UNIVERSITY OF TRENTO

Master’s Degree

in

Data Science

MACHINE LEARNING OPTIMISATION OF SOFTWARE DEVELOPMENT PROCESSES: A REAL-CASE APPLICATION OVER TICKET ISSUING DATA

Supervisor

IVANO BISON

Candidate

LEONARDO PAJER

Academic Year

2021/2022

Sommario

INTRODUCTION 3
CHAPTER 1: Information flow management and product development: a machine learning approach 6
1.1 Information flow and its challenges 6
1.2 Introduction to the real-case scenario project and current usage of machine learning methods in software development. 9
1.3 Real-case classification project structure and literature overview of machine learning application in ticket issuing processes 15
CHAPTER 2: Artificial Intelligence for information flow and software development optimisation. 18
2.1. Project's framework and AGILE method 20
2.2 Strategic planning and data collection 25
2.3 Feature selection, prediction objectives assessment and Results analysis............ 26
CHAPTER 3: Case Study. 29
3.1 Objective............ 29
3.2 Data Description 30
3.2.1 Data Cleaning............ 33
3.2.2 Data transformation 33
3.3 Methodology 34
3.3.1 Feature engineering 34
3.3.2 Metrics and algorithm 36
3.4 Exploratory Data Analysis............ 39
3.5. Prediction results and knowledge extraction............ 45
CHAPTER 4: Conclusions. 51
4.1 Summary of classification results 51
4.2 Limitations and further research............ 52
BIBLIOGRAPHY............ 54
APPENDIX............ 56

INTRODUCTION

In software production processes, maintenance is a crucial task that involves several activities. Source code must be kept up to date and possible defects in performance must be corrected. Moreover, maintenance tasks requirements should involve as little effort as possible in their comprehension (Panichella et al. 2014), to give more time to maintainers to resolve the tasks, hence improving efficiency and keeping software maintenance costs low. Additionally, one of the most crucial challenges in software development maintenance tasks is information flow (Eppinger, 2001), which can prove to be detrimental for a firm in terms of performances, credibility, and morale. In fact, while issue tracking systems are an important tool for maintenance staff to enable rigorous yet effective software development, many developers still end up with a rapidly growing workload and lose control of it (Bissandyè et al, 1023; Panichella et al., 2014 in Kallis et al, 2019).

To address these challenges, the major goal of this paper is to use Artificial Intelligence (AI) to examine the flow of tickets issued during the production processes of a Software Development Department (SWD). With this in mind, we aim to contribute to the exploration of the potential of AI techniques in Software Development.

Specifically, to try to answer to the limitations inherent of SWD just depicted, we will first propose a framework for interdepartmental collaboration, in the development of AI techniques in the service of the software development department, with the hypothesis that improved information flow might lead to improved performance; in the case study, then, within the framework proposed, we will train several widely-validated classification Machine Learning algorithms, such as Logistic Regression, Random Forest and XGBoost with the objective of predicting the probability that a ticket issue, related to software development tasks, will be solved within the assigned deadline given at the time of its creation. This is important for developers who may need time to handle new issues (Izquierdo et al., 2015), as this approach classifies tickets at the time they are reported, saving developers time to identify ticket issues that require more attention from the start, avoiding the risk of escalation.

The innovation apported by this research, instead of being in the algorithms side, can be found in the way in which the classification model is integrated in the SWD production processes and how this could lead to an information flow improvement. Ultimately, the following are the major contributions of this essay:

- An artificial intelligence framework that fosters collaboration and thus information flow between AI experts and software development specialists, especially in the strategic planning, data collection, model selection, and target definition phases, in accordance with the objectives of Software Development.
- A case-study, implementing the proposed framework, adopting Artificial Intelligence, and in particular Machine Learning algorithms such as Random Forest, Logistic Regression and XGBoost, with the objective of predicting the probability that a ticket issue, related to software development tasks, will be solved within the assigned deadline given at the time of its creation.

To show our contributions, since ticket issuing control is a subset of information flow management, we first give a definition of these concepts, and we investigate the applications of Artificial Intelligence in connection with information flow and software development, in Chapter 1.1 and 1.2. Secondly, we examine the scientific literature in merit the uses of Machine Learning (ML), a branch of Artificial Intelligence, that specifically dealt with ticket issuing processes, in Chapter 1.3. After the main concept depicted in this essay are explained and contextualised in Chapter 1, Chapter 2 aims at describing the proposed framework. In fact, in Chapter 2.1 we present the framework and its main advantages, and in chapter 2.2 and 2.3 we explore the points of contact between Software Development and AI experts within the process of deployment of a machine learning algorithm. In Chapter 3, we present our case study, the application of Artificial Intelligence to the management of ticket issuance flow. In Section 3.1, we present the goals and rationale for the case study. Information about the dataset we used and the preprocessing tasks, such as data cleaning and data transformations, can be found in Section 3.2, while information about the selection of metrics and algorithms and the training procedure can be found in Section 3.3. The insights we obtained from our data thanks to survival analysis techniques such as COX regression, Kaplam-Meier estimation and Nelson-AAlen estimation are presented in Chapter 3.4. In particular, the results of this analysis, which were fundamental for the development of new features from the raw dataset available to us and finally used in the classification task through random forest, logistic regression and XGBoost, are presented in Chapter 3.5. Finally, Chapter 3.6 presents the results of the classification algorithms, which, after fine-tuning based on the chosen metrics, Recall, and cross-validation, show that Random Forest comes out on top with a recall of 0.923. In the concluding Chapter 4, the results of the case studies are discussed in relation to the application of the proposed framework. In particular, the impact that the classification model applied in the case study had on achieving the software development goals and improving the information flow is examined.

CHAPTER 1: Information flow management and product development: a machine learning approach

The historical moment in which we live in is called Information Age and this because information is the lifeblood of society. Information has several applications; according to Sandra Braman, professor of Communication, Law and Policy at the Texas A&M University, information can be used and analysed in 4 ways: as a resource, as a commodity, as perception of patterns, and as a constitutive force in society (Braman, 1989). Managing Information flows is one of the core activities of any business, which not always address the challenge with the appropriate number of resources. For these reasons, one of the main contributions of this essay relies on information flow management and, in particular, on the proposal of an artificial intelligence implementation framework that fosters collaboration and thus information flow between AI experts and software development specialists, especially in the strategic planning, data collection, model selection, and target definition phases, in accordance with the objectives of Software Development. To be able to understand the key concepts of the presented framework, in chapter 1 we will give their definition and we will explore the point of contacts between Artificial Intelligence and Information Flow Management in the application of optimisation tasks within the Software Development environment.

1.1 Information flow and its challenges

The focus of this paper relies, with the development of a classification algorithm which predicts ticket issues escalation, in information flow management which is considered by the MIT’s Professor of Management Science and Innovation Steven D. Eppinger as the lifeblood of processes such as product development (Eppinger, 2001). Information flow is defined as the exchange of information among people, processes, and systems within an organisation. Managing Information flows is one of the core activities of any business, which not always address the challenge with the appropriate number of resources. The difference between informed and uninformed decisions might result in dramatic outcomes in fields like negotiation, stock market or product development. Information can boost a business output by only having all employees aligned on the same goal, and the application of information flow management techniques affects all areas of an organisation. In a typical project development process, the various teams working on a project will meet at the beginning of the project, to have a clear view of the desired output and at the end for the final adjustments; some other meetings might happen in between with all the teams working for a specific project. While these leads to a considerable saving of time, and hence money, on the other side may cause miscommunication. In this way small problems can escalate in a failed delivery. Moreover, and frequently not addressed by businesses, when little is said between colleagues, teams will be working within themselves and will likely not let other groups to know the state of their work, creating a cascade reaction of missed interactions in which every team tries to solve the stack of problems by themselves, slowing down the productivity and production. All these challenges may cause the development process to be disjointed. Developer, designer, and marketers might also lose the scope of the project and start to focus on their personal part instead of the bigger picture.

Several can be the directions of the information flow. In most of traditional business, especially if we think about Italy, in which, according to Il Sole 24 Ore, small[1] and traditional businesses occupy 82% of the work force, the information flow is vertical, top to bottom. A vertical information flow assumes that the ideas, decisions, and proposals are dictated from the vertices of the company, leading to a one-way communication. Employees, as we were discussing before, might feel a lack of inclusion and therefore, the working environment might suffer from it. On the other hand, in case of horizontal information flow, ideally information is equally shared and available. In this scenario there are risks as well. For example, different teams working at the same project, even though sharing information might suffer of a lack of supervision in more technical aspects; for example, a Social Media Manager of a company might as well know all data that his/her colleagues have at their disposal for the necessary analysis, but he/she must rely on them for the consistency of their analysis. To fill the gap, company adopting horizontal management and information flow adopts a hybrid strategy, in which the macro-organisation is horizontal but internally each team have a vertical structure.

From this short introduction to the problem what emerges is also the scope of this research: the hierarchical organisation of a business generally resembles its information flow management and whether this is horizontal, vertical, or diagonal this cannot be proven to be the right or the wrong organisational choice. The aim of the research is to offer a practical perspective on the topic hence, to present a real-life application of information flow optimisation whose results are measured in terms of

[1] For small business we intend a business classified as Piccola Media Impresa which means that the total revenue must not overcome the threshold of 50 million of Euros and the business employs less then 250 workers. Source: Recommendation n. 2003/361/Ce of European Comission of 6 May 2003.

information flow benefits, data integration and risk management. The question that guided this paper was: is Artificial Intelligence, if implemented in business, able to produce sensitive results in terms of information flow and productivity management without the need of drastic organisational changes?

In the second paragraph of this research, we will explore more deeply the challenges that are inherent in the software development environment and how artificial intelligence, in particular machine learning, helped in tackling these challenges. The decision of exploring particularly the software development environment relies on the fact that in the second part of this thesis, will be presented the real-case project, conducted in Maggioli S.p.a., specifically under the Research and Development department, which focus was software development and information flow optimisation.

1.2 Introduction to the real-case scenario project and current usage of machine learning methods in software development.

The decision of exploring explicitly the software development environment relies on the fact that in the second part of this thesis, will be presented the real-case project, focal point of this thesis, which was conducted in two stages in Maggioli S.p.a., under the Research and Development department, which main goal was to develop a classification algorithm to predict, at the moment of reception, which issues related to the software development process would not respect the resolution time assigned. This reason also justifies the choice of exploring briefly what information flow is and how it affects a business. As you can see, in the project done in Maggioli the topics of software development and information flow where simultaneously addressed. As an example of how deeply these two managerial fields where related let us focus on the question that motivated the research; changing the core structure of a business in specific scenarios surely can prove to be beneficial but at the same time requires planification, vision, time and has costs at least in the short term. This makes the choice of a reorganisation of a business less likely, especially for traditional firms. In this way, there is less space for information flow enhancement and hence companies would suffer from it in terms of productivity, alignment, and morale. The crucial point of this research is to find a way to optimising information flow and in general the productivity of a company without requiring expensive and drastic reorganisation changes.

To achieve the result the first step is to review what where the challenges of software development, to find its weakness points, and hence to have an idea of which are the optimisation possibilities.

According to Zhang and Tsai (D Zhang, JJP Tsai - Software Quality Journal, 2003, p. 87) Software Quality Journal, 2003) the following essential difficulties inherent in developing large software still hold true today and they are:

- ***Complexity****: Software entities are more complex for their size than perhaps any other human construct. Many of the classical problems of developing software products derive from this essential complexity and its nonlinearity increases with size.*

- ***Conformity****: Software must conform to the many different human institutions and systems it comes to interface with.*

- ***Changeability**: The software product is embedded in a cultural matrix of applications, users, laws, and machine vehicles. These all change continually, and their changes inexorably force change upon the software product.*

- ***Invisibility**: The reality of software is not inherently embedded in space. As soon as we attempt to diagram software structure, we find it to constitute not one, but several, general directed graphs, superimposed one upon another. (Brooks.*

To address these problems, algorithms of machine learning[2] and artificial intelligence, developed for mainly scientific purposes, started to enter in the environment of software engineering.

In 1992 at the conference of the American Association for Artificial Intelligence, Selfridge saw the path of a collaboration between software engineering and Machine Learning. He said: “For systems of the future, we need to think in terms of shifting the burden of evolution from the programmers to the systems themselves. [..] There is the need to explore what it might mean to build systems that can take some responsibility for their own evolution” (Selfridge, 1993). His words are clearly an advanced thought, considering the times. In fact, he seeks for automation to increase productivity and keep up with customers’ requirements, the two main problems of this fast-paced digital era. From these words we take the first hint that the approach of artificial intelligence for business output optimisation and customer care has roots in the scientific community.

[2] Machine Learning is the study of computer algorithms that improve automatically through experience. Applications range from datamining programs that discover general rules in large data sets, to information filtering systems that automatically learn users' interests (Mitchell, 1997).

The rapid development of machine learning algorithms and tools, and easier access to available frameworks and infrastructures have greatly fuelled this era of the development of ML-based software solutions for real-world problems. Software companies, big or small, are striving to adopt ML in software applications, to improve their products and services. However, the promising potentials of machine learning accompany multifaceted challenges to the traditional software development processes and practices (Khomh, Antoniol, 2018 & Schelter et al., 2018).

A big challenge, in adopting machine learning techniques for software development purposes is the overload of work that the development phase in each cycle can go through. By nature, then, teams must be versatile, creative, and with strong and cross-sector knowledge. Even more, machine learning techniques requires an understanding of mathematics and statistics that software developer might not have, or might have not used for long, so a transition to the adoption of machine learning faces several setbacks.

A model that was designed to be slim and versatile is the agile method. This makes the agile method well-suiting the application of machine learning techniques to make several tasks completely automated, hence time saving and with reduced maintenance costs. All controls systems, even more, are required to be data-driven[3], to allow high rates of automation over control processes and risk assessment.

According to Kumeno, "the systematic development, deployment and operation of ML applications faces major difficulties. The methodologies and tools of software engineering have greatly contributed to a wide range of activities in the lifecycles of

[3] Data-driven control systems are a broad family of control systems, in which the identification of the process model and/or the design of the controller are based entirely on experimental data collected from the plant. Source: https://en.wikipedia.org/wiki/Data-driven_control_system

traditional information systems but are difficult to implement in ML application projects because ML applications and traditional software systems differ in fundamental ways" (Kumeno, 2019, p. 1). Even if Kumeno points out well the problem of the cooperation between machine learning development and software engineering, still the use of machine learning is of great employment alongside software engineering.

Zhang and Tsai in their research "Machine Learning and Software Engineering" tried to capture the several real case projects that involved both Software engineering and Machine Learning applications. They also categorised the different uses to give an analytical viewpoint of the development panorama:

1. **Prediction and estimation**. In this group, ML methods are used to predict or estimate: (1) software quality, (2) software size, (3) software development cost, (4) project or software effort, (5) maintenance task effort, (6) software resource, (7) correction cost, (8) software reliability, (9) software defect, (10) reusability, (11) software release timing, and (12) testability of program modules.
2. **Property and model discovery**. ML methods are used to identify or discover useful information about software entities.
3. **Transformation**. Here ML is used to guide developers in the development choices.
4. **Generation and synthesis**. Machine Learning techniques are used to augment data, to generate trial data and help with descriptive analysis.
5. **Reuse library construction and maintenance.**
6. **Requirement acquisition.** With the adoption of data mining tools, it i possible to retrieve requirements in unsupervised ways.

7. **Capture development knowledge.** Machine learning have been proven to support evolving knowledge and domain analysis methods that capture emerging knowledge and synthesize it into generally applicable forms (Zhang, D., Tsai, JJP., 2003).

As we can see the applications of machine learning based methods are broad and vary. Certainly, the diffusion of algorithms like Decision Trees and linear regression have led the path to automation, with the merit of offering the chance of control and gaining insightful information out of them. In fact, their success is found in the interpretability of the output, that, even if not significant in terms of accuracy still leads to helpful knowledge acquisitions in terms of requirements acquisition.

From the Zhang and Tsai examination of the literature that relates machine learning and software development we had confirmation that the scientific ground floor is sound and hance, that our real-life business application might generate the desired optimisation output.

In the first paragraph we tried to focus the terminology and problem definition of our thesis, while in this chapter we explored the scientific literature to understand what uses of machine learning, if any, were done in the software development environment. Now in the third paragraph of this introductory chapter, we will explore in detail the application related to information flow optimisation and ticket issuing of customer service requests that in the literature have seen machine learning deployed.

1.3 Real-case classification project structure and literature overview of machine learning application in ticket issuing processes

Thanks to the insightful work of Thang and Tsai we have a full picture of the uses that paired machine learning with software development in the scientific literature. To understand better our intention of the literature review that follows in this paragraph and why this is related to the real-life project we are going to present in the second part of this thesis, we will now show the steps, or structure, of the project.

The first step was, as it is for this paper, to examine the possibility of implementing artificial intelligence techniques within a business in its software development department. As we already explained, this is also the main purpose of this first chapter.

The second step, which was to gather data from several sources already set the possibility of applying machine learning to improve performances. In fact, as Thang and Tsai reported, two of the main fields of knowledge, that saw in scientific literature machine learning applied, were **generation and synthesis** and **property and model discovery**. Thanks to this, the project followed with the analysis of data gathered and machine learning was implemented mainly for insight gaining and feasibility assessment purposes.

The third step, after an evaluation of the requirements of the company, consisted in examining what the scientific literature offered in terms of applications of machine learning for optimisation, that was compatible with the type of data at our disposal and requirements of the company. The result of this third step is the literature overview that follows.

Artificial Intelligence, and in particular a branch of it, machine learning, found several applications in the literature about machine learning and ticket issuing. As a first example Kallis, R., Di Sorbo, A., Canfora, G., & Panichella, S. (2019, September)., in their research developed an automated system for labelling tickets based on machine learning techniques. They adopted an Algorithm, called FastText, developed and open sourced by Facebook which adopts linear model for text labelling with rank constraint and loss approximation. The aim of their research was to build an application that automatically classifies the incoming new issue. The objective at the base of their project was to save working time that otherwise employees of a company might lose in doing it manually. This project helped in focusing the final use of our data. In fact, this is a perfect example of the integration of several layers of optimisation: on the first hand it offers a way for improving information flow. In this way, developing a machine learning algorithm to classify issues is providing useful, consistent, and aligned information accessible any time and by every member of each team. Secondly, the information is simple and does not require any technical knowledge, if not the knowledge of the business core activities. Third, the application of the algorithm only requires the ticket issues dataset and a laptop, hence its application is possible in businesses with limited resources. As we can notice this paper already is a useful hint that our research questions might find positive answers. Nonetheless, with our project, we are interested in producing a strong impact on the business. For this reason, we will use the paper just presented as a starting point, to set the right basis to a more specific optimisation task.

The idea behind the model implemented in the real-case project finds its roots in the work of Lloyd Montgomery and Daniela Damian (2017) researchers at the University of Victoria. Their interest came from an IBM request of designing a way to avoid escalation of customer care ticket issuing, and their purpose was described by the authors in this way:

> *Our work provides a step towards simplifying the job of support analysts and managers, particularly in predicting the risk of escalating support tickets. Through iterative cycles of design and evaluation, we translated our understanding of support analysts' expert knowledge of their customers into features of a support ticket model to be implemented into a Machine Learning model to predict support ticket escalations (*L. Montgomery and D. Damian. What do support analysts know about their customers? on the study and prediction of support ticket escalations in large software organizations. 2017*).*

In their work they extracted relevant features in an innovative way, both internally, cooperating with customer support employees (domain knowledge), and externally, by analysing customer requests and complaints to help in revaluating the requirements of the company. Their results are encouraging, and they are the product of the state-of-the-art procedure for model prediction in the customer care domain. In fact, the model evaluation was made over 2.5 million support tickets and 10,000 escalations, obtaining a recall of 79.9% and an 80.8% reduction in the workload for support analysts looking to identify support tickets at risk of escalation (L. Montgomery & D. Damian, 2017, p.1).

The fourth step of the project will concern the selection of the best model for the data at our disposal. In their work Montgomery and Damian proposed several algorithms that we also adopted as reference point for our research.

In this third paragraph, we explored the potential applications of machine learning in ticketing optimization and how this relates to the real-world application project presented in the second part of the paper. Our focus, as mentioned earlier, is limited to ticket issuance after a customer contact. This results in a more limited scope of the research problem and a limited number of scholarly sources addressing the same research objective. Nevertheless, the literature review provided some insights that may be useful for the real project that is the subject of this paper.First, it helped draw attention to the various ways we could have processed and analysed the data available to us. Second, it helped us put this research under a scientific perspective and approach the challenges with a rational and repeatable structure. Third, it guided us in the method by helping us choose both the right features and the right class of algorithms to process the right data sets.

CHAPTER 2: Artificial Intelligence for information flow and software development optimisation.

In the first Chapter we explained the key concepts underlying this essay and how Artificial Intelligence, and specifically machine learning, can prove to be beneficial for firms both in terms of production performance and in terms of information flow improvement.

In this second chapter, with the aim of improving the ticket resolution risk prediction, we propose a framework based on the integration of artificial intelligence into the software development process. Our framework aims at interactivity and synergy between software development specialists and Research & Development experts: each decision made by the R&D experts depends on specific inputs from Software

Development, while the models and results produced must be interpretable to inform Software Development decision making. Moreover, the Agile method will be presented, with particular focus on the effects of its adoption within companies which develop Software. We will, in fact, explain the main benefits of the adoption of the Agile method when combined to the framework we propose, especially benefits related to information flow improvement and hence the general impact of Machine Learning models in Software Development. To do so, we will first present in Chapter 2.1 the structure of the framework and its contributions. While in Chapter 2.2 and 2.3 we explore deeply the actions involving not only AI experts but also Software Development Specialists.

2.1. Project's framework and AGILE method

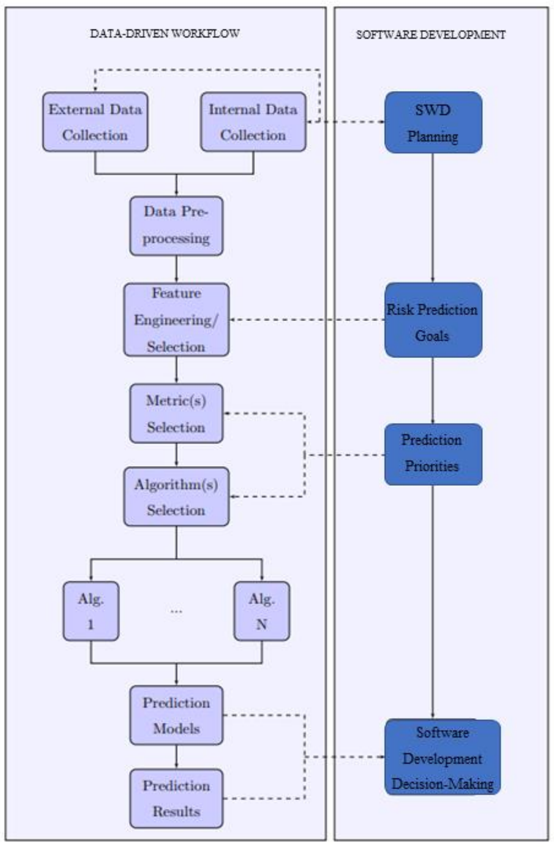


Figure 1. Workflow pipeline

The illustration of the framework we present is shown in *Figure 1* and as we can see, a typical pipeline for implementing predictive algorithms is shown on the left. We start by collecting data from internal and external sources and then proceed to preprocessing the data to assess their consistency. The next step is the selection of relevant information to be processed (Feature selection[4]), and the transformation of this information to improve performance (Feature Engineering[5]). Two other selection processes are necessary for algorithm development, namely metric selection, i.e., selecting the best statistic that measures the quality of our performance, and model

[4] In machine learning and statistics, feature selection, also known as variable selection, attribute selection or variable subset selection, is the process of selecting a subset of relevant features (variables, predictors) for use in model construction. Source: https://en.wikipedia.org/wiki/Feature_selection

[5] Feature engineering or feature extraction or feature discovery is the process of using domain knowledge to extract features (characteristics, properties, attributes) from raw data. The motivation is to use these extra features to improve the quality of results from a machine learning process, compared with supplying only the raw data to the machine learning process.

selection. The latter involves applying one or more machine learning techniques to the data we have, to select the model that best describes and predicts our data set, depending on the metric selected.

In this section of this essay, we want to introduce the core idea of this framework, which, as mentioned earlier, is interdepartmental collaboration. To this end, we examine the connection between the R&D department that developed the algorithm and the software development department, the centre of the optimization process.

Looking at the right side of *Figure 1*, which is the software development department contribution to the workflow, this is our answer to the desired increase in information flow. In this specifically applied framework, which can be generalised however, the software development participated in four steps of the workflow in the optimization process. The first important contribution to the workflow is in the data collection phase; here, software development participates in the planning phase, providing the R&D department with the field knowledge needed to access departmental data and understand the dynamics in their creation. A second contribution in the workflow comes in the Risk prediction goals.

An elevated level of interdepartmental collaboration is required here since the specificity of the optimisation process mandate deep understanding from both teams and hence communication. In fact, the goal of the collaboration in this step is to plan the key performance indicators to be optimised and thus the objective function to be predicted by the R&D-developed algorithm. Depending on the optimisation goal, a metric to evaluate the performance of the model must be selected. To do so Software Development comes in help a third time to assess which mistakes of a prediction algorithm are causing less damages, to focus the attention to the most relevant issues. Finally, the results of the predictive analysis are shared with Software

Development which evaluates the impact of the model developed by R&D and then takes the action needed to deploy the model.
As we remind, our goal, joined with the developing of a Machine Learning algorithm, is to improve information flow between employees of a business. The workflow we presented was designed with this intention, to make all the process, even if data-driven, supervised by Software Development specialists.

The application of this framework was possible thanks to an organisational model adopted in the company, host of the project. The name of the method is Agile method and more than e method itself, is a way of thinking about a software project. One of the inconveniences of this abstraction is that the model is not clearly defined. Nevertheless, one of the key aspects is iteration. All processes of Software Development are repeated from the first deliverable build to the most updated one. This already is a starting point to tackle with change in specifications that might occur over time. Moreover, differently from static and non-iterative methods, it welcomes change, since it is the change in customer’s requests that keep the software at pace with demand, hence giving it a higher chance to resist to time through evolution.

As Greer and Hamonther point out, typical differences with non-agile approaches are the emphasis on small self-organizing teams, the idea of continuous design improvement, test-driven development and continuous integration (Greer and Hamonther, 2011, p.943). The structure of the team matters in the agile methods; typically, they tend to be small and independent teams, aiming at parallelizing the work and make it complementary. In this way the project develops horizontally instead of vertically, and each team works by priorities within it. This approach was typically found with incremental methods. As we can see, for now, agile methods are a combination of incremental and iterative ones, trying to solve the two biggest problem of modern software industry, which are to reduce time wastes, parallelizing tasks and keep up with the constantly changing specifications from the customer's side.

Change is the core driver of the agile methods, so everything must flow in the smoothest way possible to avoid overloads of work. The shift is from the rejection of a method that wanted to finally give clear prescriptions to software engineers on how to approach a complex problem which software development is, to an approach in which creativity is required to solve problems quickly as they occur (Cockburn and Highsmith, 2001), without blocking the whole development chain.

New standards are required to better deal with changing needs. And these new rules must be people-oriented and flexible, offering "generative rules" over "inclusive rules" which break down quickly in a dynamic environment (Cockburn and Highsmith, 2001).

If we must fix a date for the beginning of a serious elaboration of the agile methods, we can find in the publication of the Agile Manifesto in 2001. Beck is one of the most influential promoters of these methods, with him Cockburn and ten other influential names of the software development environment.

Ultimately, the principles defined in the Agile manifesto are the following:

1. Our highest priority is to satisfy the customer through early and continuous delivery of valuable software.

2. Welcome changing requirements, even late in development. Agile processes harness change for the customer's competitive advantage.

3. Deliver working software frequently, from a couple of weeks to a couple of months, with a preference to the shorter timescale.

4. Businesspeople and developers must work together daily throughout the project.

5. Build projects around motivated individuals. Give them the environment and support they need and trust them to get the job done.

6. The most efficient and effective method of conveying information to and within a development team is face-to-face conversation.

7. Working software is the primary measure of progress.

8. Agile processes promote sustainable development. The sponsors, developers, and users should be able to maintain a constant pace indefinitely.

9. Continuous attention to technical excellence and good design enhances agility.

10. Simplicity--the art of maximizing the amount of work not done--is essential.

11. The best architectures, requirements, and designs emerge from self-organizing teams.

12. At regular intervals, the team reflects on how to become more effective, then tunes and adjusts its behaviour accordingly.

Source: https://agilemanifesto.org/iso/en/principles.html

Here some final considerations must be done. The first one concerns the problems that the agile method is trying to solve, which are changing requirements and customer satisfaction. With iterative blocks of features every cycle of a feature goes through redefinition of requirements, adapting rapidly to customer needs. This, on the other hand, may result in a loss of efficiency in the case of static systems, in which one single initial design is more fitting the purpose.

Another trouble that teams adopting agile methods might incur in is unreliability. In fact, lacking a sound documentation some fields as security and health care makes them hardly a first choice. Reliability, though, has been proven in certain circumstances by these state-of-the-art methods. ‘Agile Methods for Safety-Critical Open-Source Software’, describes experiences in the application of Agile Methods to an open-source software project, delivering a safety critical software product. [..]For example, the paper describes additional practices used to manage and trace requirements, the practice of ‘safety-by-design,’ the use of continuous integration and testing facilitated by tool support. [..] Overall, the authors claim from their experiences that agile methods, augmented with ‘the right amount of ceremony,’ can indeed be used for developing safety critical software (Greer and Hamonther, 2011, p.943).

A last problem that by design can arise while adopting Agile methods regards the user experience. In fact, agile methods privilege deliverability over usability and in certain circumstances, where several different functionalities of the software must be implemented overload the engineers, the delivery might come with a cost. Nonetheless, there have been cases such as ‘User Experience Design and Agile Development: Managing Cooperation through Articulation Work’ in which a cooperation between separate teams of development and user experience solved the problem brilliantly, even if with an economical price.

In this section, we explored the working framework of the real-world machine learning application project and explained the choices we made to achieve our goal of improving information flow. We also briefly introduced the agile methodology that helped us build a framework that successfully shares domain knowledge and effectively contributes to the overall performance of the department with an optimization task.

2.2 Strategic planning and data collection

The effectiveness of the proposed framework depends on two fundamental requirements. First, a risk management plan must be designed for the software development task in question. This plan should clarify the objectives and define the scope of risk management. Questions that would need to be addressed with respect to the objectives include what outcome is needed and when, and what decisions are needed and when. In terms of scope, stakeholders should decide whether to focus on the enterprise, on internal or external risks, and on the specific types of those risks. Other important planning decisions include whether certain types of risks cross boundaries and how much effort should be devoted to the process.

The outcome of SWD planning leads to the second requirement of the proposed framework, which is the collection of relevant data. As analysed in the previous section, there are numerous data sources related to SWD, which can be divided into internal and external data sources. Internal data sources include development stage, and production data. External sources are not directly related to the software development and may include customer relations data, such as contacts and interactions. Given the variety of sources, it is of utmost importance to make an informed selection based on the risks identified in Software development planning.
Indeed, the right sources must be selected and the constraints on their use, such as privacy, must be identified. Finally, especially in case of repetition of the task, but in any case, as a good practice, a fundamental step is to plan processes to keep the data stream updated in order to have reliable information at the service of decision making and information flow.

2.3 Feature selection, prediction objectives assessment and Results analysis

The next task of the SWD side of the framework is to give R&D the knowledge \necessary for target specification of the optimisation algorithm. Ideally, this task concerns the objectives that the software development has, and they could be found thanks to previous studies of the data available to it, also in collaboration with R&D to improve the flow of information, or in specific planning requirements. An important challenge that could affect SWD planning in this specific task concerns data collection constraints. Small and traditional companies may not be able to integrate the different data sources captured, resulting in gaps in the reconstruction of the history. Moreover, if the optimization task planned by SWD is not linked to the data available to R&D, the process would be significantly delayed in the best-case scenario. For these two reasons, effective collaboration between R&D and SWD on this task is key to the success of the entire project.

Before applying the algorithm and immediately after data selection, the data must be prepared for the optimization algorithm through data preprocessing. This last step includes several techniques. The first class of techniques is called data cleaning: Here, the main goal is to introduce as little bias (error) as possible into the data set, and to this end, incomplete entries, and outliers (always after assessing the impact) and duplicate entries can be removed (Malley, Ramazzotti, Wu, 2016, p. 115). The second class of techniques is called data integration and deals with the connection between heterogeneous data sources (Malley, Ramazzotti, Wu, 2016, p.115). In the data preprocessing phase, further steps can be applied for the above purpose. R&D and SWD again work together here to extract the knowledge from the data in the best possible form so that it can then be processed by the optimization algorithm.

After the preprocessing of the data, the actual selection of the variables, the so-called features, takes place in the so-called feature selection. Here, SWD guides R&D in selecting the most useful variables or features that the machine learning model will use for risk prediction. In addition, SWD's expertise allows it to suggest slight modifications or transformations of the selected features to best fit the intended requirements.

The final interaction that R&D and SWD have before the algorithm is implemented concerns the selection of metrics and the algorithm. A fuller discussion of this last issue will occur in the next case study; for now, two points stand out. The first is that the SWD must be included in the process, since the selection of metrics strictly depends on the goal of the optimization process. The second point is that the selection of algorithms also depends on the plans of the SWD, since, for example, different algorithms can be applied depending on the main purpose of the task, whether it is prediction or knowledge extraction.

After the predictive algorithm is created, the results are shared between R&D and SWD, and SWD can finally make its decisions based on sound analysis guided by strategic planning and collaboration between departments.

In this chapter, we have examined in detail our proposed framework for software development optimization and, more specifically, for risk prediction. In the next chapter, we present our case study in which the framework examined in this section is used with the purpose of performance improvement and to promote information flow and knowledge sharing using machine learning prediction algorithms.

CHAPTER 3: Case Study.

Chapter 1 Introduced the key concepts presented in this essay while Chapter 2 focused on the definition of the framework we propose for interdepartmental collaboration in case of implementation of Artificial Intelligence techniques. This chapter instead presents our case study, which focuses on the objective of this essay which is adopting Artificial Intelligence, and in particular Machine Learning algorithms such as Random Forest, Logistic Regression and XGBoost, with the objective of predicting the probability that a ticket issue, related to software development tasks, will be solved within the assigned deadline given at the time of its creation.

To do so, we will first present the setting of the Case Study, and the requirements of Software Development. Furthermore, we will present our dataset and the data preprocessing procedures we adopted. Since in machine learning algorithm implementations the results frequently depend from an accurate dataset understanding and domain knowledge, we provide a statistical exploratory data analysis, that with the supervision of Software Development, department object of the optimisation, helped us in selecting and creating features that were as most informative as possible. Finally we show the results of the algorithms we chose which are Random Forest, Logistic Regression and XGBoost and we discuss the performances, considering also the requirements that the firm, and especially the Software Development department had.

3.1 Objective.

The case study we will report now is the product of two distinct internship experiences conducted by the author of this thesis in the same company, Maggioli S.p.a. under the Research & Development department supervision. The main goal of the first and second internship are linked and in particularly the second part concerns the cloud implementation of the algorithm developed in first place.

A first application of the framework we proposed in Chatper 2 can be already found in the definition of the objectives of the first experience. In fact, the definition of the objectives of the internship is the result of the interaction between Software Development department and Research & Development department. After R&D preliminary analysis and reports, an initial assessment of the most relevant problems in SWD were conducted. At this step, then, the necessities that emerged was a deeper exploration of the factor negatively influencing the development process.

The goal of the first internship was to identify, using machine learning, the factor that influenced the most resolution time of ticket issues. In this case, ticket issues are customer requests for modifications or implementation in the software, which is the product focus of this optimization task. The internship objective is too broad to directly build a risk prediction, and thus optimization in this case, model. Indeed, during the first week of the experience, we arrived at the task definition, which was to predict which ticket issues, at the time of their creation, would exceed the desired resolution time. The concept for this prediction exercise came from a review of the Key Performance Indicators, and when SWD determined which ones were the most important, we chose one to use for a prediction task. The KPI chosen specifically concerns the annual respect of the ticket issue resolution time on at least 80% of the issues, according to the priority provided at the time of their generation.

3.2 Data Description

The final dataset used for prediction was the result of several interaction with both Software Development specialists and Data engineer and Database responsible. In fact, in the data collection and feature selection processes a continuous exchange of information was needed to find the most meaningful features to be fed to the algorithm. Moreover, thanks to the exploratory analysis [Chapter 3.4] conducted before the application of the risk prediction model, we were able to individuate patterns within data that contributed to the feature selection process. While in the next sections of this paragraph we will explore deeply the preprocessing steps of our final dataset, in this section we will give a full description of the data used in the application of the predictive task.

Our final risk prediction dataset consists of 32296 rows and 7 columns. Oversampling approaches were used to deal with class imbalance in the target feature, as we will see in more detail in the next sections. The dataset's 7 columns indicate the final attributes used in determining the optimal model, while each of the 32296 rows corresponds to Software Development ticket issues. The first portion of the internship focused on the first months of 2021, and the history of the data that we were interested in was rebuilt from January 1st, 2020 to February 4th, 2021. The decision to choose the first day of 2020 rather than reconstructing the history prior to this data is based on Software Development requirements and, more broadly, the company's current situation. In fact, Software Development reported an increase in productivity in 2020, and searching for patterns and relationships prior to this date would have resulted in biased mistakes. To be more specific, we purposely injected bias into the algorithms with this move, but when modelling static processes, a larger dataset and hence more historical data is usually beneficial. In this case, however, selecting earlier years would be ineffective in properly describing the company's improved production status in that particular year.

As previously said, there are seven final features, and their selection is the product of R&D and SWD departments:

1. Priority: This feature refers to an incoming issue's given priority at the time of its origination. It's a four-level ordinal variable, with each level indicating a higher priority assigned to the ticket issues. The four levels are G1, G2, G3, and R, which correspond to resolution times of 15, 30, 60, and 180 days, respectively.
2. KPI: This feature is the target of our risk prediction algorithm, and it is a binary feature that is a category variable with two levels. The two levels are 0 and 1, with 0 being allocated to all issues that were resolved within the assigned resolution period and 1 being assigned to all ticket issues that were not resolved according to the priority assigned at the outset.
3. Risk is an ordinal variable, similar to priority. In this case, however, it is the result of the feature engineering process, which was derived from the data exploration task's ideas. The levels of this attribute were allocated to ticket issues from various project areas, resulting in a range of typical lifespans that greatly influenced the classifier method. In any case, we will go over the process of its formation and meaning in the next sections.
4. Month_creation: this feature represent only the month of an issue’s creation, hence it is a categorical variable with 12 levels representing each month of the year. For the same reasons we included the risk variable we included this one in the dataset’s features.
5. Azienda: Azienda is the Italian word for firm or business, and this feature contains information about which company was involved in the software development process, either Maggioli S.p.a. or another company for which they produce software. The focus here was on quality assessment to see if internal task performance matched, excelled, or underperformed external task performance.

6. Project_category: This feature contains 39 levels and it is a nominal variable containing information regarding which category correspond to the ticket issue, according to internal classification of SWD.
7. Area: Similarly to project_category, this feature contains information on the area assigned to the thicket issues. It is a nominal variable with 17 levels.

3.2.1 Data Cleaning

As mentioned in the last chapter, data cleaning entails removing any potential sources of mistake or bias from the dataset. Duplicates can be removed, null values can be handled, incomplete entries can be removed, and outliers can be removed.

Data cleaning operations were essentially never performed in this case study, in contrast to what is typically done in machine learning algorithm applications. There are various reasons for this. To begin with, our dataset lacks any continuous features, obviating the option of removing outliers. Second, the data we used was extracted by both R&D and SWD, who had already checked for duplicates or incomplete entries prior to the analysis. Third, during the feature selection process, which included both R&D and SWD, a discussion about the integrity and reliability of data was held, and missing values were found only in features that were not considered reliable to begin with, and thus were omitted since our framework's first step, data collection.

3.2.2 Data transformation

Data transformation refers to the changes made to data based on a specific interest. Data transformations are used for a variety of purposes, including data integration, which requires data from several sources to be analysed together, and data preparation, which ensures that data provided to the algorithm is in the correct format. After a statistical exploratory investigation that identified outliers or unusual behaviours, other probable reasons for data transformation may be discovered.

The reason behind transformations in our dataset was to convert raw data from one data type[6] to another and was applied to all features except Risk and KPI. In using a technique known as labels encoding, features like priority, azienda, project_category, and area were changed from string type to int type. This method produces a transformation from text to integer numbers, which creates a mapping of the variable's categories, or levels, to numbers. For example, if a dataset has a feature named “Name” that contains persons names like Agathe, Gonzalo, Paolo, Michela, Agathe, the final dataset will instead contain 01,02,03,04,01 after the mapping.

As mentioned in the data description section, Month_creation is an extraction of the month number from the date of creation of a ticket issue. The date of creation's original data type was datetime, while the final data type was int, which stands for integer numbers.

Risk and KPI, not being originally part of data collected, did not face data transformations since their design was already accounting for model generation requirements.

3.3 Methodology

3.3.1 Feature engineering

The main goal of feature engineering is to gain the most possible informative potential within the dataset in analysis. To do so, several techniques might be

[6] In computer science and computer programming, a data type or simply type is an attribute of data which tells the compiler or interpreter how the programmer intends to use the data. Most programming languages support basic data types of integer numbers (of varying sizes), floating-point numbers (which approximate real numbers), characters and Booleans.

applied. For example, preliminary statistical analysis can reveal patterns through data such as correlation between variables. In case of algorithms that enforce linear algebra (e.g. linear regression) correlation between features of the dataset introduces no additional useful information for the model, hence transformations such as synthesis of multiple features might be created (engineered). In our research feature engineering was applied for the generation of two features: Risk and KPI.

The first stage in creating a set with the maximum potential for descriptive and predictive power was to develop the target variable, which is a feature of a dataset about which you want to learn more. Using historical data, a supervised machine learning algorithm learns patterns and discovers relationships between other features in your dataset and the target. As previously stated, the outcome of the SWD and R&D planning phase highlighted the need for a more in-depth study of the dynamics that were causing delays in the software development processes of ticket issues' resolution time. We investigated the many possibilities available to us and focused on a KPI that was of particular relevance for SWD, found among the production standards and indicators. It was about meeting the annual ticket issue resolution time on at least 80% of the issues, based on the priority assigned at the time of their generation. We extracted the target feature from this KPI, which is why it's named KPI. It was developed from raw data. In fact, in this feature engineering task, the target variable assumed two levels: 0 for issues resolved within the allocated priority and 1 for issues not resolved within the intended period.

Feature engineering is frequently used after a statistical analysis, which is completed before the algorithm is applied, as was the case in our work. To be more specific, our feature, risk, is the consequence of survival analysis approaches being applied. The goal of survival analysis, which is a branch of statistics, is to estimate the time it will take for a specific event to occur. Using these methodologies, we analysed the projected resolution time of ticket issues across multiple levels of features to determine which features had the most impact on ticket issue resolution time. As a result of survival analysis, and its sharing across R&D e SWD brought up the idea to build a new feature for the final dataset that categorised issues belonging to different types of actions. The original feature was called issue_type and concerned the development process being implemented. For example, some ticket might refer to processes related to correction of errors, some others to transcoding, some others to deployment of new releases and so on. As we will fully examine in the section about Data Exploration, some types of issues were more likely to have a higher resolution time then others and so we built a new feature called “Risk”, with three ordinal levels to capture clusters of issues with similar expected resolution time according to our survival analysis.

3.3.2 Metrics and algorithm

The last major decision to make was how we would evaluate prediction outputs and which learning algorithms would generate these outcomes. To evaluate predictions of classification algorithms, metrics are adopted. Metrics are statistics that quantifies, according to different parameters and purposes, the performance of a classification or prediction algorithm. Since our problem aims to predict if a ticket issue would respect the assigned resolution time, our problem falls in the category of binary classification, which, in turn, restricts the choice of the metric. Nonetheless, several are the available options and the metric choice depends on the objectives of the classification task and on the structure of data. The most general metric for classification problem is called accuracy and it simply measures how many times overall the classification algorithm correctly predicts the target feature. In this measurement although resides an intrinsic problem. If a dataset is highly unbalanced over the target feature, for example with class distribution 99% over 0 and 1% over 1, no matter the predictive power of the algorithm developed, it will perform with an accuracy of 99% by always assigning 0 to all predictions. For this reason, two other metrics, called recall and precision, instead of focusing on the overall correct prediction, focus in the within class ratio of correct positives. Recall (or sensitivity) explains how many of the actual positive cases we were able to predict correctly with our model. On the other hand, precision explains how many of the correctly predicted cases turned out to be positive. Other metrics are frequently used in machine learning algorithms performance assessment, however since we are interested in the individuation of the issues that are more likely to face delays in their resolution, we identified the best suiting metric in recall. In fact, recall is designed to be applied in cases where False Negative is of higher concern than False Positive and in our project, predicting as non-problematic an issue that will overcome resolution time comes with higher costs then a non-problematic issue predicted as problematic.

When it comes to algorithm selection, we must weigh the trade-off between performance and interpretability. Despite our desire to appropriately classify as many problematic issues as possible, we must keep in mind that the internship's primary goal was to examine the factors that negatively affect software development production. As a result, the interpretability of our results is critical, and state-of-the-art classification techniques based on neural networks were ruled out from the start. Moreover, the dimension of data at our disposal is not high and all labels represent classes. This limits the choice of the algorithm since not all algorithms can handle properly categorical data. Our data are not strictly categorical and to handle this, we mapped categories to numbers. The decision of not applying the one-hot-encoding to our data, which is a typical choice in case of categorical data, relies on the large number of different categories of all features we adopted. If applied in our dataset, incurring in the curse of dimensionality[7], we would transform our data in a space so big in dimensions that every point of our dataset would be isolated in this new space, stopping the algorithms to learn similarities within the dataset.

When both interpretability and classification performance are important, classification trees or their ensembled versions, Random Forest classifiers, linear models such as logistic regression, or improved versions of classification trees such as gradient boosting trees are commonly used in computer science. We validated all of them using Cross Validation, a resampling method that uses various chunks of the data to test and train a model on successive iterations based on this assumption ([8]) The models we chose were also in response to a different requirement: the computational power we had at our disposal was limited. In fact, machine learning and statistical learning packages of the Python programming language were used to conduct analysis on a Dell XPS 15 2018 with an 8th generation Intel Core i7 processor. The algorithms we chose in fact are light on memory if compared to other algorithms such as Support Vector Machines or neural network and the number of trainable hyper parameters is restricted. Moreover, they all provide estimates in terms of probability and an evaluation of how much the features are influential in the classification

[7] Dimensionally cursed phenomena occur in domains such as numerical analysis, sampling, combinatorics, machine learning, data mining and databases. The common theme of these problems is that when the dimensionality increases, the volume of the space increases so fast that the available data become sparse. In order to obtain a reliable result, the amount of data needed often grows exponentially with the dimensionality. Source: https://en.wikipedia.org/wiki/Curse_of_dimensionality

[8] Source: https://en.wikipedia.org/wiki/Cross-validation_(statistics)#:~:text=Cross-validation%20is%20a%20resampling,model%20will%20perform%20in%20practice.

algorithm predictions. The dataset at our disposal presented severe unbalanced classes were 94.5% of issues belonged to class 0 and 5.5% of issues belonged to class 1. For this reason, we adopted an approach called oversampling[9] with an algorithm called SMOTE to reach a balancing of 50/50 over the target variable, and therefore, to meet the requirements of the recall metrics selected. Lastly, a train/test split approach in the evaluation of the model was adopted and the proportion of the splits was of 70/30.

3.4 Exploratory Data Analysis

After the revision of the methodological decisions, in this section we will explore in-depth the characteristics of our dataset that had an impact in the decision of the features and the algorithm we used in the classification process. To discover patterns underlying our data, we used a set of statistical techniques known as survival analysis, as previously discussed. Most specifically, this group of approaches was used to determine which types of issues were more likely to survive (i.e. having a longer resolution time).

The first strategy used to achieve this goal employs a statistic known as the Kaplan-Meier Estimator, which is used to approximate event survival functions. We learned two things because of this estimation: To begin, there is a 7-day average resolution time, with a minimum of 1 hour and a maximum of 899 days. Second, because the Kaplan-Meier estimator generates a survival probability function estimate, it can be plotted for inference as in *Figure 2* below.

[9] Within statistics, Oversampling and undersampling in data analysis are techniques used to adjust the class distribution of a data set (i.e. the ratio between the different classes/categories represented). These terms are used both in statistical sampling, survey design methodology and in machine learning. Source: https://en.wikipedia.org/wiki/Oversampling_and_undersampling_in_data_analysis

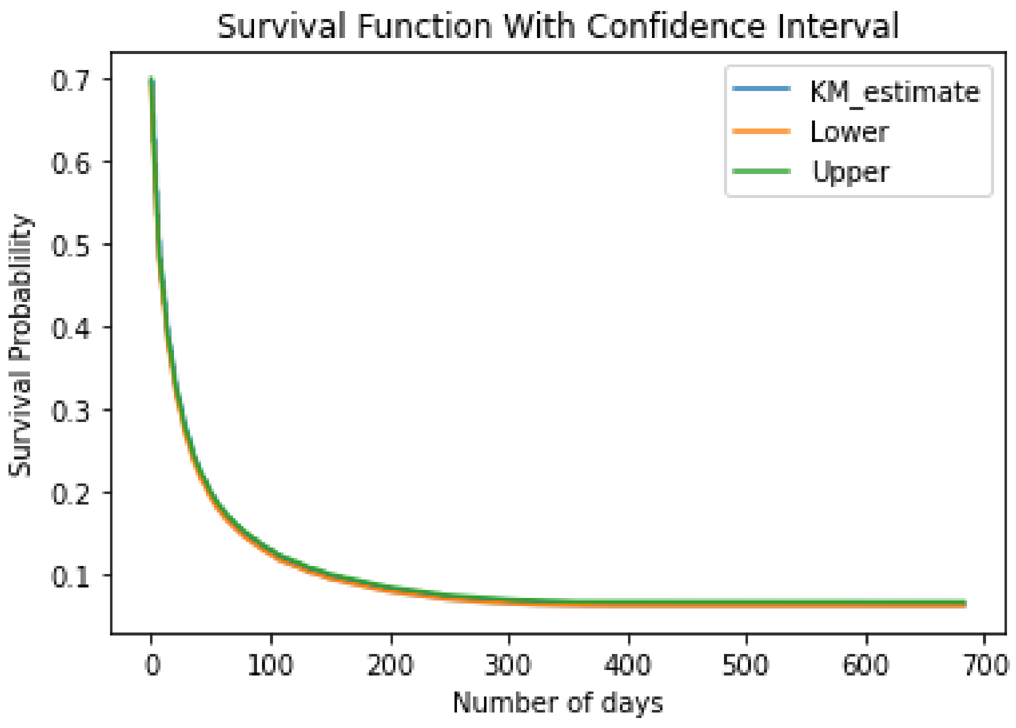


Figure 2. Kaplan-Meier estimated survival probability function.

The Kaplan-Meier Estimation procedure yielded this graphic, which depicts the survival function. The survival probability is plotted on the y axis, while the number of days before resolution is plotted on the x axis. Because the associated survival probability is 0.7, we can see that the likelihood of resolving an issue in one day, which is on the 0 of the x axis, is 0.3. Furthermore, the probability function falls rapidly; in fact, the related survival probability after about 20 days is 0.3, implying that 70% of the issues are handled within 20 days. In Figure 2 the survival function is estimated with a confidence interval of 95% and is represented by the orange line (upper bound) and green line (lower bound).

In our study, we used a comparable estimator that, instead of calculating the probability function of individual events occurring at specific periods, depicts the cumulative hazard of a specific event occurring after a set amount of time. The Nelson-AAlen estimator is its name, and its plot is shown in Figure 3.

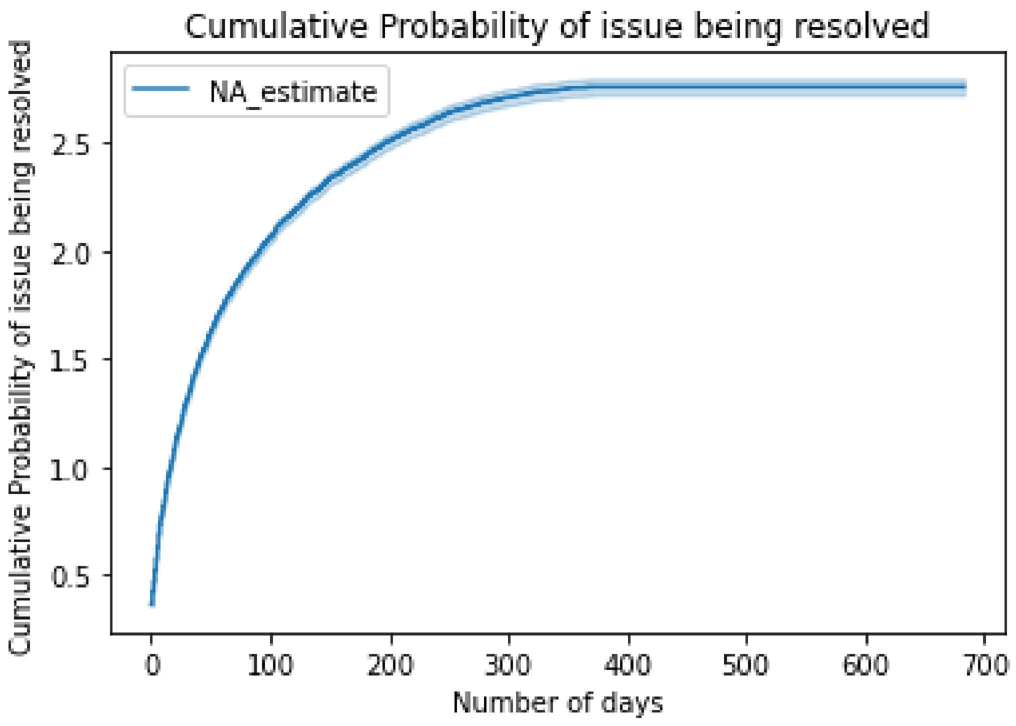


Figure 3 Nelson-AAlan cumulative hazard function plot.

The differences between the Nelson-AAlen and Kaplan-Meier estimators are minor, and we favoured the latter over the former for the sake of knowledge acquisition and interpretability due to the intelligibility of the output, which is probability linked rather than cumulative hazard. In reality, after estimating the survival probability function for the entire dataset, we looked at how the survival probabilities of different categories of ticket issues differed. SWD identified five concerns for R&D to investigate, and the resulting survival probability curves are depicted in Figure 4 in the plot below.

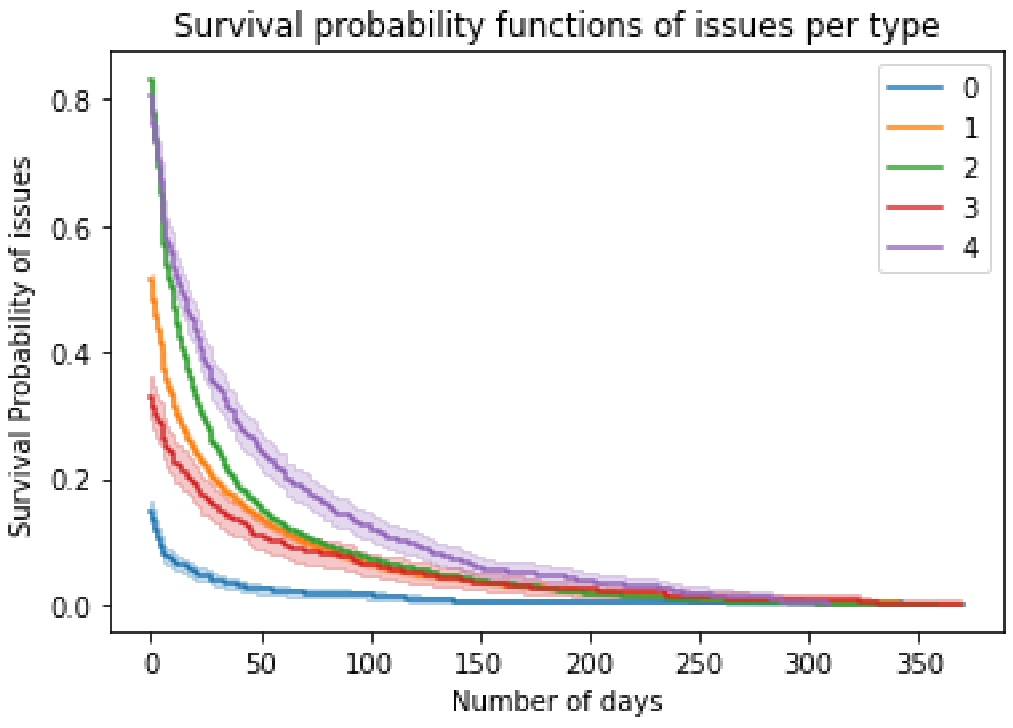
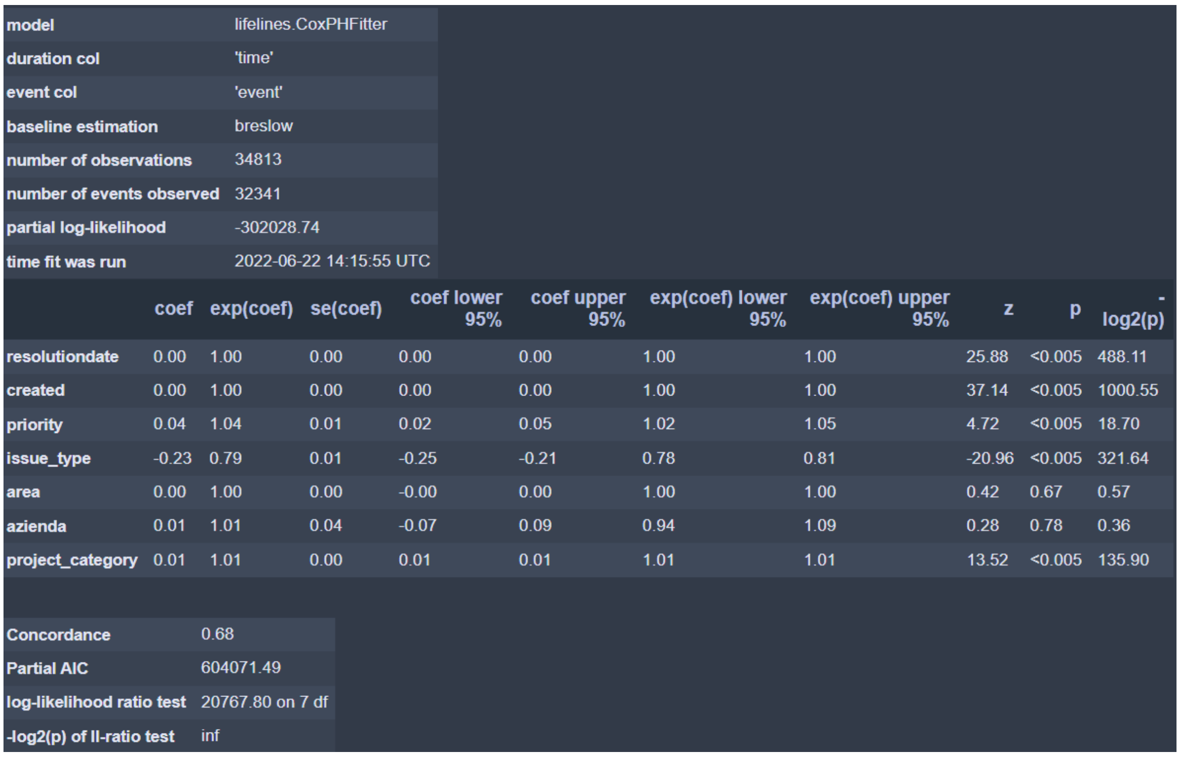


Figure 4. Probability function of survival of issues per type

Different types of issues yield different probability distributions, as seen in Figure 4. Indeed, the issue depicted by the blue line, which belongs to the category of issues number 0, already has a survival chance of 0.17 on day 0. As a result, these issues, which belong to category 0 = Attività, have been recognized as having the lowest overall risk, and will thus occupy the first level of the feature "Risk," which was engineered following the application of this analysis and is included in our final dataset. Purple and green lines, on the other hand, indicate issues in categories

4=Trascodifica and 2=Produzione, respectively, and have the highest and similar expected resolution time, occupying the third level of the final dataset feature "Risk." Issues 1=Assistenza and 3=Rilascio were chosen to occupy level 2 of the feature "Risk," as illustrated by orange and yellow lines in Figure 4.



model	lifelines.CoxPHFitter
duration col	'time'
event col	'event'
baseline estimation	breslow
number of observations	34813
number of events observed	32341
partial log-likelihood	-302028.74
time fit was run	2022-06-22 14:15:55 UTC

	coef	exp(coef)	se(coef)	coef lower 95%	coef upper 95%	exp(coef) lower 95%	exp(coef) upper 95%	z	p	-log2(p)
resolutiondate	0.00	1.00	0.00	0.00	0.00	1.00	1.00	25.88	<0.005	488.11
created	0.00	1.00	0.00	0.00	0.00	1.00	1.00	37.14	<0.005	1000.55
priority	0.04	1.04	0.01	0.02	0.05	1.02	1.05	4.72	<0.005	18.70
issue_type	-0.23	0.79	0.01	-0.25	-0.21	0.78	0.81	-20.96	<0.005	321.64
area	0.00	1.00	0.00	-0.00	0.00	1.00	1.00	0.42	0.67	0.57
azienda	0.01	1.01	0.04	-0.07	0.09	0.94	1.09	0.28	0.78	0.36
project_category	0.01	1.01	0.00	0.01	0.01	1.01	1.01	13.52	<0.005	135.90

Concordance	0.68
Partial AIC	604071.49
log-likelihood ratio test	20767.80 on 7 df
-log2(p) of ll-ratio test	inf

Figure 5. Cox regression over our dataset.

We applied to our data another survival analysis technique, called Cox Regression. Cox Regression It is a cumulative proportional hazard regression that is used to model the time to a particular event based on the values of given covariates. While the previous two analyses focused solely on estimating the survival probability at a certain time, Cox regression models also take into account the factors that influence the resolution time. It covered a particularly important role since it can be used to estimate the linear relations between events and features, and the results were a second aspect that led us to believe that the feature Risk was worth developing. In fact, if we look at the results shown in Figure 5, the feature "issue_type" was meaningful and influenced the resolution time thanks to its inclusion. We are confident in saying that thanks to a recorded p. value lower than 0.005, which gives us statistical confirmation that "issue_type" is influencing the resolution time of the ticket issues.

The last finding we gained from our data analysis was involving the month of creation of the ticket issues.

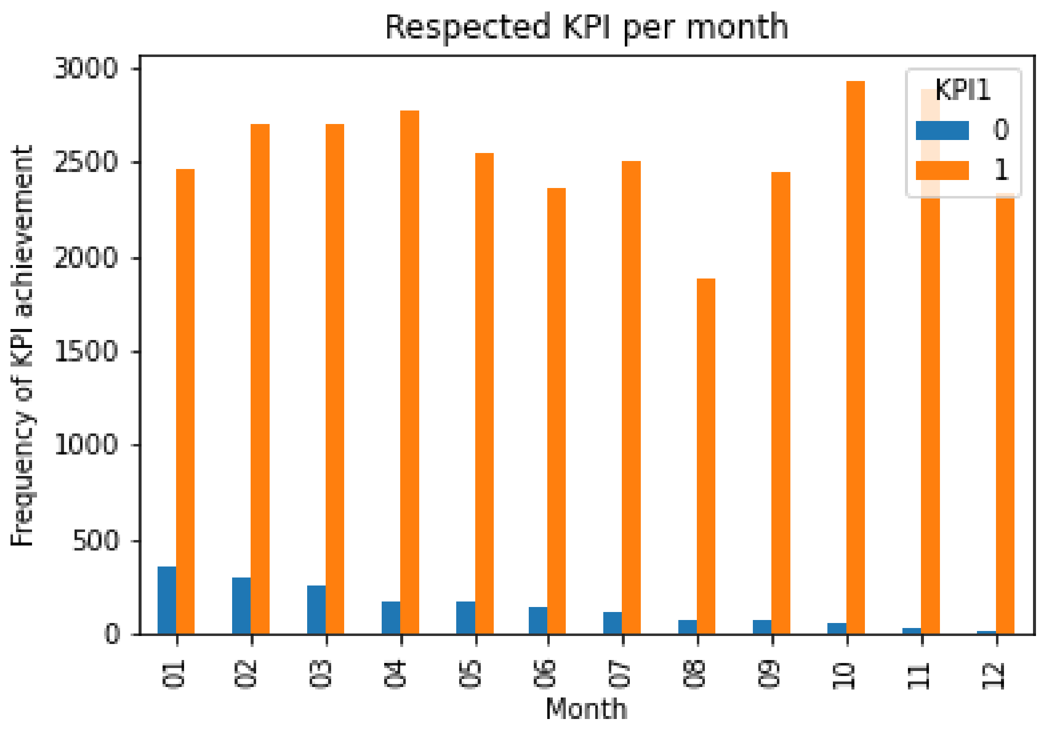


Figure 6. Count of issues solved (KPI=0) and not solved (KPI=1) per month

A decreasing risk of not respecting the resolution time of a ticket issue is associated with the month of its creation. As we can see in Figure 6, in the first trimester, the risk of not solving an issue is considerably higher then the rest of the year and in particular it is double then the second trimester and up to 15 times higher then in the last trimester. For this reason, we included the feature “month_creation” in our final dataset.

3.5. Prediction results and knowledge extraction

We looked at the dataset and the methodology we used in the earlier sections of this case study while the outcomes of the prediction task and the final pick of the best fitting algorithm will be presented in this section. To accomplish so, we compared the performance of various classification models using the chosen metric, which is Recall.

The choice of Recall is based on the fact that we are interested in identifying issues that are more likely to experience delays in resolution, hence we found recall to be the most appropriate statistic. In fact, recall is intended to be used in situations where False Negative is a greater worry than False Positive, and in our project, predicting as non-problematic an issue that will take longer to resolve then predicted has higher costs than predicting as problematic a non-problematic issue. In truth, recall (or sensitivity) describes how many of the actual positive cases our model was able to predict properly.

On the other hand, the choice of the algorithm, accordingly to what described in the method section, was between Random Forest classifier, logistic regression and XGBoost.

Random Forest was shown in our case study to be the most accurate in terms of recall. Leo Breiman invented the Random Forest in 2001 as an ensemble method (it aggregates numerous Decision tree predictors). It uses an effective method called bootstrap aggregation (Breiman, 1996), as well as the random subspace method (Ho, 1995), to grow individual trees into an extremely powerful aggregated predictor, capable of classification and regression, with better generalization error than a single

Decision tree. We tested the algorithm on the test split and optimised its hyperparameters via Cross Validation over 10 folds according to Recall metric. 4 were the trained parameters,

1. min_samples_split [3, 5, 10]: min samples split concerns the minimum number of samples required to split an internal node. That traduces is in a way to control the learning degree of the algorithm. The smaller number of samples the more the tree will try to model each specific ticket issue. In this case we chose small minimum sample split numbers since the combination of categorical variables in the dataset is big and so each single combination could provide important information to the algorithm. Even more thanks to Bootstrap Aggregation (Breiman, 1996) the algorithm introduces random noise to each decision tree to make it more consistent against overfitting. Overfitting is a concept in data science, which occurs when a statistical model fits exactly against its training data. When this happens, the algorithm cannot perform accurately against unseen data, defeating its purpose. Generalization of a model to new data is ultimately what allows us to use machine learning algorithms every day to make predictions and classify data (IBM website[10]).
2. n_estimators [100, 300]: the number of Decision Trees to generate in the forest is controlled by this hyperparameter. There are two reasons for choosing 100 and 300 as options. First, 100 is one of the best criteria for achieving a good trade-off between descriptive power and generalisation power as a rule of thumb. Second, we chose 300 to force the system to extract as much information from the dataset as feasible.

[10] Source: https://www.ibm.com/cloud/learn/overfitting#:~:text=Overfitting%20is%20a%20concept%20in,unseen%20data%2C%20defeating%20its%20purpose.

3. max_depth [3, 5, 15, 25]: The maximum depth is a regularization parameter. It has to do with the greatest depth to which a single decision tree can go. The more max depth increases, the more the tree will try to split into numerous tiers of nodes in order to learn as much as possible from the input. We trained multiple parameters because we did not know how well the algorithm would handle overfitting before using it.
4. max_features [2,3,4,5]: It's a parameter that controls the model's randomness. In reality, if max feature is set to 2, the decision trees will only look at two features instead of all of them when deciding on the best potential split of a node. We attempted all the available situations excluding the extreme cases 1 and 6 because our features, excluding the target features, were 6.

We also trained, as already introduced, Logistic regression models and XGBoost Models, and we adopted the same procedures of cross validation and fine-tuning. Especially, Logistic regression was optimised with recall as objective according to hyperparameters such as penalty, ([“l1”, ”l2”, ”elasticnet”]), the regularization parameter, max_iter ([50,100,150]), which is the number of iteration the algorithm does to achieve convergence, and C ([0.1, 0.3, 0.5, 1],) which is the cost function. Best results were achieved with parameters {'C': 0.5, 'max_iter': 100, 'penalty': 'l2'} with a mean test recall score of 0.788, ranking third. Full results can be found in the Appendix in Table 2 and the code used for implementation in Code 3 in the Appendix. Ranking second in terms of recall results was XGboost. XGBoost is an algorithm belonging to scikit-learn package of python programming language. Specifically, it is a Gradient Boosting algorithm which objective is to solve a loss function, adopting a gradient descent approach, and, similarly to Random Forest, trees as weak learnes, This algorithm strength, and what distinguish it from Random

Forest, is the fact that XGBoost trains its weak learners on the error residuals of the previous iterations, giving more importance to misclassified data entries. Below in Table 1 we can see the full results of the application of XGBoost on our dataset.

Table 1. XGBoost results with CV parameters.

mean_test_precision_score	mean_test_recall_score	mean_test_accuracy_score	param_max_depth	param_gamma	param_n_estimators
0.825	0.923	0.864	25	1	300
0.825	0.923	0.864	25	1	100
0.825	0.922	0.863	15	1	300
0.825	0.922	0.863	15	1	100
0.826	0.922	0.864	25	0.3	100

We fine-tuned our algorithm using cross-validation according to the metric we chose, Recall, and the Hyperparameters we tested were “max_depht” ([3, 5, 15, 25]) which involves the maximum depth of the trees used as learners, “gamma” ([0.1,0.3,0.5,1)], which specifies the minimum reduction in the loss function required by decision trees to split a node, and finally “n_estimators” ([100,300]), which is the parameter that controls the number of trees to use as weak learners. The code of the implementation can be found in the Appendix, in the Code 2 section.

Table 2: top 5 results Random Forest with CV parameters

mean_test_precision_score	mean_test_recall_score	mean_test_accuracy_score	param_max_depth	param_max_features	param_min_samples_split	param_n_estimators
0.826	0.923	0.864	15	5	5	100
0.826	0.923	0.864	15	3	3	100
0.826	0.922	0.864	15	3	5	100
0.826	0.922	0.864	15	2	5	100
0.825	0.922	0.863	15	5	3	100

The algorithm that ranked first according to our evaluation criteria is Random Forest and the results are shown in Table 2. Even if in terms of Recall, our reference metric, Random Forest matched the performance of XGBoost, reaching a Recall of 0.923, it scored best overall, considering also other metrics such as accuracy and precision. In fact, comparing the mean test precision score of the two abovementioned algorithms, Random Forest achieved slightly better results, with a score of 0.826 instead of 0.825. We can notice also that the best parameters for XGBoost are:

{'gamma': 1, 'max_depth': 25, 'n_estimators': 300}. On the other end, the best parameters regarding Random Forest were: {‘max_depth’:15, ‘max_features’:5, ‘min_sample_split’:5, ‘n_estimator’:100}. Overfitting is a major risk in the use of machine learning algorithms, and it is also a concern in our situation. In fact, perfectly described train datasets lack generalization capability over the test set. Comparing the best hyperparameters chosen for XGBoost and Random Forest, we can notice some differences. First, if we consider the chosen values for parameter max_depth, it is lower for Random Forest (15) then it is for XGBoost (25). This means that to reach the best performances, XGBoost needed to build deeper trees to model data and hence, trying to capture sparse behaviours (overfitting) and hence the generalisation power of this algorithm might suffer. Random Forest instead achieved best performance with the same hyperparameter set at 15, which is a good signal to assess that the model is not overfitting. Another comparable signal comes from the number of trees generated. XGBoost needed more trees to match Random Forest performances as it required 300 trees instead of 100, implying that Random Forest described our problem better in terms of recall with a lower effort.

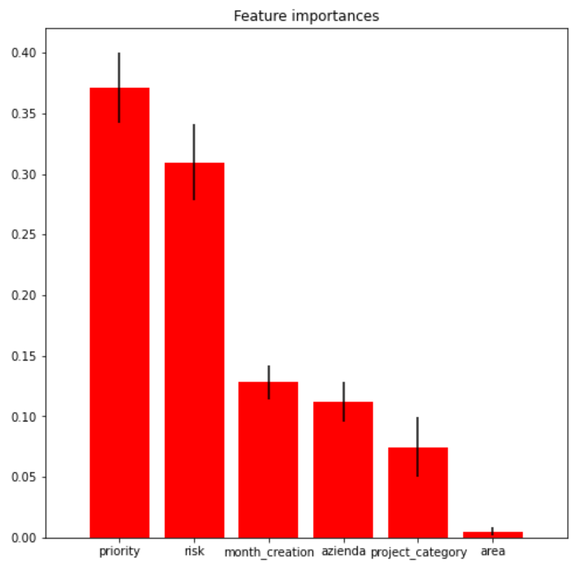


Figure 7. Random forest feature importance

These last considerations guided us in the decision of adopting Random Forest for our classification problem, even if in typical machine learning applications, XGBoost outperforms Random Forest.

The descriptive capacity of Random Forest, particularly the information it gives in terms of feature importance, was the second goal in its application. The mean and standard deviation of impurity decrease accumulation within each tree are used to calculate feature importance, and the plot of the final model's feature importance can be seen in Figure 7 above.

The features that most contributed to describing the classification is priority, immediately followed by risk. As we can notice, the feature that we engineered thanks to the exploratory data analysis findings contributes significantly, as it ranks second, to the classification performances. Another finding that came from the statistical analysis, and after confirmed by SWD, is that the month of creation of an issue influences the respect of times assigned to it. Lastly, features such as area, project_category and azienda, but especially area, did not prove to be as effective in the description of the problem as we initially thought. One hypothesis is that the high number of categories present in the feature “area” created noise in the dataset, making it an unlikely choice for Random Forest in the contribution of node splits.

These insights were shared with Software Development, which, in conjunction with Research & Development's expertise in classification algorithms, received insight into the elements that have the most detrimental impact on the software development process. A presentation about the analysis and classification technique was organized with R&D and Software Development at the end of the internship, and this case study project received positive comments as a result. In fact, the deployment in the production processes was desired, and this was the goal of the second internship experience.

CHAPTER 4: Conclusions.

The findings of the classification algorithm will be briefly summarized in this last Chapter 4, and the performances will then be evaluated in connection to the use of the suggested framework. We focus on the effects of the classification model used in the case study on meeting the objectives of software development and enhancing information flow. The limits of the suggested investigation will then be examined. Finally, additional research is suggested, in part as a result of the constraints depicted.

4.1 Summary of classification results

In order to summarize the case-study results in this paragraph, we will first look at how they relate to the major point of this essay. This essay's first contribution focuses on creating a framework for applying artificial intelligence that encourages communication and information sharing between experts in AI and software development, especially during the phases of strategic planning, data collection, model selection, and target definition, in line with the goals of software development. Since the classification algorithm was implemented within the time of an academic internship, the results we obtained in this regard were encouraging. Second, it was possible to pool the knowledge of both departments to identify the most insightful features for the Machine Learning models because of the ongoing information flow between the departments of Research & Development and Software Development. Thirdly, it provided the company with a framework for comparison that will be used in other projects including machine learning applications, helping to remove barriers to the use of artificial intelligence as a source of optimization and improved information flow.

The second contribution of this essay was the prediction of the probability that a ticket issue, related to software development tasks, will be solved within the assigned deadline, given at the time of its creation. We chose Logistic Regression, Random Forest classifier, and XGBoost as classification algorithms for this task because of their potential for both knowledge extraction and classification performance, as well as their widely-validated implementations in scientific research. The objective parameter of the algorithm fine-tuning process and the metric for performance measurement chosen was Recall, because it is commonly used in cases where False Negative is of greater concern than False Positive; in our project, in fact, predicting as non-problematic an issue that will overcome resolution time comes with higher costs then a non-problematic issue predicted as problematic. Logistic Regression ranked third with a registered Recall of 0.788, while XGBoost and Random Forest matched in performance, registering a Recall of 0.923. Comparing other measurement metrics however, Random Forest outperformed XGBoost, which in turn showed also signs of overfitting. This finally led to the adoption of Random Forest as the best model.

4.2 Limitations and further research

As we mentioned above, the choice of the algorithms relied on the interest of the firm in understanding the dynamics underlying software development, jointly with the interest of optimising those processes with machine learning models. To meet both requirements, state-of-the-art algorithms such as neural networks, who might outperform Random Forest, were excluded from the available choices. Moreover, data at our disposal was limited, in both numerosity and feature count. A particular limitation deriving from our dataset relied on the unbalanced distribution of the target variable, which forced our decision to apply oversampling techniques on our data, which in turn might lead to biased results.

While we reached the objectives of Software Development, our work is only an example of how machine learning could be implemented in the Software Development processes and several can be the methodological improvements for research to follow. For example, the focus could be 1) on data integration; more resources, including, between others, customer satisfaction data or social media interactions might prove to be beneficial both in terms of classification performance and requirements acquisition, driving decision making. In addition to an expanded dataset, 2) techniques belonging to Natural Language Processing to extract relevant information from ticket issues employees' notes could be explored. Since Cloud computing and storage are becoming one of the most adopted solutions for firms completing the digital transition, one other way that future research could explore is 3) the best set up of the deployment phase of machine learning applications. Lastly, 4) unsupervised learning algorithm could be used to derive insights from data, that from a statistical data analysis and a supervised approach, we might have not identified.

BIBLIOGRAPHY

Braman, S (1989)
Defining information: An approach for policymakers, Telecommunications Policy, 13 (3) (1989), pp. 233-242

T. F. Bissyandé, D. Lo, L. Jiang, L. Réveillere, J. Klein, and Y. Le Traon, (2013)
“Got issues? who cares about it? a large scale investigation of issue
trackers from github,” in 2013 IEEE 24th international symposium on
software reliability engineering (ISSRE). IEEE, 2013, pp. 188–197.

Breiman, L. (1996).
Bagging Predictors

Breiman, L. (2001).
Random Forest

Cockburn, A. and Highsmith, J. (Sept. 2001).
Agile Software Development: The Business of Innovation, IEEE Computer, pp. 120-122

Eppinger, S.D. (2001).
Innovation at the speed of information
Harvard Business Review, 79 (1) (2001), pp. 149-158

Greer, D. and Hamon, Y. (2011)
Agile Software Development, in Software: Practice and experience, 2011 - Wiley Online Library

Ho, T. K. (1995).
Random Decision Forests

J. L. C. Izquierdo, V. Cosentino, B. Rolandi, A. Bergel, and J. Cabot, (2015).
“Gila: Github label analyzer,” in 22nd IEEE International Conference
on Software Analysis, Evolution, and Reengineering, SANER 2015,
Montreal, QC, Canada, March 2-6, 2015, 2015, pp. 479–483.

Kallis, R., Di Sorbo, A., Canfora, G., & Panichella, S. (2019, September).
Ticket tagger: Machine learning driven issue classification. In 2019 IEEE International Conference on Software Maintenance and Evolution (ICSME) (pp. 406-409). IEEE.

Khomh, F., Antoniol, G. (October 2018).
"Bringing AI and machine learning data science into operation." Redhat Blog. Available at: https://www.redhat.com/en/blog/bringing-ai-and-machine-learning-data-science-operation

Kumeno, F. (2019).
Sofware engneering challenges for machine learning applications: A literature review. Intelligent Decision Technologies, 13(4), 463-476.

Malley, B., Ramazzotti, D., & Wu, J. T. Y. (2016).
Data pre-processing. Secondary analysis of electronic health records, 115-141.

Montgomery, L., & Damian, D. (2017, September).
What do support analysts know about their customers? on the study and prediction of support ticket escalations in large software organizations. In 2017 IEEE 25th international requirements engineering conference (RE) (pp. 362-371). IEEE.

S. Panichella, G. Bavota, M. D. Penta, G. Canfora, and G. Antoniol, (2014)
“How developers’ collaborations identified from different sources tell us
about code changes,” in ICSME. IEEE, 2014, pp. 251–260

Selfridge, O. (1993)
The gardens of learning: a vision for AI, AI Magazine 14(2): 36–48

Schelter, S., Biessmann, F., Januschowski, T., Salinas, D., Seufert, S., and Szarvas, G. On Challenges in Machine Learning Model Management. Bulletin of the IEEE Computer Society Technical Committee on Data Engineering, 2018

Zhang, D., Tsai, JJP. (2003)
Machine learning and software engineering, in Software Quality Journal, 11, 87–119.

APPENDIX

Table 3 Results Logistic Regression

mean_test_precision_score	mean_test_recall_score	mean_test_accuracy_score	param_max_iter	param_C	param_penalty
0.769	0.788	0.776	100	0.5	l2
0.769	0.788	0.776	150	0.5	l2
0.769	0.788	0.776	100	1	l2
0.769	0.788	0.776	150	1	l2
0.769	0.788	0.776	100	0.3	l2

Code 1: random forest implementation

```
from sklearn.model_selection import GridSearchCV
from sklearn.ensemble import RandomForestClassifier
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer, recall_score,
accuracy_score, precision_score, confusion_matrix


rf = RandomForestClassifier(n_jobs=-1)

param_grid = {
    'min_samples_split': [3, 5, 10],
    'n_estimators' : [100, 300],
    'max_depth': [3, 5, 15, 25],
    'max_features': [2,3,4,5]
}

scorers = {
    'precision_score': make_scorer(precision_score),
    'recall_score': make_scorer(recall_score),
    'accuracy_score': make_scorer(accuracy_score)
}

def grid_search_wrapper(refit_score='precision_score'):
    """
    fits a GridSearchCV classifier using refit_score for optimization
    prints classifier performance metrics
    """
    skf = StratifiedKFold(n_splits=10)
    grid_search = GridSearchCV(rf, param_grid, scoring=scorers, refit=refit_score,
                           cv=skf, return_train_score=True, n_jobs=-1)
    grid_search.fit(os_data_X, os_data_y.values)

    # make the predictions
    y_pred = grid_search.predict(X_test.values)

    print('Best params for {}'.format(refit_score))
    print(grid_search.best_params_)

    # confusion matrix on the test data.
    print('\nConfusion matrix of Random Forest optimized for {} on the test data:'.format(refit_score))
    print(pd.DataFrame(confusion_matrix(y_test, y_pred),
            columns=['pred_neg', 'pred_pos'], index=['neg', 'pos']))
    return grid_search

grid_search_rf = grid_search_wrapper(refit_score='recall_score')
```

Code 2: XGboost implementation

```
from numpy import mean

from sklearn.datasets import make_classification

from sklearn.model_selection import cross_val_score

from xgboost import XGBClassifier

from sklearn.model_selection import RepeatedStratifiedKFold

from sklearn.model_selection import GridSearchCV

from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold

from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer, recall_score,
accuracy_score, precision_score, confusion_matrix


model = XGBClassifier(n_jobs=-1)


param_grid = {

    'gamma': [0.1, 0.3, 0.5, 1],

    'n_estimators' : [100, 300],

    'max_depth': [3, 5, 15, 25],


}


scorers = {

    'precision_score': make_scorer(precision_score),

    'recall_score': make_scorer(recall_score),

    'accuracy_score': make_scorer(accuracy_score)

}


def grid_search_wrapper(refit_score='precision_score'):

    """

    fits a GridSearchCV classifier using refit_score for optimization

    prints classifier performance metrics

    """

    skf = StratifiedKFold(n_splits=10)
    grid_search = GridSearchCV(model, param_grid, scoring=scorers, refit=refit_score,
                               cv=skf, return_train_score=True, n_jobs=-1)
    grid_search.fit(os_data_X, os_data_y.values)

    # make the predictions
    y_pred = grid_search.predict(X_test)

    print('Best params for {}'.format(refit_score))
    print(grid_search.best_params_)

    # confusion matrix on the test data.
    print('\nConfusion matrix of Random Forest optimized for {} on the test data:'.format(refit_score))
    print(pd.DataFrame(confusion_matrix(y_test, y_pred),
                 columns=['pred_neg', 'pred_pos'], index=['neg', 'pos']))
    return grid_search



grid_search_xgb = grid_search_wrapper(refit_score='recall_score')
```

Code 3: Logistic regression implementation

```
from numpy import mean
from sklearn.datasets import make_classification
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import RepeatedStratifiedKFold
from sklearn.linear_model import LogisticRegression
from sklearn.model_selection import GridSearchCV
from sklearn.model_selection import train_test_split, GridSearchCV, StratifiedKFold
from sklearn.metrics import roc_curve, precision_recall_curve, auc, make_scorer, recall_score,
accuracy_score, precision_score, confusion_matrix



logistic_regression = LogisticRegression(n_jobs=-1)

param_grid = {
    'C': [0.1, 0.3, 0.5, 1],
    'max_iter' : [50, 100, 150],
    'penalty': ['l1', 'l2','elasticnet'],

}

scorers = {
    'precision_score': make_scorer(precision_score),
    'recall_score': make_scorer(recall_score),
    'accuracy_score': make_scorer(accuracy_score)
}

def grid_search_wrapper(refit_score='precision_score'):
    """
    fits a GridSearchCV classifier using refit_score for optimization
    prints classifier performance metrics
    """
    skf = StratifiedKFold(n_splits=10)
    grid_search = GridSearchCV(logistic_regression, param_grid, scoring=scorers, refit=refit_score,
                           cv=skf, return_train_score=True, n_jobs=-1)
    grid_search.fit(os_data_X, os_data_y.values)

    # make the predictions
    y_pred = grid_search.predict(X_test)

    print('Best params for {}'.format(refit_score))
    print(grid_search.best_params_)

    # confusion matrix on the test data.
    print('\nConfusion matrix of Random Forest optimized for {} on the test data:'.format(refit_score))
    print(pd.DataFrame(confusion_matrix(y_test, y_pred),
                 columns=['pred_neg', 'pred_pos'], index=['neg', 'pos']))
    return grid_search


grid_search_logistic = grid_search_wrapper(refit_score='recall_score')
```

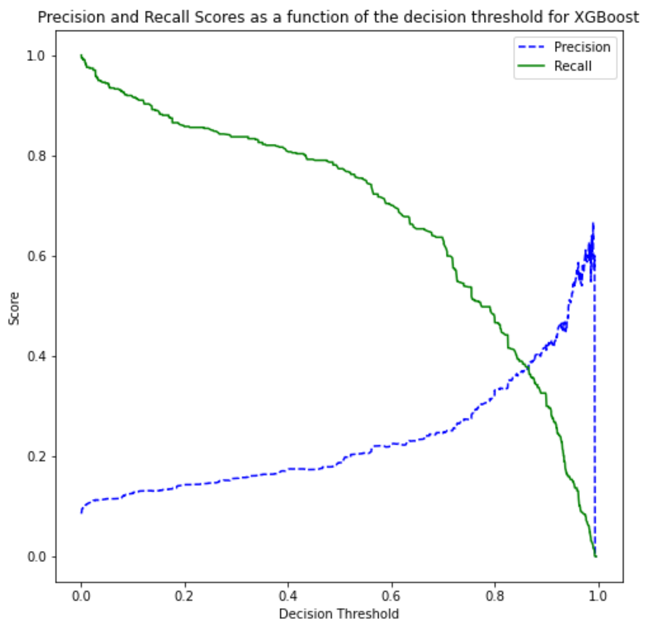


Figure 8. Precision and recall scores as a function of the decision threshold for XGBoost

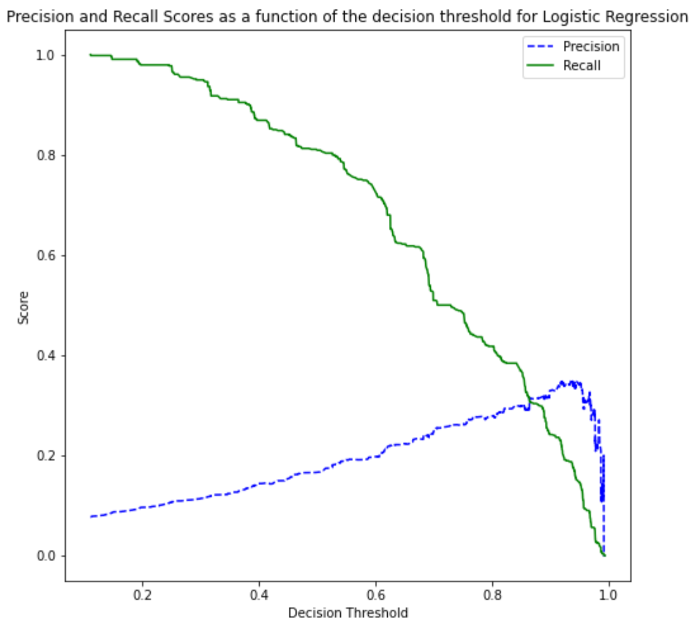


Figure 9. Precision and recall scores as a function of the decision threshold for XGBoost

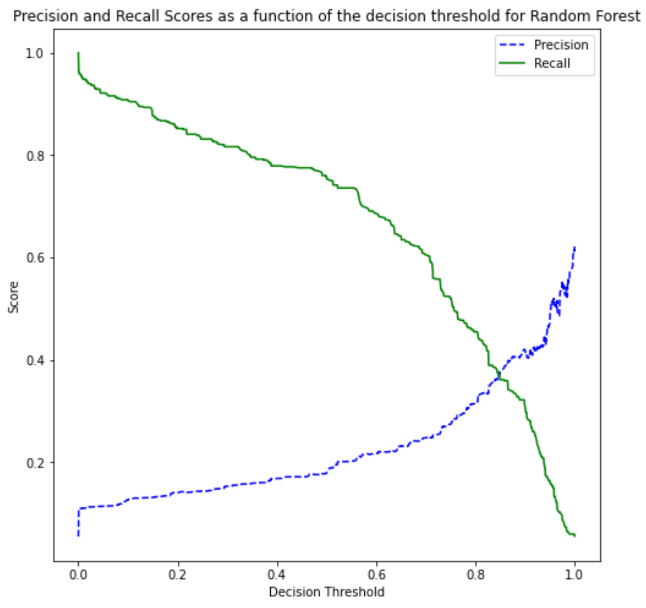


Figure 10. Precision and recall scores as a function of the decision threshold for Random Forest