to work properly, features should be scaled. The reason behind that is that we compute distances between data points and feature with different scales will have a different impact on that distance. For that we use *RobustScaler* which is a Sklearn object that scales features using statistics that are robusts to scalers. It is important to note that we only scale the training data and not the whole data. If we did that, we would be cheating since we would use test data information and we are not allowed to do that. Thus, we set the scaling parameters only with training data and when we use the classifier to predict new data, we apply the same scaling operation that we applied on the training data.

- **n_neighbors**
  This parameter controls the number of neighbors that will be used (the parameter k).

- **weights**
  This parameter controls the weight function used in prediction. It two values:

uniform or *distance*. The former weights all points equally while the latter weights points by the inverse of their distance. That means that closer points will have a greater influence than further ones.

- **metric**
  This parameter is important because this classifier's performance heavily relies on this metric [41]. It selects the distance metric that will be used. The two more common ones are Manhattan distance and Euclidean distance. However, we will also try Minkowski distance, which is a generalization of both of the previously described distances.

7.1.4 Pipeline

In Figure 7.2 we can see which steps this program automates. In any classification problem, we start with raw data. Then starts an iterative process consisting on data cleaning, feature selection and construction. This is complicated to automate because it usually requires domain knowledge and the supervision of a human. Moreover, it is hard to determine when this task should end as new features can always be derived from existing ones. Building a program that automates all this process certainly is very complex. Our program starts automating the model selection step. Indeed, many classification algorithms exist and there is not one that works for every dataset. That is why we must try them all and select the one that provides the best performance. We also have to explore the parameters of those algorithms and finally validate their performance.

Model selection

We have decided to use the following algorithms: decision trees, random forest, KNN, naive bayes and logisitic regression. For each of those five algorithms, we also define the parameter space that we will explore. We favored those algorithms since they are the most classical ones and they have been extensively studied. There is abundant literature about them and they are already implemented in Sklearn. Moreover, their simplicity is also convenient since in order to tune the hyperparameters of an algorithm one must know the insights of it. Lastly, complex algorithms are harder to tune since it is easier to overfit.

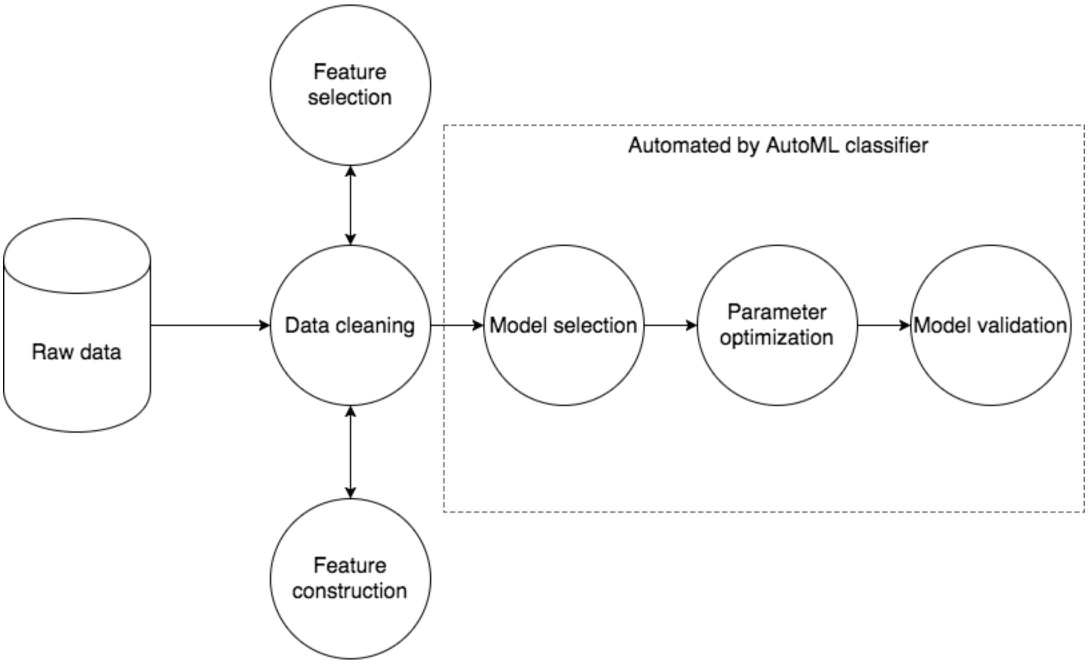


Figure 7.2: AutoML classifier pipeline

Parameter optimization

We have included two methods that can be used in order to search in the parameter space: grid search and random search. The search method can be defined in the dataset's configuration file. In the case of grid search, we explore each possible combination of parameters and we compute the performance of the classifier using cross validation. In the case of random search, we sample each parameter and compute the performance of the classifier using cross validation.

Model validation

The final step is to decide which model provided by each algorithm do we finally select. In order to do that we run those models against a validation set, which is data that none of the algorithms have seen. This is a good measure of generalization. We select the model that provides the higher score. Finally, we test that model against the test set to confirm that indeed it generalizes well and that the score on the validation set was not chance. Ideally, the scores of the validation and test set should be similar since that would mean that the model generalizes well.

7.1.5 Preprocessing

In the previous section we said that this program did not automate preprocessing. But even if it does not fully automate this process, it performs some basic operations that are needed to ensure that the program can handle general datasets. A limitation of Sklearn library is that classifiers only work with numerical data. However there are many datasets which have categorical features. Not supporting those datasets would hurt the ability to generalize. To solve this issue, we use one hot encoding. In Section 4.1.5 we explained how this method works and what are its limitations. To deal with the curse of dimensionality problem, we will limit the amount of new features that we create to the 50 more frequent values. That means that if a feature has more than 50 different values, we will use the one hot econding technique to encode those values and will ignore the rest.

Another basic operation that is performed is the handling of missing values. Again, we are forced to deal with them since Sklearn library does not allow null values. That means that we have to impute those values. Many algorithms and techniques exist to do that and we decided to use the more simple ones due to its ease of implementation and time complexity. To impute missing values we must distinguish two features: numerical features and categorical features. For the former, we just replace the missing value with the mean of that feature. For the latter we replace it with the mode, which is the most frequent value.

7.2 Server

The server acts as an intermediate component between the client and the classifier.

7.2.1 Technology

The server has been designed using Python language and the framework that is used is Flask. Flask is a popular web microframework that allows to build fast prototypes. In the scope of this project, we used Flask to build a server which exposes a series of endpoints that the client hits to retrieve useful information. Requests are sent using *HyperText Transfer Protocol* (HTTP) with *GET* and *POST* methods. Responses are sent in *application/json* format to ease its manipulation in the client side with Javascript.

7.2.2 Endpoints

An endpoint is a URL that can be used to communicate with an API. Next we will list the endpoints that the server exposes to the client. If the request has a body, we will show an example of it. Otherwise we will omit the request in the specification. Note that in order to avoid unnecessary *POST* requests, we add variable parts to the URL. During the description we will display those parts using angle brackets "< >". For instance, <*dataset*> means that this part will be replaced by the identifier of the dataset. Indeed, *Flask* handles that syntax in a convenient way as we can see in the following code snippet. Finally, to test those endpoints we have used *Postman* which is an application that allows to easily generate HTTP requests to the server.

```
@app_config.route('/config/read/<dataset>', methods=['GET'])
def read_config(dataset: str) -> Json:
```

The response from the server is always a *JSON* with the structure below. That confirms to the client that the request has been successfully processed. If there is any kind of error, the response message will be replaced by the reason that caused said error. If the response has no additional content, we will omit it in the specification.

```
Response
{
   "response": "OK"
}
```

Upload dataset (POST /upload/dataset)

Upload a dataset. The file is in the header of the request.

```
Request
Key:"file"; Value: "iris.csv"
```

List uploaded datasets (GET /upload/list)

List uploaded datasets and basic information about them. That information cosists of the number of features, the number of observations and whether the dataset has been trained or not.

```
Response
{
   "datasets": {
      "adult": {
         "features": 14,
         "observations": 32562,
         "trained": false
      },
      "iris": {
         "features": 4,
         "observations": 150,
         "trained": false
      }
   },
   "response": "OK"
}
```

List uploaded datasets (GET /train/<dataset>)

Train the dataset identified by <dataset>. After this process the server obtains the model which can be used to make predictions.

Results (GET /results)

List results of trained datasets. It provides, for each trained dataset, the name of the best classification algorithm along with its train, validation and test score.

```
Response
{
   "adult": {
      "algorithm": "Decision Tree",
      "test_score": 83.65384615384616,
      "train_score": 85.47717842323651,
      "validation_score": 87.5
   },
   "wine": {
        "algorithm": "Random Forest",
        "test_score": 100,
        "train_score": 100,
        "validation_score": 96.29629629629629
    },
    "response": "OK"
}
```

Read configuration (GET /config/read/<dataset>)

Retrieve configuration of <dataset>. It returns the contents of the configuration file of the dataset.

```
Response
{
    "config": {
        "accuracy": "yes",
        "data_file": "iris.csv",
        "f1": "no",
        "folds": "5",
        "grid_search": "yes",
        "iterations": "2",
        "jobs": "-1",
        "missing_value": "?",
        "precision": "no",
        "random_search": "no",
        "roc_auc": "no",
        "seed": "1",
        "separator": ",",
        "target": "class"
    },
    "response": "OK"
}
```

Write configuration (POST /config/write/<dataset>)

Write configuration of <dataset>. It allows to modify the configuration file of the dataset.

```
Request
{
   "target": class,
   "folds": 10
}
```

Predict (POST /predict/single/<dataset>)

Predict a single instance of <dataset>. The request contains the name and value of each feature, while the response contains the predicted class.

```
Request
{
   "sepal_length": "5.1",
   "sepal_width": "3.5",
   "petal_length": "1.4",
   "petal_width": "0.2"
}

Response
{
   "prediction": 0,
   "response": "OK"
}
```

Predict (GET /predict/list)

List trained datasets.

```
Response
{
   "datasets": [
      "adult",
      "iris",
      "wine"
   ],
   "response": "OK"
}
```

Predict (GET /predict/features/<dataset>)

Retrieves features and an example of possible values of <dataset>.

```
Response
{
   "features": [
      "sepal_length",
      "sepal_width",
      "petal_length",
      "petal_width"
   ],
   "target": "class",
   "values": [
      "5.1",
      "3.5",
      "1.4",
      "0.2"
   ],
   "response": "OK"
}
```

7.3 Client

The client is the module that the user interacts with. It is charged to read data and to display the results to the user. In this project, the user interface is a web page.

7.3.1 Technology

To build this web page, we have used HTML and CSS for the layout. Those languages are both markup languages, that is they only control the layout of the webpage but no logic can be defined in them. To ease the development, we have used a library named Bulma, which is very similar to Bootstrap. Bulma is a free and open-source web framework for designing web applications. It contains design templates for components such as forms, buttons, navigation tabs or tables.

On the other hand, we also need to make the web page interactive and interact with the Document Object Model (DOM). For that we use Javascript which is a language that allows to program the behavior of web pages. Furthermore, we will use the library jQuery to have a simpler and faster development. The philosophy of jQuery could be summarized as "write less, do more" and includes AJAX calls which are needed to communicate with the server. Finally, Dropzone library is used to provide drag and drop file uploads.

7.3.2 Datasets

This is the default tab of the web application. In Figure 7.3 we can see a preview this tab. The user can visualize all the datasets that have been uploaded to the system with some basic information about them. On the bottom there is a zone where the user can either drag and drop a *CSV* file or explore his filesystem and select it.

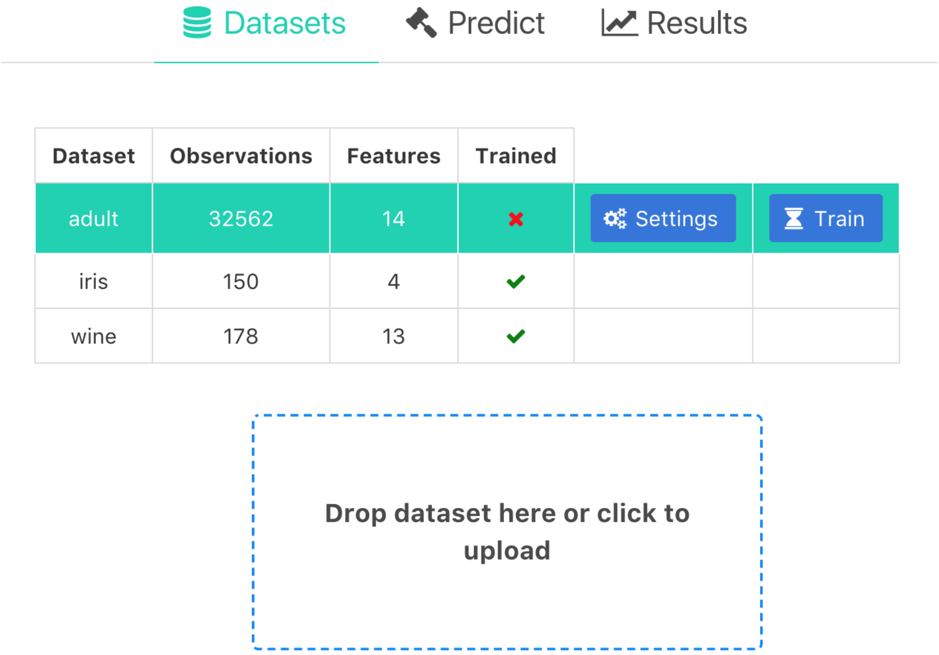
When the user clicks the Settings button, a modal window is opened and the content that appears is the one that can be seen in Figure 7.4. The user can examine and modify the settings associated with the dataset.

7.3.3 Predict

This is the tab where a user can interact with the model. The user must input manually the value of each feature. Once the user has filled the form, he can press the "Predict" button and the predicted class of that sample will be displayed in the class field.

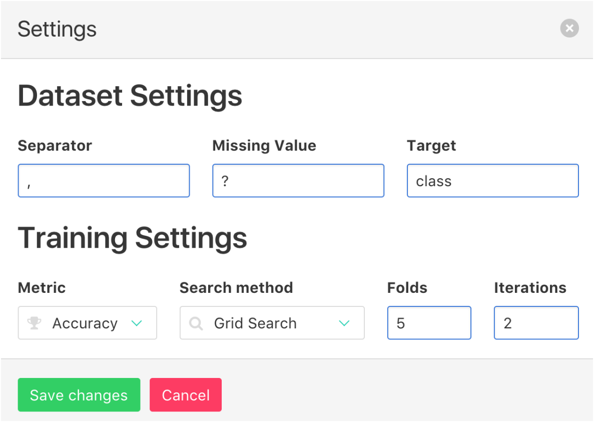
7.3.4 Results

This is the third tab of the web page. The user can visualize the results of the trained datasets. Results appear in the form of a table and the information that is displayed



Dataset	Observations	Features	Trained		
adult	32562	14	✗	Settings	Train
iris	150	4	✓		
wine	178	13	✓		

Figure 7.3: Datasets tab: general information about datasets



Dataset Settings

Separator: ,
Missing Value: ?
Target: class

Training Settings

Metric: Accuracy
Search method: Grid Search
Folds: 5
Iterations: 2

Figure 7.4: Settings window: configure the settings of your dataset

for each dataset is the following: train, validation, test scores and which algorithm has been finally selected.

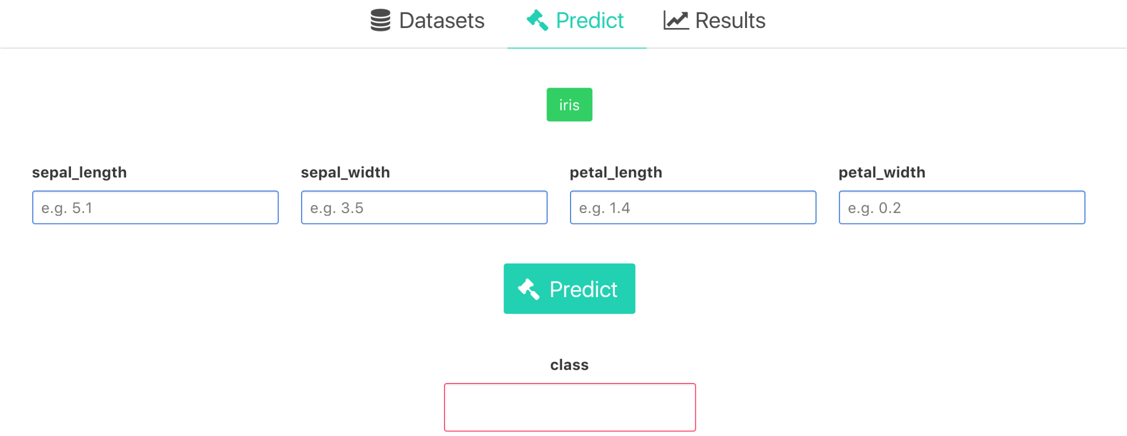
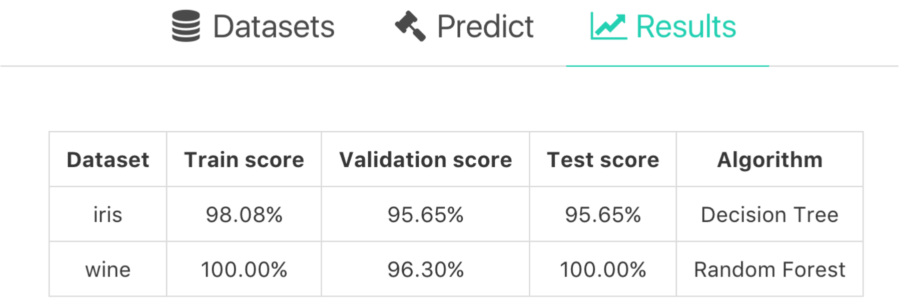


Figure 7.5: Predict tab: predict class of an observation



Dataset	Train score	Validation score	Test score	Algorithm
iris	98.08%	95.65%	95.65%	Decision Tree
wine	100.00%	96.30%	100.00%	Random Forest

Figure 7.6: Results tab: visualize training results of datasets

7.4 Interaction

In Figure 7.7 we can see how those three components interact. We will start with the user that uploads a dataset via the client. Once the dataset has been uploaded to the server, it calls a bash script that calls the classifier which performs the training of the dataset. When the classifier ends, the model and other relevant data such as the scores are persisted in a *.pkl* file. This format is convenient to communicate between Python programs. The server then reads the file and is able to load the corresponding model and the results associated with it. Then the client can access this information and display it to the user. Finally, the user can also impute new examples which are predicted using the model. Note that the classifier is only used to determine the model and once it has been computed, the client and the server do the rest of the job by themselves.

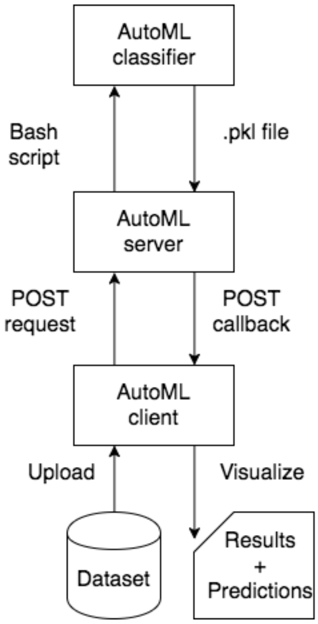


Figure 7.7: AutoML overview: interaction between the three modules

The motivation behind this partition is to favour modularity and reusability. Each component uses its own technology stack and has a different purpose. Moreover, it eases the development and testing of those components as we can bypass some layers. For example, we can test the server via Postman or we can test the classifier without having to interact with the client. Note that the final user does not acquire all the information that happened behind the scenes while the developer might want to access to more detailed information. Those different use cases also motivate the need of having different components. Finally, reusability is also favored by this design. The classifier could be used in another project with a different front-end. None of those benefits would apply if the project had been build in a monolithic component.

8 Clinical data analysis

Hospital de Sant Pau has provided a dataset in *CSV* format. This dataset corresponds to the records of the Sant Pau emergency service from 2008 to 2014.

Every row is an admission. Note that there may be more than one admission per patient, but many patients have only one admission. In Table 8.1 there is an explanation of the meaning of each feature. We want to predict if, once a patient has been discharged, he will be readmitted in the following 30 days.

8.1 Preprocessing

The raw dataset consists of 21042 observations and 56 features. Before starting experimenting with any dataset it must be meticulously analyzed in order to understand where the data comes from, which are the features and what kind of prediction we want to achieve.

First of all we must read the data in order to load it into memory and process it. There are two main languages that are used to perform this task in the data science community: R and Python. We will use the latter since it is easier to integrate this step into the AutoML classifier component, which is in Python. Moreover, Python provides a library named Pandas which is very similar in terms of functionality to R. Indeed, we can build a dataframe from the dataset and manipulate it in a convenient way.

We will begin with identifying each column. There are features that are mostly empty so we can remove them from the dataset. In fact, having a feature that has an unknown value for most of the data does not provide a lot of information. Thus, we will count the unique values of each feature. We performed this operation over each feature and we discovered that the following features could be discarded: *T_reg*, *D_naix*, *Sexe*, *Muni*, *Dist*, *Pais*, *UP_desti*, *Programa*, *CE1-CE5*, *PP*, *PS1-PS9*, *PX1-PX2*, *H_ingres*, *H_alta*, *Finan*, *Gestation time*, *Weight and sex of first/second baby*, *Identification number of each record*, *Tcis*, *Cis*, *Tpi* and *Cdi*. In total, the are 35 features that we initially discard due to excessive missing data.

Another problem might be a column that acts as an identifier. This is easy to detect as each value appears exactly once. Usually an identifier does not provide any kind of information, so it can also be dropped. We see that the feature *Assistance number* acts as an identifier, so we discard it.

Feature	Description
T_reg	Type of register
CIP	Personal identification code
D_naix	Birth date
Sexe	Sex
Muni	Municiplality
Dist	District
Pais	Country
Historia	Clinical history number
T_act	Activity type
Finan	Economic regime
D_ingres	Time of admission
C_ingres	Circumstance of admission
Pr_ingres	Origin
D_alta	Time of discharge
C_alta	Circumstance of discharge
UP_desti	Destination provider unit
Programa	Specific program
DP	Main diagnostic at admission time
DS1-DS9	Other diagnostics at admission time
CE1-CE5	External cause
PP	Main procedure
PS1-PS7	Other procedures
PX1-PX2	External procedures
H_ingres	Admission hour
H_alta	Discharge hour
Ser_alta	Discharge assistance service
T_gestacio	Gestation time
Pes_1r-Sexe_1r	Weight and sex of first baby
Pes_2n-Sexe_2n	Weight and sex of second baby
Num_assis	Assistance number
Numero_id	Identification number of each record
Tcis	Health document type
Cis	Health document code
Tpi	Administrative document type
Cdi	Administrative document code

Table 8.1: Description of the features of Sant Pau dataset before preprocessing

The following step is to construct new features. *D_ingres* and *D_alta* are two dates that indicate the time of admission and discharge respectively. Raw dates are useless for classifiers but useful features can be extracted from them. In this case, it seems reasonable to compute the total stay of the patient in days. To obtain that value, we simply have to subtract *D_ingres* from *D_alta*. By performing this operation on every sample, we add a new feature named *Num_Dias*. Another feature that we can extract is the number of previous admissions that a patient has. In fact, when a patient is admitted we know how many times he has been admitted in the past. This might be relevant since a patient that has been admitted multiple times might be more likely to be readmitted in 30 days than one that has never been readmitted. To construct that new feature, named *Prev_Ingr* we must group the dataset by patient. We can do that using the feature that identifies a patient which is *Historia*. Once the admissions are grouped, we can simply count how many previous admissions each patient has.

The most important feature of the dataset is the target feature, that is the feature that we want to predict. As we said, we want to predict whether or not a patient will be readmitted in the following 30 days after being discharged. Hence, a feature must encode this information. To generate this feature we group the admissions by patient and then we chronologically sort those admissions. That way, for each patient, we have a sorted list of all his admissions. Then we can iterate through that list and measure the time elapsed between consecutive admissions. This can be done using the release date of the first and the admission date of the second. If the time elapsed is less than 30 days, we mark the class of the first admission as 1, otherwise it is 0. Note that a direct consequence of this process is that all the admissions from a patient who has only one admission in total will belong to class 0.

To conclude the preprocessing we will take a look to the final dataset. It has 18677 admissions from class 0 and 2365 from class 1. In Table 8.2 we list the features that have finally been taken into consideration for the later analysis.

8.2 Descriptive analysis

Before starting to use a machine learning algorithm to predict new observations, it is useful to deeply explore the data that we have. We might discover some insights by analysing it.

Feature	Description
C_ingres	Circumstance of admission
Pr_ingres	Origin
C_alta	Circumstance of discharge
DP	Main diagnostic at admission time
DS1-DS9	Other diagnostics at admission time
Ser_alta	Discharge assistance service
Prev_Ingr	Previous number of admissions
Num_Dias	Days of hospital stay
target	Whether or not the patient is readmitted in the following 30 days

Table 8.2: Description of the features of Sant Pau dataset after preprocessing

We will only take into account the features that we have kept after preprocessing since the other ones will be discarded.

Clinical history number

This feature identifies each patient. We can see that, out of the 21042 admissions of the dataset, there are 12028 unique patients. That means that some patients have been admitted multiple times so indeed this data will be useful for a classifier that tries to predict whether or not a patient will be readmitted. From this feature we can easily obtain information about the number of admissions of each patient by grouping the observations by clinical history number. In Table 8.3 we can see the summary statistics. The mean of admissions per patient is 1.75 and the standard deviation is 1.56 which both make sense. We notice that the maximum value is 28, which means that there is a patient that has been admitted 28 times. At first it is surprising but it is plausible since it is an edge case and the data comes from an interval of seven years.

Date of admission

This feature indicates the date of the admission. We can see that the first date is 01/01/2008 while the last is 31/12/2014 so we can confirm that the data is included

count	mean	std	min	25%	50&	75%	max
12028	1.75	1.56	1	1	1	2	26

Table 8.3: Summary statistics of admissions per patient

count	mean	std	min	25%	50&	75%	max
2554	8.2	3.61	1	6	8	10	23

Table 8.4: Summary statistics of admissions per day

between years 2008 and 2014. We also remark that there are 2554 different days were they were admissions. If we compute how many days exist between the beginning of 2008 and the end of 2014, we obtain 2557 days, which means that there are 3 days without any admission. We find that those days correspond to 13/09/2013, 28/08/2010 and 17/06/2012. If we group observations by date we can obtain information about the number of admissions that occur in a given day. In Table 8.4 we see the summary statistics of admissions per date. The mean is around 8, which means that there have been 8 admissions per day on average. Finally, the day that had more admissions was 10/01/2014 with 23.

Date of discharge

This feature indicates the date of the release. We can see that the first date is 02/01/2008 while the last is 31/12/2014 so again it is compliant with the dataset time range. In this case there are 2529 unique dates which means that there are 28 days without any release. If we group observations by date we get the number of discharges per day. In Table 8.5 the summary statistics are displayed. The mean is almost similar to the one obtained in date of admission, which makes sense since every observation has both an admission and release date. It is slightly higher than the other one because there are less unique days. It is interesting to see that the standard deviation is one day higher than in admissions and that the day of more discharges was 21/12/2012 with 27.

To conclude the study of dates of admission and discharges, we study Figure 8.1 which shows a plot representing the number of admissions and discharges over

count	mean	std	min	25%	50&	75%	max
2529	8.3	4.4	1	5	8	11	27

Table 8.5: Summary statistics of discharges per day

mean	std	min	25%	50&	75%	max
8.29	10.7	0	2	5	10	172

Table 8.6: Summary statistics of hospital stay

time, grouped by months (that means that the values on the y axis are aggregated by months, so for example 300 corresponds to 300 admissions during a month). We note that both measures present the same shape, they almost overlap. In fact, discharges is similar to admissions but shifted a little bit to the right. The difference is the number of days that lasted the stay in the hospital. We also detect peaks located at the end of each year. This is surprising as we would expect the observations to be uniformly randomly distributed among time but it is clearly not the case. We can wonder whether or not this data is biased or if we received a sample of the original dataset.

Hospital stay

This feature is a derived one that represents the length of the stay in the hospital in days. In Table 8.6 are displayed the summary statistics. We see that the longest stay has been of 172 days while the shortest is 0 which means that the patient was released the same day as he was admitted.

Previous admissions

This is another derived feature which represents the number of previous admissions that a patient had when being admitted. In Table 8.7 we find the result of the summary statistics. Patients have a mean of 2.14 previous admissions and the patient who had 28 admissions is the one that has 27 previous admissions.

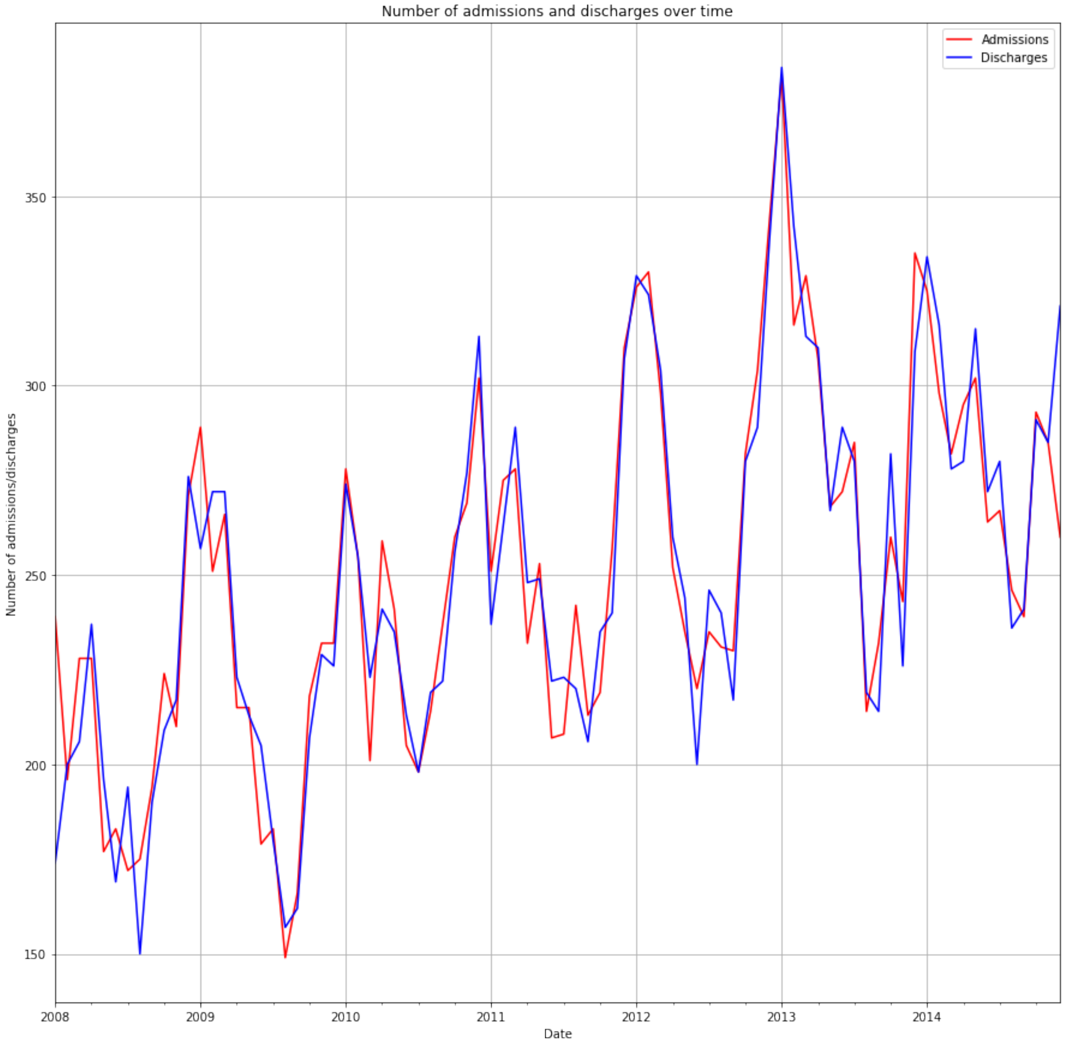


Figure 8.1: Number of admissions and discharges over time

mean	std	min	25%	50&	75%	max
2.14	3.12	0	0	1	3	27

Table 8.7: Summary statistics of previous admissions

Remaining features

On the one hand the remaining features to analyze are all the related with diagnostics at admission time. Since there are ICD-9-CM codes it is not easy to interpret the results. In Table B.1 we list the fifteen most common codes and explain to which disease they correspond. On the other hand there are features that encode the circumstances of admission and release, which are also codes which are not interpretable. For all those features we also checked that they had reasonable values but we will not display the results here.

9 Results

In this chapter we will display the results of the automatic predictive build that we designed. It is imperative to validate that our software works as intended and that it produces the desired results. We will separate this chapter in two different sections: AutoML performance and clinical data results. In the former we will compare the results that our program obtains with external ones and with the same datasets. That way, we can quantify how well it works. In the latter section we will use our program on clinical data.

For all those experiments, we run five times the program with different seeds and the results that we show are the mean of those five executions. This is important especially on smaller datasets where the variance can be high depending on how the data was initially split. Moreover we will perform a 10-fold cross-validation to prevent overfitting.

9.1 AutoML performance

We have chosen three different datasets to test our program. To select them we have taken into account their characteristics making sure that they are diverse between them. We also have searched for datasets which had already been analyzed in order to have reliable performance benchmarks. Those three datasets have been gathered from UCI (University of California Irvine) machine learning repository.

9.1.1 Iris dataset

Iris dataset [14] is probably the most popular dataset that exists. This dataset contains 3 classes of 50 instances each and each class corresponds to a type of iris plant. Each instance has 4 numerical features and there are no missing values. It is suitable to perform tests as the training of any model is fast with so few instances. Moreover it is a very simple dataset so if the performance is not high it probably means that there is some kind of error.

We can compare our results with the ones included in *Classification Of Complex UCI Datasets Using Machine Learning And Evolutionary Algorithms* [28]. In this paper the author presents the results he obtained using machine learning algorithms on a series of datasets and iris is included among them. Table 9.1 presents the results

Search method	Best algorithm	Accuracy	Data scientist Accuracy
Grid search	Naive Bayes	0.954	0.961
Random search	Random Forest	0.967	0.954

Table 9.1: Iris dataset results

Search method	Best algorithm	AUC	Data scientist AUC
Grid search	Decision Tree	0.756	0.787
Random search	Logistic Regression	0.765	0.775

Table 9.2: Adult dataset results

and we can see that they are close to the ones found by the real data scientist. It makes sense since, as we said, this is an easy dataset.

9.1.2 Adult dataset

Adult dataset [12] is one of the most popular datasets of UCI. This dataset is from the census bureau and the task is to predict whether a given adult makes more than $50.000 a year based on features such as education, age or workclass. It has 32562 instances and 14 features, both categorical and numerical and there are missing values.

In Table 9.2 we present the results of the adult dataset. In [40] we can see a reference for each algorithm. We remark that results from our program are similar to the ones obtained by a real data scientist. Nonetheless, we note that the data scientist has encountered better scores using more complex algorithms such as XGBOOST.

9.1.3 Congressional Voting dataset

Congressional voting dataset [13] is from the second session of the 98th Congressional Quarterly of Almanac. It includes the votes of each of the house of representatives congressmen on 16 votes. That means that there are 16 features, one for the outcome of each vote. The outcome can be for, against or unknown disposition which is imputed as a missing value. The task is to predict whether the representative was a democrat or republican one. It has 435 instances so it is a small dataset.

Search method	Best algorithm	Accuracy	Data scientist Accuracy
Grid search	Logistic Regression	0.939	0.960
Random search	Decision Tree	0.954	0.956

Table 9.3: Congressional Voting dataset results

We can compare our results with the ones included in *Classification Of Complex UCI Datasets Using Machine Learning And Evolutionary Algorithms* [28], which is the same paper that was used to compare the iris dataset. In Table 9.3 we display the results. We observe that similar results are obtained with the same algorithms.

9.2 Clinical data results

9.2.1 Sant Pau dataset

This is the dataset that we analyzed in Chapter 8 and building a model for that dataset is part of the main objectives of this project. Note that this dataset was previously studied by [34] so we know the results that a real data scientist has obtained beforehand. That way, we will be able to compare those results.

Table 9.4 displays the results obtained with the dataset. We see that both methods find similar results and that they are slightly better than the ones found by the real data scientist. Note that in the original paper [34], it was announced that an AUC of 0.887 was obtained, but the director of this project personally communicated that there was a mistake on the evaluation and that the real value was 0.688, which was obtained using random forests of sizes 50-100. The results of our program are close from the best ones found by the data scientist. Our system did well exploring naive bayes algorithm but failed to explore successfully a more complex algorithm such as random forest, that seems more promising according to the job of the real data scientist. With this dataset we break into some of the limitations that this system appears to be. In fact, this dataset is probably more complex than the others.

Confusion matrices of grid and random search are presented in Tables 9.5 and 9.6 respectively. We can see that the results again are similar in both. We observe that the classifier fails to identify properly readmissions. Indeed, we see that the readmitted class is underrepresented with respect to the not readmitted one. All in

Search method	Best algorithm	AUC	Data scientist AUC (with same algorithm)	Data scientist AUC (best score found)
Grid search	Naive Bayes	0.677	0.540	0.688
Random search	Naive Bayes	0.647	0.540	0.688

Table 9.4: Sant Pau dataset results

		Predicted Class	
		Not readmitted	Readmitted
Actual Class	Not readmitted	1906	896
	Readmitted	127	228

Table 9.5: Sant Pau confusion matrix on test set with grid search classifier

		Predicted Class	
		Not readmitted	Readmitted
Actual Class	Not readmitted	1851	951
	Readmitted	130	225

Table 9.6: Sant Pau confusion matrix on test set with random search classifier

all, this experiment confirms that our software's performance is comparable to the data scientist one but at the same time we recognize that it could even be improved by exploring better more complex algorithms such as random forests.

9.2.2 Ictus dataset

The last dataset that we will examine is the ictus dataset. This dataset was also provided by the director of this project and is made of 364 instances with 75 features each. Each instance is a patient who had an ictus and the goal is to predict what the outcome of this patient will be. There are four possible outcomes: home, socio-health, exitus and hospital acute. In Figure 9.7 we show the results that we obtained. In this case, random search has achieved a higher performance than grid search. The study that was performed on this dataset was focused on clustering, so we have no references. Nevertheless, the director of this project suggested that he reached a similar performance with decision trees than the ones that we got here with random forests, so we can indeed validate our results.

Search method	Best algorithm	Accuracy
Grid search	Logistic Regression	0.618
Random search	Random forest	0.709

Table 9.7: Ictus dataset results

Tables 9.8 and 9.9 show the confusion matrices of the best grid search and random search classifiers respectively (which are logistic regression and random forest). We see that both results are similar. Random forest better identifies exitus patients and they both identifying patients who ended entering the hospital. This coincides with the results that the director of this project obtained analysing this dataset.

		Predicted Class			
		Home	Socio-health	Exitus	Hospital
Actual Class	Home	24	3	1	0
	Socio-health	5	7	3	0
	Exitus	1	6	3	0
	Hospital	2	0	0	0

Table 9.8: Ictus confusion matrix on test set with grid search classifier

		Predicted Class			
		Home	Socio-health	Exitus	Hospital
Actual Class	Home	27	1	0	0
	Socio-health	8	6	1	0
	Exitus	4	2	6	0
	Hospital	1	1	0	0

Table 9.9: Ictus confusion matrix on test set with random search classifier

10 Conclusions

10.1 Technical conclusions

In order to rigorously evaluate this project, we can ask ourselves whether or not we have met the objectives that were defined at the beginning.

We have preprocessed and analyzed a clinical dataset successfully and we have carried out an exploratory analysis that has provided some curious insights about the dataset. After the preprocessing we obtained a similar class distribution than the one that was achieved in [34], the project that was taken as a reference and as starting point of this one. Hence, we can validate that the analysis was valid. Nonetheless, we must mention the fact that the quality of this dataset was a little bit disappointing as a lot of features had missing values and had to be discarded. For example, it would have been interesting to have available the birth date of patients in order to derive its age.

We have implemented an automated model builder for classification problems that is able to construct a model that can be used to predict unseen data. The process if fully automated as the user only has to provide a dataset and a basic configuration file and the system takes care of the rest of the process. In order to accomplish that objective, we have designed a software that executes a series of steps and creates the best model that it finds.

The exploration of hyperparameters has been implemented with two distinct search methods: grid search and random search. If we had more time, we would have tried to compare the performance of those techniques. However, we have tested our program and the results were comparable to the ones that a real data scientist. It should be said that more complex algorithms usually make better predictions but they also tend to overfit more easily. As a consequence, its hyperparameters are also more difficult to tune. All in all, we can conclude that we managed to search successfully between the hyperparameter space of the four classification algorithms that we implemented.

Finally, we have designed a web application that allows a user without experience in machine learning interact with it. Indeed, we have managed to present machine learning, which is an intrinsically complex field, under the appearance of an easy-to-use tool that makes predictions. The web allows the user to upload the dataset, configure its settings and compute a model. Then he can manually impute a new instance and see what class does the model predict. He can also visualize the performance of the model on training, validation and testing data.

10.2 Personal conclusions

Before starting this project my knowledge about machine learning was scarce as I did not have the opportunity to dive into this topic at university. Due to this project I have learned a lot of concepts and hands-on experience on the topic. I have learned how to train and validate predictive models to perform classification tasks. I have also realized how important it is to plan and invest time in the management of project. Moreover, I have succeeded at designing a web application to interact with the automated predictive builder that I implemented. If I had been told that at the end of the project I would have been able to design such an interface, I would not have believed it. Indeed, I had never developed a web application and at the beginning it was an arduous task. All in all, this is the first time that I have experience acting as a full-stack developer. I think it is a great experience since knowing all the technology stack allows you to develop any kind of project in the future.

10.3 Future work

Due to the breadth of the project, several extensions can be implemented to improve or enhance this project:

- **Adapt the system to the size of the dataset**
  The current implementation of AutoML classifier does not take into account the size of the dataset. It could be interesting to make some decisions based on the number of samples and/or the number of features. In fact, there are hyperparameters that are more suited for larger datasets and others that work better in smaller. Moreover, the computation time can be relatively high in large datasets, so it would be interesting to prune the search.

- **Implement more classifiers in order to increase the performance of the final model**
  The more classifiers there are, the higher is the chance to find the one that better fits the problem. In the software we build, it is very easy to add a

classifier to the system. Furthermore, adding a classifier will never damage the performance of the program, it will only penalize the computation time of the overall system.

- **Study in detail the influence of hyperparameters in algorithms**
  Due to the time limitation of the project, experiments about the influence of hyperparameters in algorithms have not been able to carry out. Now that we have implemented all the infrastructure that is needed, it would be interesting to design experiments that test how each hyperparameter impacts the performance of a given algorithm.

- **Add preprocessing techniques to improve the performance of the classifier**
  Preprocessing of data is a key step in any machine learning problem. A meticulous data cleaning might have more impact on the final performance of a classification algorithm than any kind of hyperparameter setting.

- **Improve the user experience of the web application**
  As it is, there is no help nor a tutorial for a new user that interacts with the web application. Moreover, if something goes wrong in the server side the user does not receive feedback about what went wrong and how he can fix the error. That can lead to an unsatisfactory experience.

- **Compare the performance of this software against other automated machine learning projects**
  In the state of the art section we have cited some platforms that perform a similar task than the one defined in this project. We could design an experiment with a series of representative datasets and compare the performances of both systems. It is not trivial to design a fair experiment, as we probably should take into account the computation time and the computation power used by them. In fact, we should somehow ensure that they are given the same conditions.

Bibliography

[1] Agencia estatal boletín oficial del estado. https://www.boe.es/diario_boe/txt.php?id=BOE-A-1978-31229. Online. Accessed: 2017-05-29.

[2] Businessdictionary, online business concepts dictionary. http://www.businessdictionary.com/definition/action-plan.html. Online. Accessed: 2017-03-05.

[3] Teamgantt, online gantt chart software. https://www.teamgantt.com/. Online. Accessed: 2017-03-05.

[4] Datarobot. https://www.datarobot.com/, 2016. Online. Accessed: 2017-02-25.

[5] Hyperopt: Distributed asynchronous hyperparameter optimization in python. http://jaberg.github.io/hyperopt/, 2016. Online. Accessed: 2017-06-06.

[6] Skytree platform. http://www.skytree.net/, 2016. Online. Accessed: 2017-02-25.

[7] Tuning the hyper-parameters of an estimator. http://scikit-learn.org/stable/modules/grid_search.html#, 2016. Online. Accessed: 2017-03-23.

[8] Apis of scikit-learn objects. http://scikit-learn.org/stable/developers/contributing.html#apis-of-scikit-learn-objects, 2017. Online. Accessed: 2017-09-11.

[9] Auto sklearn, 2017. Online. Accessed: 2017-10-11.

[10] Overfitting — Wikipedia, the free encyclopedia, 2017. Online. Accessed: 2017-10-01.

[11] Sigmoid function — Wikimedia commons, 2017. Online. Accessed: 2017-10-01.

[12] Uci: Adult dataset, 2017. Online. Accessed: 2017-10-15.

[13] Uci: Congressional voting dataset, 2017. Online. Accessed: 2017-10-15.

[14] Uci: Iris dataset, 2017. Online. Accessed: 2017-10-15.

[15] Weka 3: Data mining software in java, 2017. Online. Accessed: 2017-10-11.

[16] APPLE. Macbook pro technical specifications. `https://support.apple.com/kb/sp649?locale=en_US`, 2014. Online. Accessed: 2017-03-09.

[17] ARIAN HOSSEINZADEH, MASOUMEH IZADI, AMAN VERMA, DOINA PRECUP, AND DAVID BUCKERIDGE. *Assessing the Predictability of Hospital Readmission Using Machine Learning* (2013). Proceedings of the Twenty-Fifth Innovative Applications of Artificial Intelligence Conference.

[18] ARLINE, K. Business news daily: *Direct Costs vs. Indirect Costs: Understanding Each*. `http://www.businessnewsdaily.com/5498-direct-costs-indirect-costs.html`, 2015. Online. Accessed: 2017-03-07.

[19] BERGSTRA, J., AND BENGIO, Y. Random search for hyper-parameter optimization. *Journal of Machine Learning Research 13* (Feb. 2012), 281–305.

[20] BREIMAN, L. Random forests. *Mach. Learn. 45*, 1 (Oct. 2001), 5–32.

[21] DE CATALUNYA, G. The electrical mix. `http://canviclimatic.gencat.cat/en/redueix_emissions/factors_demissio_associats_a_lenergia/index.html`, 2016. Online. Accessed: 2017-03-09.

[22] DOMINGOS, P. A few useful things to know about machine learning. *Commun. ACM 55*, 10 (Oct. 2012), 78–87.

[23] EVELYN ROVIRA, SALVADOR BENITO, RICARD GAVALDÀ, MIREIA PUIG, JULIANNA RIBERA. *Evaluating Preventive Measures for Heart Failure Readmissions using Machine Learning* (2016). Manuscript.

[24] FAWCETT, T. An introduction to roc analysis. *Pattern Recogn. Lett. 27*, 8 (June 2006), 861–874.

[25] FEURER, M., KLEIN, A., EGGENSPERGER, K., SPRINGENBERG, J., BLUM, M., AND HUTTER, F. Efficient and robust automated machine learning. In *Advances in Neural Information Processing Systems 28*, C. Cortes, N. D. Lawrence, D. D. Lee, M. Sugiyama, and R. Garnett, Eds. Curran Associates, Inc., 2015, pp. 2962–2970.

[26] FORTMANN-ROE, S. Understanding the bias-variance tradeoff. `http://scott.fortmann-roe.com/docs/BiasVariance.html`, 2012. Online. Accessed: 2017-08-07.

[27] GUO, G., WANG, H., BELL, D., AND BI, Y. Knn model-based approach in classification.

[28] GUPTA, A. Classification of complex uci datasets using machine learning and evolutionary algorithms, 2015. Online. Accessed: 2017-10-11.

[29] JAMES, G., WITTEN, D., HASTIE, T., AND TIBSHIRANI, R. *An Introduction to Statistical Learning: With Applications in R*. Springer Publishing Company, Incorporated, 2014.

[30] JASDEEP SINGH MALIK, PRACHI GOYAL, MR.AKHILESH K SHARMA. *A Comprehensive Approach Towards Data Preprocessing Techniques & Association Rules* (2014).

[31] LUO, G. Predict-ml: a tool for automating machine learning model building with big clinical data. `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC4897944/`, 2016. Online. Accessed: 2017-02-25.

[32] MURPHY, K. P. *Machine Learning: A Probabilistic Perspective*. The MIT Press, 2012.

[33] PARIKH, R. Garbage in, garbage out: How anomalies can wreck your data. `https://blog.heapanalytics.com/garbage-in-garbage-out-how-anomalies-can-wreck-your-data/`, 2014. Online. Accessed: 2017-02-25.

[34] ROVIRA, E. *Predicción de reingresos de pacientes hospitalarios* (2016). Trabajo de fin de grado, FIB, UPC.

[35] RUSSELL, S. J., AND NORVIG, P. *Artificial Intelligence: A Modern Approach*, 2 ed. Pearson Education, 2003.

[36] SKERRETT, I. Eclipse community survey 2014 results. `https://ianskerrett.wordpress.com/`, 2014. Online. Accessed: 2017-02-25.

[37] SNOEK, J., LAROCHELLE, H., AND ADAMS, R. P. Practical bayesian optimization of machine learning algorithms. In *Advances in Neural Information Processing Systems 25*, F. Pereira, C. J. C. Burges, L. Bottou, and K. Q. Weinberger, Eds. Curran Associates, Inc., 2012, pp. 2951–2959.

[38] TAPE, T. G. Interpreting diagnostic tests. `http://gim.unmc.edu/dxtests/roc3.htm`, 2017. Online. Accessed: 2017-06-07.

[39] THORNTON, C., HUTTER, F., HOOS, H. H., AND LEYTON-BROWN, K. Auto-weka: Combined selection and hyperparameter optimization of classification algorithms. In *Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining* (New York, NY, USA, 2013), KDD '13, ACM, pp. 847–855.

[40] TOPIWALLA, M. Machine learning on uci adult data set using various classifier algorithms and scaling up the accuracy using extreme gradient boosting, 2017. Online. Accessed: 2017-10-01.

[41] WANG, F., AND SUN, J. Survey on distance metric learning and dimensionality reduction in data mining. *Data Min. Knowl. Discov. 29*, 2 (Mar. 2015), 534–564.

A Technology

- **Python**: https://www.python.org/
- **Flask**: http://flask.pocoo.org/
- **Conda**: https://conda.io/
- **Bulma**: http://bulma.io/
- **Dropzone**: http://www.dropzonejs.com/
- **Postman**: https://www.getpostman.com/
- **PyCharm**: https://www.jetbrains.com/pycharm/
- **GitHub**: https://github.com/
- **Sklearn**: http://scikit-learn.org/stable/
- **JavaScript**: https://www.javascript.com/
- **jQuery**: https://jquery.com/
- **Draw.io**: https://www.draw.io/

B List of ICD-9-CM codes

The International Classification of Diseases (ICD) is maintained by the World Health Organization and is used as the standard diagnostic tool for epidemology.

Disease	ICD-9-CM code	Occurrences
Congestive heart failure	428.9	6371
Acute bronchitis	466.0	928
Left heart failure	428.1	868
Coronary atherosclerosis	414.01	778
Atrial fibrillation	427.31	624
Pneumonia	486	559
Respiratory failure, acute	518.81	497
Chronic bronchitis	491.22	495
Other primary cardiomyopathies	425.4	380
Respiratory failure, acute and chronic	518.84	351
Myocardial infarction, acute, subendocardial	410.71	341
Urinary tract infection	599.0	341
Pneumonitis due to solids and liquids	507.0	274
Valvular disorder, aortic	424.1	125
Myocardial infarction, acute, anterior	410.11	200

Table B.1: ICD-9-CM codes of the fifteen most common diseases in the Sant Pau dataset