Product Release Forecasting

Application of machine learning and statistical models on agile processes

Pol Monroig Company

Bachelor Thesis

Specialization in Computing

Supervisor: Nuria Fernandez - Susana Barrera

Tutor: Maria Josep Blesa Aguilera

October 13, 2021

Abstract

Predictability is a very challenging subject but at the same time is very exciting because it can help us learn about the future. We have been tasked with creating a forecasting algorithm to help teams work better. We decided to create different solutions, each with its peculiarities. Some involve statistical models and other machine learning models. The data we are working on is very dirty and needs to be treated carefully. Nevertheless, we managed to develop algorithms with promising results. In the future, improving these algorithms and thinking of new approaches is a must-do.

Resum

La predicatbilitat és un tema molt desafiant, però al mateix temps és molt emocionant perquè ens pot ajudar a conèixer el futur. Se'ns ha encarregat crear un algorisme de predicció per ajudar els equips a treballar millor. Vam decidir crear solucions diferents, cadascuna amb les seves peculiaritats. Alguns impliquen models estadístics i altres models d'aprenentatge automàtic. Les dades en què treballem estan molt brutes i cal tractar-les amb cura. Tot i això, hem aconseguit desenvolupar algoritmes amb resultats prometedors. En el futur, és imprescindible millorar aquests algoritmes i pensar en nous enfocaments.

Resumen

La predicatbilidad es un area muy desafiante pero al mismo tiempo es muy emocionante porque puede ayudarnos a aprender sobre el futuro. Se nos ha encomendado la tarea de crear un algoritmo para ayudar a los equipos a trabajar mejor. Decidimos crear diferentes soluciones, cada una con sus peculiaridades. Algunos involucran modelos estadísticos y otros modelos de aprendizaje automático. Los datos en los que estamos trabajando están muy sucios y deben tratarse con cuidado. Sin embargo, logramos desarrollar algoritmos con resultados prometedores. En el futuro, es imprescindible mejorar estos algoritmos y pensar en nuevos enfoques.

Contents

1 Context and Scope **3**
1.1 Introduction . . . 3
1.2 Approaching the problem . . . 5
1.2.1 Getting to know our workplace . . . 5
1.2.2 Introducing the problem . . . 6
1.2.3 What needs to be fixed . . . 7
1.2.4 Stakeholders . . . 8
1.3 Stating the objectives . . . 9
1.4 Searching a solution . . . 10
1.4.1 Improving the current approach . . . 10
1.4.2 Related work . . . 10
1.4.3 Requirements for a good solution . . . 12
1.4.4 Validating a solution . . . 12

2 Methodology **14**
2.1 Data Collection . . . 14
2.1.1 Accessing the data . . . 14
2.1.2 Keeping up to date . . . 15
2.2 Exploratory Data Analysis . . . 15
2.2.1 Feature Selection . . . 16
2.2.2 Missing values . . . 22
2.2.3 Feature Extraction . . . 22
2.2.4 Outlier detection . . . 23
2.2.5 Other transformations . . . 24
2.3 Proposed solutions . . . 24
2.3.1 Flow Analyser . . . 24
2.3.2 Monte Carlo Simulator . . . 27
2.3.3 Effort Estimator . . . 31
2.4 Code . . . 33
2.4.1 How are modules organized? . . . 33
2.4.2 Resource Manager . . . 34

3 Conclusion and Discussion **35**

A Planning **36**

A.1 Tasks and Time estimation . . . 36

A.1.1 Setting dates . . . 36

A.1.2 Resources . . . 37

A.1.3 Documentation and Project Management . . . 37

A.1.4 Project Review . . . 38

A.1.5 Issue duration prediction . . . 39

A.1.6 Task Scheduler . . . 39

A.1.7 Future Work Forecast . . . 40

A.1.8 User interface . . . 40

A.1.9 Gantt Overview . . . 41

A.2 Risk management . . . 44

A.2.1 Deviation on original plan . . . 44

A.2.2 Inexperience in project . . . 44

A.2.3 Computational Resources . . . 44

A.2.4 Project deadline . . . 45

A.3 Changes from the original plan . . . 46

B Budget **47**

B.1 Staff Costs . . . 47

B.2 Generic costs . . . 48

B.3 Total costs . . . 48

C Sustainability **50**

C.1 Self-assessment . . . 50

C.2 Economic dimension . . . 50

C.3 Enviromental dimension . . . 51

C.4 Social dimension . . . 51

C.5 Sustainability Matrix . . . 52

D Technical competences **53**

Chapter 1

Context and Scope

1.1 Introduction

Project planning and forecasting is an important aspect in all areas of engineering [34]. It is common practice to develop projects in teams, the more the teams are in a project, the higher the complexity, coordinating between different people in different areas of the project can be exhausting and very demanding. Thus, multiple planning and developing methods have been creating, ones that are gaining a lot of popularity are **Agile** [7] methods, specially **Scrum** [43] and **Kanban** methodologies, not only in the software production but also on other non-technical areas such HR and Marketing [32][57][16], they can even be used as a personal development tool. Both Scrum and Kanban [45] are part of the broad list of Agile working techniques, increase productivity and reduce the errors in projects. They are based on a premise that project requirements and direction change over time, thus Agile is made to adapt fast to these changes without any drawbacks.

Traditionally, projects were tracked and developed using what is called a **Waterfall** or **Traditional** model [28], this model divides the entire project in a series of linear steps (i.e., Design, Development, Test) that are done only once; the problem with this approach is that once the design is done, the project is fixed to that design and those ideas, so it makes it very difficult to introduce changes without a major reconstruction and rethinking of the entire project. On the other hand, Agile makes the same step division but it creates an iterative or cyclic approach [34], where each iteration usually lasts a few weeks. At every iteration, all of the phases start again, so it is very easy to design and adapt the project to new requirements, this methodology also enables to review the progress of the project more thoroughly since it evaluates the overall development every cycle. Even though the Waterfall model tends to be more simple [42], it also means it is more rigid.

One of the main foundations of Agile is to deliver a piece of value every iteration this means that in a short period we can have multiple functionalities already developed and tested, whereas with Waterfall we would have to wait for all the initial phases to be completed until we have something valuable. Nevertheless, Agile is not for everyone and there are some cases where you would prefer a Waterfall approach (e.g., small projects), but the future trend is otherwise [41][53][16], after all, who wouldn't want to be agile, right?

Productivity and quality of the project are essential but there is a problem we are not accounting for; we don't have unlimited resources so managing time correctly is of extreme importance, this is where *predictability* comes in place. The predictability of a project is based on the current progress of the project and how much work we have left, this is a very difficult task, especially on traditional project management methodologies. Agile can help us predict our progress better by dividing tasks into multiple iterations, so we can predict when will a list of functionalities be completed. The problem comes when the requirements of the project change, communication problems arise, development issues are encountered, new defects are detected, just to say a few things.

Predictability is especially important for production releases and software that will be released as *products*. Multiple strategies and metrics have been developed over the years to overcome this problem, such as the use of story points, T-shirt sizes, and velocity prediction [14][1][2], in fact, multiple tools such as **Jira** have predictability charts integrated [54], but these methods are usually biased and error-prone to the people that define them [41] (the developer is in charge of defining a story point based on his/her past experience). So why use these metrics at all? Well, they give an estimate on the amount of "effort" required to develop certain tasks, thus for a given team and a given set of tasks we can have an approximation of the amount of work that is required to complete them. Nevertheless that only works in a very limited and short-sighted view, we cannot predict too much into the future and we are tied to the developer that assigned the effort value. Another issue with these metrics can be seen in figure 1.1 where the completion date is predicted linearly, which is unrealistic since projects tend to change over time, so this prediction is very inaccurate at the beginning of the project.

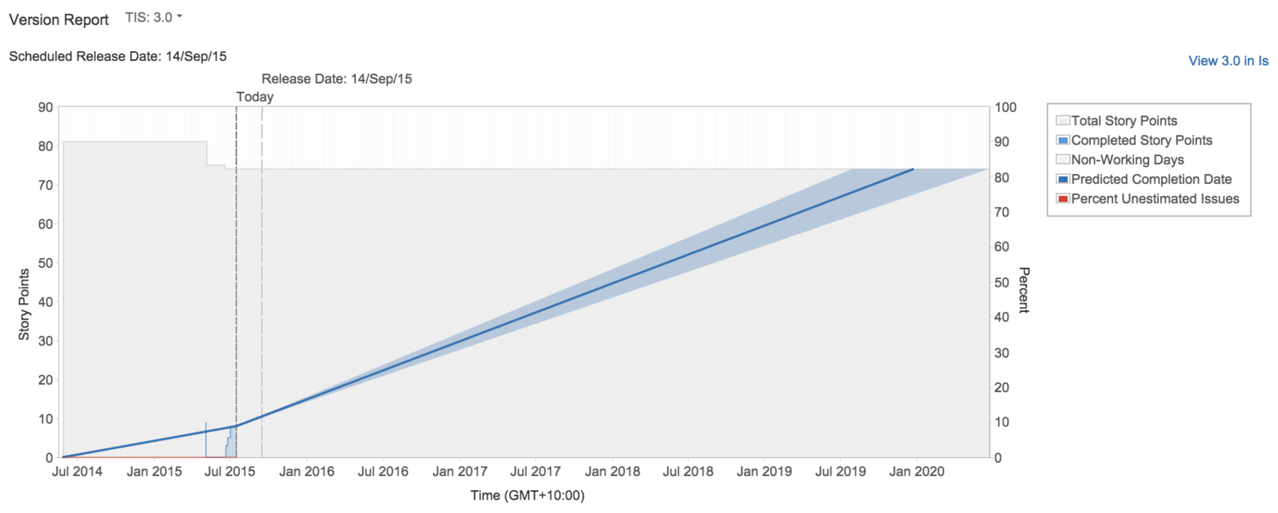


Figure 1.1: Example of how the release of a project is calculated can be found at [55], it shows how based on completed story points and the non-completed ones, a linear prediction is made with a pessimistic and optimistic date.

1.2 Approaching the problem

We have now defined the main concepts and ideas, but we must explain the problems that we are having and the possible solutions that might arise from them.

1.2.1 Getting to know our workplace

Multiple products are being released constantly and concurrently at **HP**, thus it is very important to know the release date of these products. Most products are marked by a deadline but that does not help us to know how much time it will take us to deliver the product (a deadline is the required delivery date, while on the contrary, the delivery date of the product is the date that the product will be finished or at least be completed enough). For all the reasons we stated before and more, HP has recently adopted an Agile model for all of its projects, most products are developed under Scrum, so we will center on those. We have been working in the **Large Format Printing** (LFP) section, which develops large format printers for business and industries, each printer takes an average of two to three years to develop.

Before going any further we must explain the basic foundations of Scrum, for sake of simplicity we are only explaining the most basic characteristics to understand the problem. Scrum [34] is an Agile methodology that divides the project into multiple **releases** of a few months, each release is then divided into multiple iterations called **sprints**, each sprint can last a few weeks; a special characteristic of Agile, is that it is not composed by a strict set of rules, so every company and team works differently, each might develop their own Scrum with different sprint and release lengths [43][4]. As we said before, Agile is flexible and it is specially made to adapt to your needs. A project is divided into multiple tasks or issues, these issues can be divided into 4 different categories, which you might already familiar with:

- **Epics:** they represent the building blocks of the project, epics are created at the start of the project and contain the project itself in broad and general tasks. As the project progresses, these must be divided into sub-tasks of the other types, this process is called *task refinement*.
- **User Stories:** they represent the most basic functionality of a product, there are sometimes called *enhancements* because they add a piece of functionality.
- **Spikes:** they are a type of user stories, but they are not used to add functionality to a product, at least not directly [27]. Spikes are used as research tasks to ask questions or gather information. For our purposes we will consider spikes into the same set as user stories.
- **Bugs:** also known as *defects* , they are known among developers and users alike, they represent problems in the system, things that must be fixed; they are a very clear measure of the quality of a product.

Each task can be identified by a set of features such as a summary, description, acceptance criteria, priority, among others; but one that is usually used is story points, these describe the effort system we talked about in section 1.1. From now one we must consider that each sprint lasts 2 weeks and each release 1 month.

Besides the basic Agile terminology we must describe how projects are organized. The can LFP sector be divided into multiple pieces or levels, each with a different granularity level.

- **Lab:** the research and development (R&D) lab is the maximum level of hierarchy on our current context. It represents a physical and logical place where all printers are developed and where we work with multiple projects. The research lab isn't necessary on the same place, it can be located on multiple countries and sites.
- **Project:** a project has a huge scope. A project does not have a defined end date and thus cannot be predicted, you can only get metrics on it, but these metrics depend implicitly on the programs it contains, thus, we are not going to work in this level directly. The projects we will focus on are Large Format Printing projects.
- **Program:** a program can be understood as a product that must be delivered, a program can be applied to multiple products but they must share many similarities. A program includes a set of tasks that are focused to deliver similar products or a product in particular. Predictability on this level is very important because it will help us get a clear and global view on our products progress, its quality and how likely we are to to finish on time. Unlike projects, programs have a due date, thus they can be predicted and managed. A program can only be worked on a single project.
- **Team:** below programs we can find teams. A team includes a set of people that work in similar roles and try to achieve the same objectives over and over, this gives us a little bit of stability on how they work. Even though a program is made and delivered by multiple teams, a team can work with multiple programs at the same time too, this is because programs are usually very similar products and they need to be worked on the same features in different ways.
- **Person:** finally, the most granulated level we can find is a personal level, where we have information on how each person works. We are not going to make predictions on a personal level either because it is too individualized, the predictability of a specific person is as useful as the predictability of an ant. By this I mean that a worker might change of teams, projects and companies, they are not stable, but an ant colony will have a better degree of stability.

Due to privacy and confidentiality issues, we cannot explain with more detail how projects are organized, what are they called or to what they reference.

1.2.2 Introducing the problem

Multiple techniques such as the use of task refinement, story points as well as multiple data analysis are used to manually estimate the possible time of delivery of a product as well as the JIRA prediction tool we stated before in 1.1 on page 4. Each of these strategies can inform project managers about the progress of the current project and it can help them make decisions. The problem with applying this at HP is that they work with very large teams so every developer, scrum master, project manager and member of the team has a different way of measuring them, and their estimates are not very accurate nor consistent. This can be easily visualized in figure 1.2, where there is no clear relationship between the story points and the duration. It is a very difficulty task, so any tool or prediction algorithm that is being used is not of much use. Another problem is that at the beginning of a project we only have a broad description of the things we need to do, so making a prediction based on vague and general project tasks (Epics) is very difficult, mostly because the project is always subject to changes. Finally, if you manage to predict the delivery date for a product you might need to rearrange resources and adapt to deliver the project faster, maybe drop some features or wait to fix some bugs, but as a project manager, how do you know which issues to drop and which are more important.

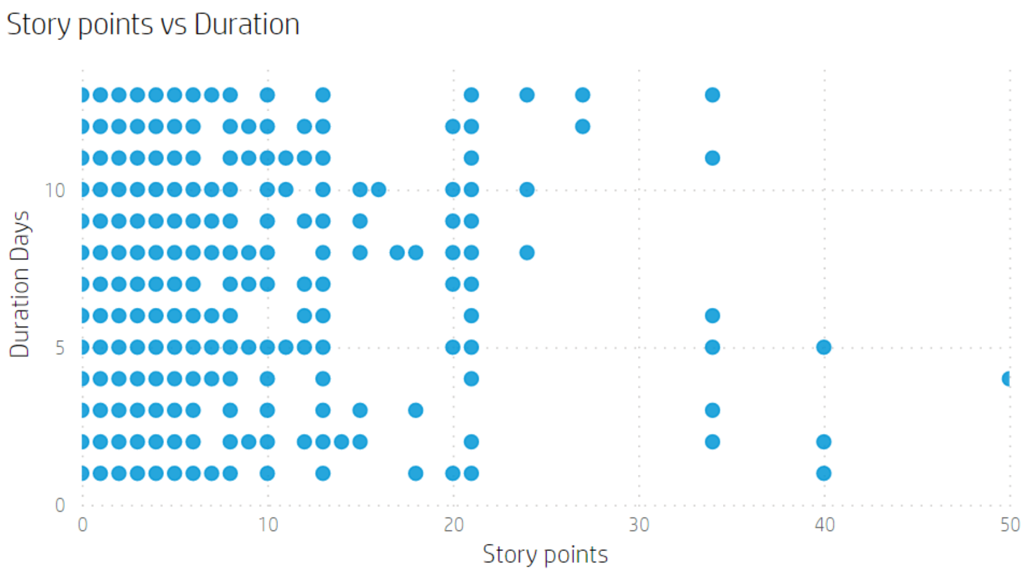


Figure 1.2: Relationship between the value of story points and the estimated duration of tasks [Own Compilation]

Also explain relationship between tshirt size and epic points.

There are other solutions that are being applied that are related to the velocity in which tasks are completed between releases. Based on the demand of tasks that need to be completed by a certain date we estimate the velocity or capacity that we might have until that date. We are trying to predict if we will have time to complete a certain amount of tasks based on the velocity we are going right now. The problem with this approach is that we are assuming that the velocity is always constant (variance is 0), that all tasks are equally difficult to complete and that it does not matter the order in which they are completed. Even though this method is used, there are many ways we could improve it.

1.2.3 What needs to be fixed

After presenting the problem at hand we can identify three main issues that need to be solved

Ⓐ Given the tasks we have right now, we don't know in which order or dates they will be completed. In other words we don't know how the velocity will vary between cycles. (Monte Carlo, estimator)

Ⓑ For each sprint or release we are working on we don't know how well is it planned, and how likely is it to be completed correctly. Most of the time we never completed all the assigned tasks in a cycle. (estimator)

Ⓒ The current prediction is not sentient on enhancements or defects that might arise in the future. This might cause planning problems and introduce out-of-scope tasks into cycles that were not meant to be there.(flow)

Ⓓ We have a difficult time estimating the effort required in each task, thus we cannot make accurate predictions on how much time is needed to complete them. (estimator)

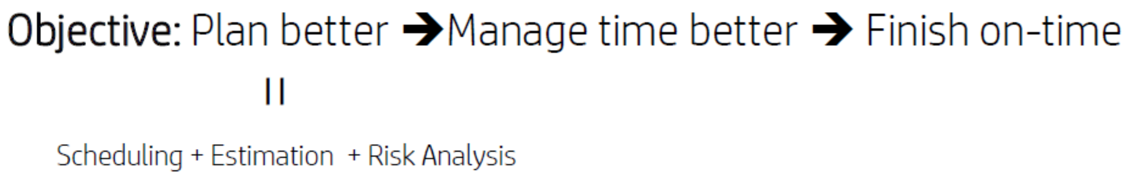
1.2.4 Stakeholders

Both project managers and developers can make use of this project since it will help them organize better and be more aware of the future. They will both be more productive since they will not have to worry about making time estimations. Having a forecast on the current progress of the project can help managers make better decisions and add more stability to the projects. Any person at HP should be able to access this set of tools so any Scrum Master and Section Manager should also be able to consult them, although some of the tools provide low level information making them difficult to interpret.

We have also worked with my supervisor has been working with me side by side, she does not have technical knowledge in the matter so we getting insight and asking technical questions to other employees from the Big Data department at HP. The project is centered on the Large Format Printing area, but they why we are building it enables us to scale it to other projects and areas. Finally other people who will also benefit from a better predictability (although indirectly) are clients who use the printers, since they will get better products.

1.3 Stating the objectives

Once we have the problem defined we can start talking about our objectives. Our objectives will define which direction we will take and its intensity. The objective of the project can be summarized in a simple phrase



Objective: Plan better ➔ Manage time better ➔ Finish on-time

||

Scheduling + Estimation + Risk Analysis

Figure 1.3: Objective keywords

The key in the objective is that we need to plan better, this can be done with three things:

- **Scheduling:** consists in knowing which tasks to do first, which are more important, and when will they be done. We need to define how tasks are spread in time to be able to complete them. Some tasks need to be completed before or after certain checkpoints, so completing them before time is not always a good idea.
- **Estimation:** once we know when we are going to do tasks we would also like to know how complex they are, or in other words, how much time will it take us to complete them. This depends on the task and on the people behind them.
- **Risk analysis:** finally, it is important to assess the risks in completing some tasks, maybe some tasks are worth completing before others, some come with more risks, maybe we need to prevent bugs that might come ahead.

Our objective is to help teams to increase their productivity and predictability, even if we don't get good predictability results, the process we take by analyzing and cleaning the data already gives us a lot of information, information that transforms into feedback which then helps teams become more predictable and consequently improves our algorithms predictions. We need to keep in mind that this is the first time we develop such tool and it will provide insight to many and hopefully inspire others to do the same; at HP, "We are reinventors".

1.4 Searching a solution

1.4.1 Improving the current approach

First of all, we could try to improve the current system by creating a better point/effort system to account for the duration of the tasks but these does not solve all the problems and it adds the additional problem of defining the point system in a way that works. Fixing the way we are working right now does not fulfil the requirements for a good solution stated in section 1.4.3.

1.4.2 Related work

Before getting our hands dirty we must find if someone has done similar work. Has someone already implemented this? Does it apply to us? How can we adapt their solution to ours? This could save us a lot of time and effort. Predictability is a hot topic online but the problem is the most approaches are based on the working methods we are already using (which we have made clear that do not work correctly). Other research [35] focus on predicting story points for each task, this could be useful if not for the fact that the story points in our data are not correctly representative of the duration of a task, even if that was not true, story points are too general to help us predict the overall duration (i.e., how many story points represent 1 day of work?). Another research we found [3] tries to solve a very similar problem, their prediction algorithm (ANDES) works by training multiple machine learning models in an online manner and creating a time date distribution for the delivery date using Monte Carlo sampling, we can see an example of this on figure 1.4; although this project is underdeveloped and it does not show any results. Nonetheless, their paper can give us a lot of information on how to settle our project.

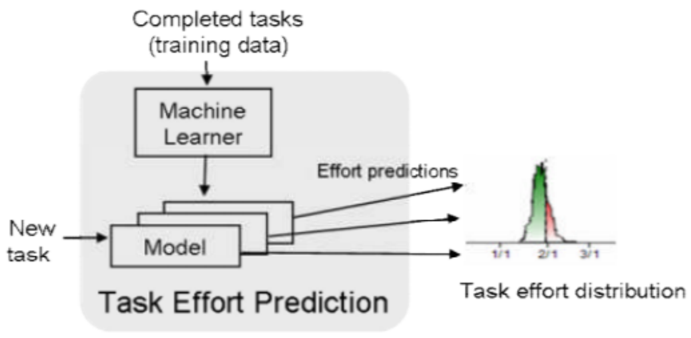


Figure 1.4: Simple diagram that explains how the *ANDES* [3] model works, this general model can help us create the foundations for our project.

In fact, ANDES is not the only project to use Monte Carlo as a technique for project estimation [31]. Although Monte Carlo approaches have good results we have also explored other methodologies to Agile predictability. There is also a metric called the Cumulative Flow, which visualizes the amount of work that is completed vs the amount of work that is created, this gives you a lot of insight on what is going on with the project and can help you find bottlenecks quickly [12][56]. The problem with this metrics is that you need to make the inference and there is not prediction to it.

Other approaches selected for stochastic processes make use of Markov property and specifically the Markov chains, which states that a stochastic process at time $X(t+1)$ only depends on the previous time step $X(t)$ [6], which can also be expressed as

$$P\{X(t+1) = j|X(t) = i\} = P\{X(t+1) = j|X(t) = i, X(t-1) = h, ...\} \tag{1.1}$$

This is a very nice property discovered by discovered by a Russian mathematician called Andrey Markov [33] but unfortunately does not apply to our context where we can sometimes find seasonality and repeated patterns between time steps.

After months of study in this project we realized that the data we are working on has many peculiarities and any generic solution that we could found would not fit correctly, that is why we decided to implement a complete solution.

1.4.3 Requirements for a good solution

We cannot jump and create the best solution out of thin air, there might be multiple solutions so covering all of them could prove very difficult, if not impossible, and we cannot expect to find them right away. What we can do is make a list of requirements that describe a good solution, where each requirement solves a different problem.

1. As a first requirement, we need to create a prediction model (i.e., machine learning system) that generates predictions based on the specific features of each issue, each feature being story points, priority, description, summary, etc. The model must be able to predict for a specific task, its individual duration. A condition that stands out is that the backlog of tasks is done concurrently between different people, so we need an algorithm that can schedule the tasks with their respective predicted duration between the different workers. It is common for issues to have an assignee, but this characteristic is not generally accurate; most of the time an issue is assigned to the responsible of the project and they are not assigned correctly until the task is already completed, thus the scheduler algorithm must be able to handle multiple possibilities and finally output the predicted delivery date based on the generated schedule. This requirement solves problem Ⓓ by getting the most out of the features we have and creating relationships that we could not find otherwise and problem Ⓐ by scheduling tasks.

2. We would also need an algorithm that can simulate different scenarios based on how tasks were completed in the past. The number of tasks that are completed in each cycle is a random variable that can be modeled using a Poisson distribution which describes the number of independent events occurring within a unit time, where each unit is represented by an iteration [6]. In reality, we will have a more complex distribution so we would need a simulation model that can make an estimate of the distribution, its expectation and variance. This is also included in problem Ⓐ.

3. Not all tasks are completed in time and sacrifices need to be made, we need a system that enables the user to analyze the current time and evaluate how likely we are to deliver all the tasks in time. This is a partial solution to problem Ⓑ because we can finish the project in time but we cannot know which tasks are critical for the product, this is only known by the people that develop it and the clients that will use it.

4. Finally, to solve problem Ⓒ we need a way to know the future or at least how will the project evolve, we need an algorithm, that given the history of past projects is able to predict which issues will arise in the future. We can describe this a approach as a discrete-time and discrete-state stochastic process [6].

1.4.4 Validating a solution

How can we evaluate the solution we have? If a forecasting system meets the requirements it means it has solved our problems, so it is in fact a solution, but that does not guarantee that the solution we have is optimal. For each requirement, we need to evaluate the solution and select the one that minimizes the error between predictions and ground truth. We also need a solution that is flexible and can adapt over time to new projects, issues, and features. Finally, we need it to be interpretable by the user, the user should be able to read the prediction, evaluate it and make clear decisions on it, this prediction should provide tools to measure its confidence.

We are working on an extremely complex project, there are a lot of factors involved and the data we are using is not the best there is, so we need to adapt to what we have. We need to be realistic, any solution we might find will not be perfect and it will always be stochastic, the objective is to work from a small solution and keep improving step by step.

Chapter 2

Methodology

Every data science project has a very similar roadmap. The first step is to get the data, this can be as easy as downloading a file from the Internet or as difficult as spending months generating it by hand [17]. After that, it is important to clean the data of any impurities, remove any outliers that can bias our results, and even create simple clusters and classifications. While cleaning the data and after cleaning it, it is of most importance to do extensive analysis, we want to find everything about the data until we found its secrets, or as Ronald H. Coase once said "if you torture the data long enough, it will confess to anything" [10]. Once we know everything we need to know, we can start thinking about which algorithms could work best, optimize their parameters and finally visualize the results.

2.1 Data Collection

The first thing that we need to do before anything is to collect the data, even though it seems like a very simple step, sometimes it involves making tricky algorithms. Here, we will explain how the data is collected, from where and when.

2.1.1 Accessing the data

As mentioned previously, teams work using Agile methodology so all the tasks that they do, when they do them, to whom they are assigned and where is tracked in a development tool called Jira, from Atlassian [25]. We could have gotten this data manually but it would become a tedious and time-intensive task, thus we made use of Jira's Python API [23], although it was poorly documented and we had to read the code to understand most of the things we needed; on top of this we also created another abstraction layer to download the data easily. One of the reasons we created this abstraction layer was due to the fact that downloading the data is extremely slow, and requesting a lot of it at the same time can cause a connection timeout, thus we had to create a paginated query, where we fetched data in multiple steps. To get the data we wanted, we are required to use a query language from Jira called JQL [24].

2.1.2 Keeping up to date

An especially challenging thing about the nature of this data is that we need to keep it up to date always. Every day new data is created and old data is updated, from time to time, features can even change and be completely different, this makes it extremely important to be regularly looking at the data and evolving the code thorough time. These regular changes in the data forced us to create a *cron job* [11] that updates and cleans the data every day, we implemented this in a windows virtual machine and inside a Jenkins server for simplicity [22].

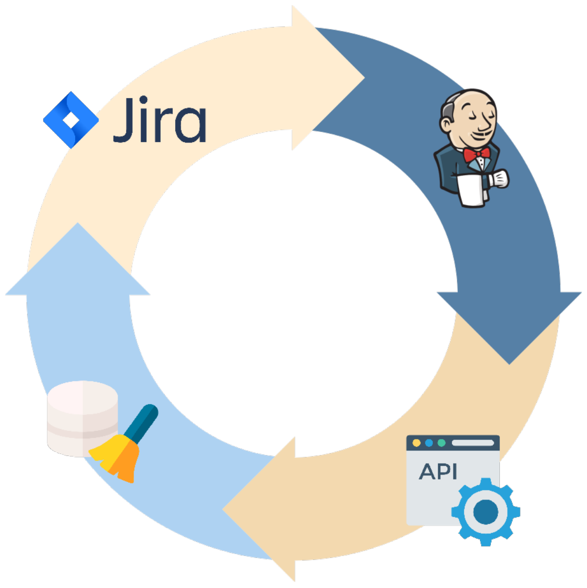


Figure 2.1: The data collection process consists on data updating on Jira, Jenkins sending the API job request and cleaning the data [Own Compilation]

2.2 Exploratory Data Analysis

Before starting to develop any solution and algorithm we spent multiple months studying the data and analyzing it from different perspectives. A particular thing to notice about data analytics is that we can never stop, once we start cleaning the data and prepossessing it, we are already doing data analysis, and after that, we can keep doing it forever or until we think we have enough information.

The main reason we started this exploratory data analysis (EDA) [17][15][50] was to provide the stakeholders a centralized place where they could visualize the project's health and performance. The term EDA was first introduced by the mathematician John Turkey in 1977 where he also created multiple visualization techniques such as the box plot [50][9]. Usually, teams and project managers made their own individual metrics and statistics, but these were made locally. This provoked a disconnection between different teams, which caused repeated metrics and forced them to learn how to create metrics because no one was specialized in this area, thus making their local metrics difficult to share, visualize and usually outdated. This and many other reasons caused us to create a site where we collected lots of metrics from many places and created a centralized website where anyone in the company has access and can share their thoughts about it. This area is always improving and it is what motivated us even further into creating predictability metrics.

As seen in figure 2.1, there is an important step when collecting the data, cleaning it. This step was extremely complicated, due to the fact the data we are using is human-made, it always contains errors, we are also working with a lot of teams at the same time which causes the to data have a higher variation. To understand better how we cleaned the data we have organized it in multiple steps.

2.2.1 Feature Selection

Depending on the algorithm we will be using in section 2.3 we need to use a different set of features. For instance, algorithms that make predictions on a time-series require the number of issues, to which iteration or sprint they belong, the date they were created, updated, and completed, and how are they organized into the different programs and projects (i.e., Flow Analyser and Monte Carlo Simulator). On the other hand, algorithms that make individualized predictions on each issue require a more specific approach and thus more detailed features (i.e., Effort Estimator). Each issue can have more than 20 features, but of all of them, only a few are relevant. These features can be separated into three categories:

- **Numerical:** this refers to features that contain numbers that can be counted (even though a sprint might be numbered it does not have the same numerical properties) such as the story points and the priority.
- **Categorical:** most features are categorical, that is because most of the time, they are used to classify issues into different areas or dates. These include the sprint, the assignee, project, reporter, status, team, and date.
- **Natural Language:** finally, features that can have random natural languages such as comments, descriptions, and acceptance criteria are difficult to parse and provide low-quality information. We were very excited about the fact that we could use deep learning natural language models to our advantage. You only need to listen to what the state-of-art models like GPT-2 say "Man is made by the imagination" [30][38]. Unfortunately, after extensive training we discarded them. We explain this in more detail further down, in this section.

Selected between the different features was easy and complicated at the same time because some features were totally useless and some required an extensive data analysis. Depending on the *issue type* we could have different features available, so we made sure that algorithms and experiments were aware of this distinction, either by separating them manually or by making use of the algorithms feature selection (i.e., an example of this is on decision trees and random forests, which have been shown to be powerful feature selectors [26][13]; they can be even used to generate a feature importance [17])

Features such as the assignee and teams would be very useful but we couldn't use them always because they are not reliable, especially on bugs and epics. The assignee can change through time, you might not know it always, bugs usually ping-pong between teams, and epics don't have a specific assignee because they represent a collection of subtasks. As an example of this, we can see in figure 2.2 a histogram on the number of times an issue is reassigned to different people.

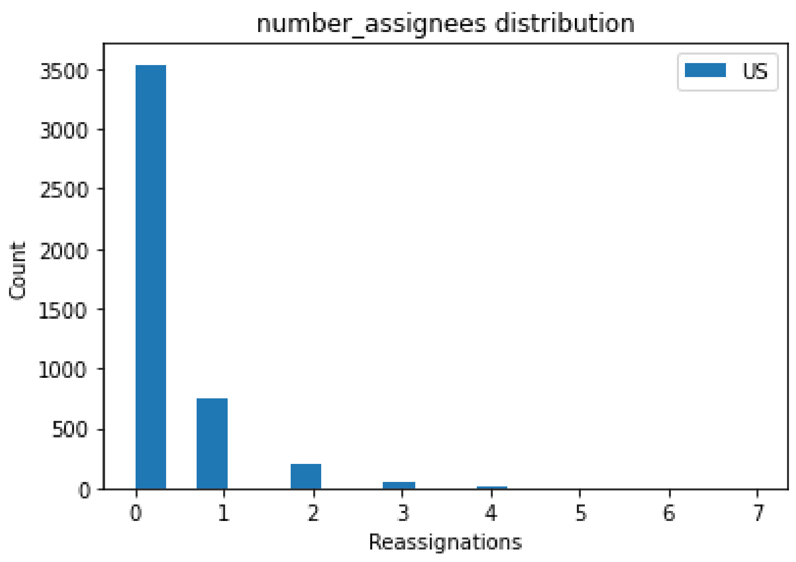


Figure 2.2: For a given program we have a histogram on the number of times a user story changes of assignee, most of the time the assignee is always the same but there are times were a user story can jump among 7 different people [Own Compilation]

Finally, there are other features such as story points that try to estimate the amount of work or the complexity that is required to complete a specific issue. Although as we have stated multiple times before, as well as in figure 1.2, the story points don't help us estimate on an overall level. Based on the team's opinion, it helps them plan their work better, but compared to the real duration of the issue it is not representative. This "real duration" we are talking about is not a real duration, because people don't track the time they take, but an approximation. We explain in more detail and justify its necessity on section 2.2.3.

Discarting Natural Language

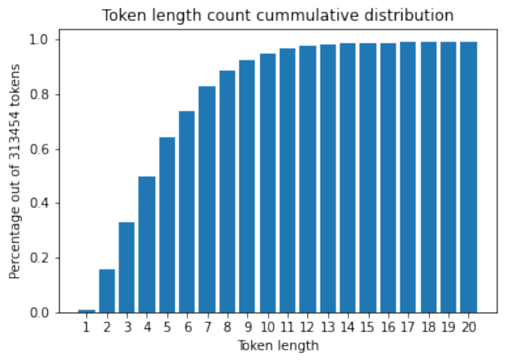
The use of natural language processing would provide us with high-level insight into the complexity of each issue. If we could capture the underlying joint probability distribution using a set of parameters, it would mean that we could interpret tasks using machine learning in the same way a human could do it, but faster and with high scalability. The idea that we can make our machine learning models have a type of representation learning [18], we wanted to create an autoencoder on the whole dataset. The advantage of representation learning is that we could train with unlabeled data, which was much more, and we could later use it as a generative model. It seemed all good and nice but when we started studying the data we found many problems, we've listed the most important:

- **Missing fields:** many tasks had missing or incomplete fields (e.g., description was empty), so it caused an inconsistency.
- **Grammar errors:** most of the tasks had many spelling errors so we had to choose a model that could adapt a large vocabulary of similar tokens.
- **Language inconsistency:** we could even find texts in multiple languages, we tried to fix this using Google Translator API but it was very problematic because it transformed the data

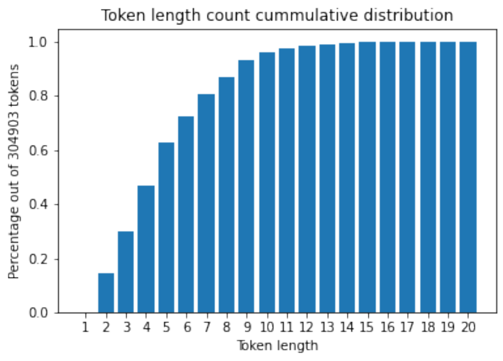
in undefined ways and it was extremely slow because we were limited on the number of API requests [40].

- **Bad filling:** fields were sometimes filled badly or with things that were not supposed to be there (e.g., code snippets on comments)
- **Useless tokens:** there were many useless tokens, such as numbers or identifiers that did not provide information at all.
- **Large vocabulary:** without any cleaning we had an extremely big vocabulary.

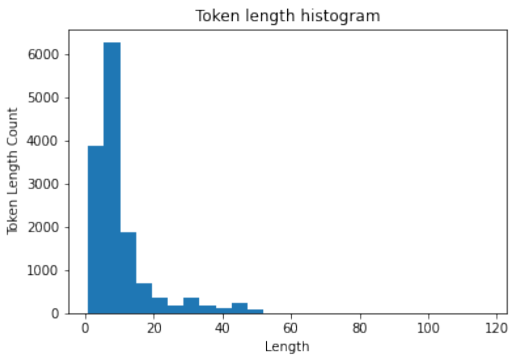
Even after these errors we still tried to train the algorithm. To clean it we added a lot of additional tokens and used FastText as an embeddings model because it is especially known to work with arbitrary vocabulary, and it is especially good at parsing data with grammar errors (i.e. Twitter), this is due to the n-grams tokenizer it uses instead of word tokenizers like most embedding models [58][8]. We tried to use Bert but it added more complexity than we needed and no benefit [49]. Another issue we had was that most tokens were very short and others repeated a lot, so we had to read manually all the possible words and try to find categories such as *Printer* or *Ink.* The cleaning process we made, included text parsing, case folding, pattern substitution, tokenization, stop words removal, outlier removal, and finally word embeddings. Since we finally discarded this solution we will not go into depth on how we implemented every transformation, we will only show the difference after applying them in figure 2.3.



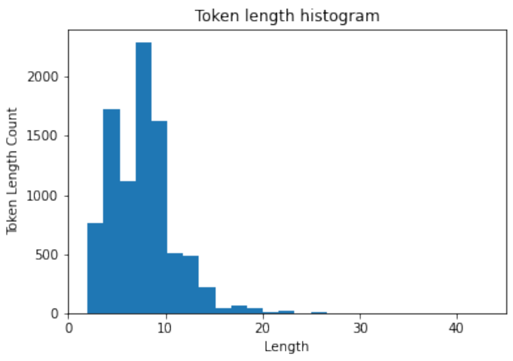
(a)



(b)



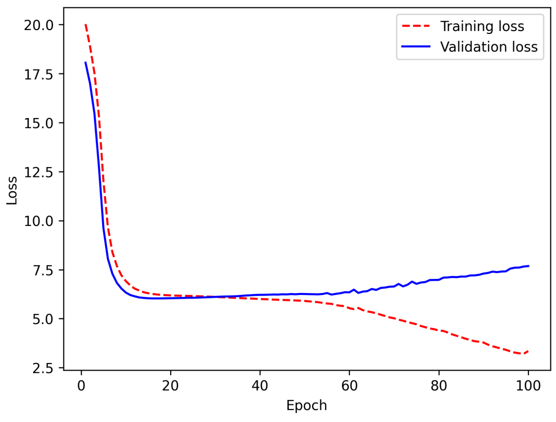
(c)



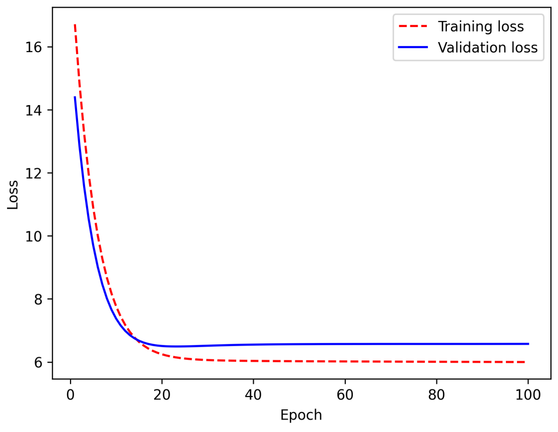
(d)

Figure 2.3: Here we show the difference before and after we cleaned and parsed the words. Of 313454 different tokens, we managed to reduce the vocabulary size to 304903, a difference of almost 3 percent. We also managed to reduce how the word length is distributed, we favored medium words over small and large (some tokens had over 20 characters) [Own Compilation]

Even after all the cleaning, we tried multiple models with lots of parameters and hundreds of experiments but they always had learning problems, either they learned the training data perfectly and it dismissed anything new (overfitting) or both training and validation loss was high (underfitting). We tried many models using different types of memory cells (i.e., GRU and LSTM) but none worked. Trying to change the parameters only changes the rate at which the model overfitted but it always did, this difference can be visualized in figure 2.4.



(a)



(b)

Figure 2.4: Figure (a) shows significant overfitting on epoch 60 with a very high loss and figure (b) shows less overfitting but it does not learn anymore so it is as useful or worse. The difference between the two models was only their complexity and learning rates, so even though the models were different the results were equally bad [Own Compilation].

After all the tests we even tried to find possible clusters of tokens, we did this by applying a dimensionality reduction technique (PCA) to 2 or 3 dimensions which explained up to 36% variance, we needed to add a lot of features to be able to improve the variance and we could only visualize 3. For all this reasons we finally discarded NLP as a forecasting option.

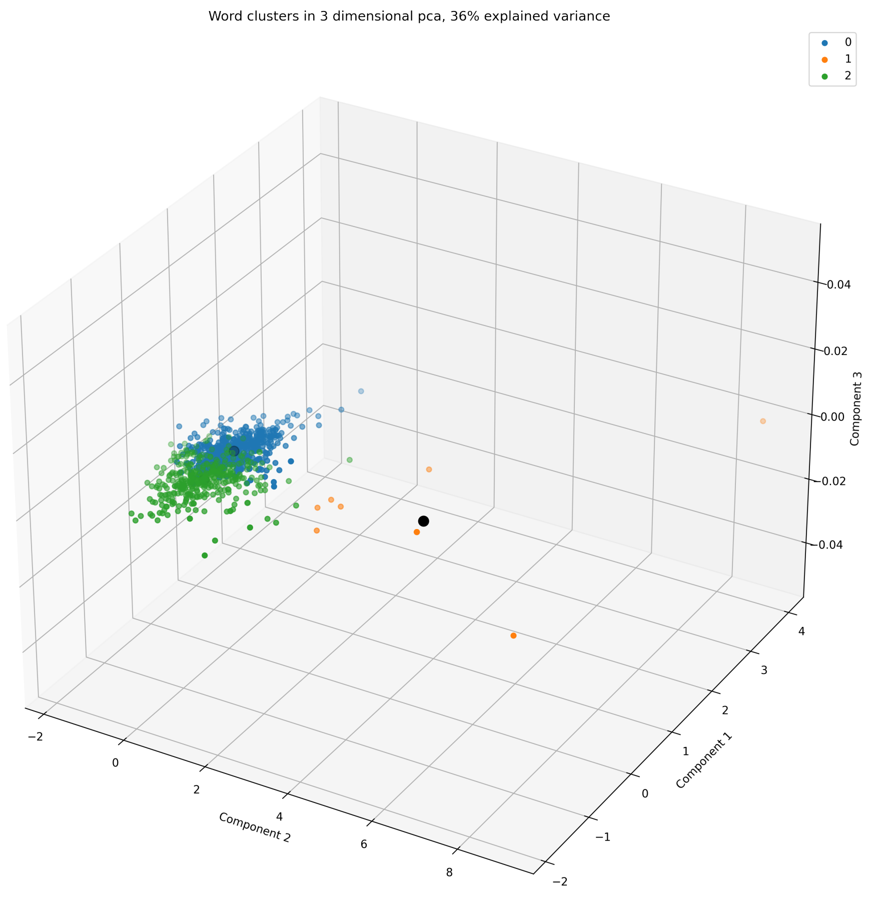


Figure 2.5: Shows how words are distributed based on a 3-component PCA with 36% explained variance, note that we have a high vocabulary size of more than 30 WORDS, but here we show how we can represent 36% of those words with only three real values [Own Compilation]

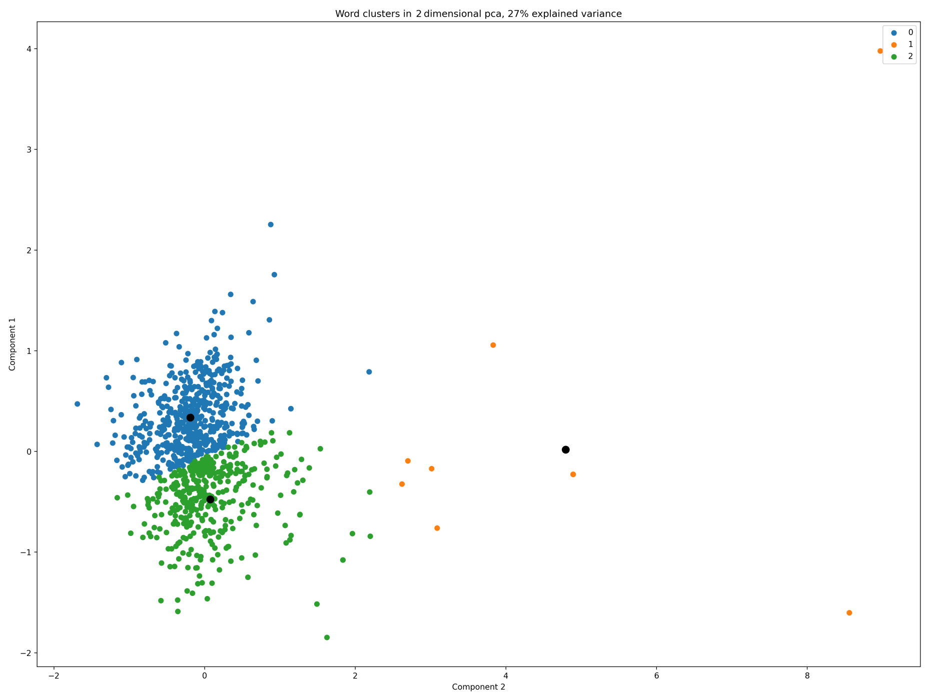


Figure 2.6: Shows how words are distributed based on a 2-component PCA with 27% explained variance [Own Compilation]

2.2.2 Missing values

Due to security concerns, we cannot delve specifically into how we transformed each feature, thus we will only explain which problems are the most common and how we can deal with them. Before starting anything else it was especially important to fill in the blanks. Since most of the features are optional people tend to miss them, which causes errors and confusion. Other values might be written by hand and might contain grammar errors. We also need to infer projects, programs, and teams from a more long and complex text, so we usually apply regular expressions to detect and parse any necessary features. Dealing with empty values changes depending on how we use them, for the *Effort estimator* algorithm we will fill them using a variety of algorithms such as K-Nearest Neighbours or the mean, which can be easily done using Sklearn imputer [44]. Otherwise, we might fill them using an empty label such as ”Empty” or ”No program”.

2.2.3 Feature Extraction

After the basic feature cleaning, we needed to create new features according to our needs. Extracting features could be done in one of two ways:

- **Based on other features:** the most common way to create new features is by using existing features to our advantage, we didn't combine any of the features because it didn't make any sense in our case, but we did modify them. Some of the features we created were thanks to advanced text parsing where we found information in natural language, similar to what we explain in the previous section (i.e., an example of this is the sprints, projects, teams, and programs). This can sometimes prove very difficult because the characteristics of these features tend to change in a short period of time, so we need to be prepared for everything. Another feature we created was called *Epic points*, this is equivalent to story points but for epics. The epic points try to estimate how big an epic is in number of sub-tasks and complexity, this is calculated using a point system assigned to each T-shirt size. Epic points seem very useful in predictability but have the same problem story points have; furthermore, epics don't have T-shirt size (see figure 2.7).
- **Based on the history:** Apart from the basic features that a task might have, there is also an additional component called the *history*, this history contains very important information because it contains how the issue has evolved through time; this gives us information on how its status changed as well as all its features.

As promised we will now explain how the duration is calculated. The duration of an issue is not tracked so we calculated based on the history of the issue. On small tasks such as stories and bugs the duration is the number of workdays between the date its status changed to *In Progress* to when it changed to *Resolved*; a typical duration might last between 2 to 10 days. Big tasks like epics depend on user stories, so their duration is the sum of the duration of all its sub-tasks. This estimation might have many outliers and might be wrong in some cases but it is the only time measure we have from the tasks, so for the sake of argument we must assume it is a good estimator; to be sure this was

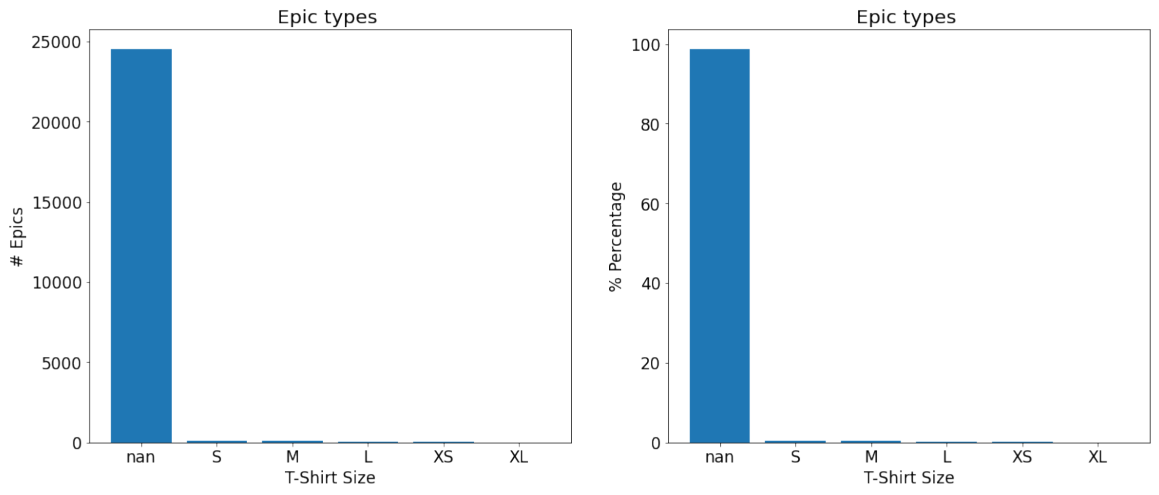


Figure 2.7: Figure shows how the number of epics with tshirt size is extremely skewed and almost non has. The ones that have are too few to use them for predictability [Own Compilation]

good enough we compared how much tasks where completed in a fixed interval (i.e., sprint) to their estimated duration and found out that the error was minimal, we also made further experiments by making predictions with the *Effort Estimator* before the issues were completed and compared the results, which also validated the hypothesis.

2.2.4 Outlier detection

The data we are working on is extremely variate and we need to be able to distinguish between useful and garbage data. To differentiate it we created the following criteria

- **Date of creation:** and important thing to notice is the date of creation, oldest tasks are part of a migration to agile, thus people were not working correctly with the software so most of these issues were badly organized and had many errors. They didn't have the same tracking or same features as now. As a rule of thumb, we took issues that were at most 1 year old.
- **Duration filtering:** another important filter is the duration, by eliminating issues that were not inside a reasonable range. A reasonable range is represented by the duration of a sprint, approximately 10 days since we added a little margin. Issues that were completed too fast were also eliminated. The duration range was also based on the opinions of the different team members since they are the ones that should know better.
- **Query filtering:** another thing we were very careful about was filtering issues based on their characteristics, this was done when we fetched the data using Jira's JQL.
- **Data picking:** finally, a thing that was very important was that we hand-picked the projects and programs we wanted, even though there are hundreds of programs we only kept the most important and the ones that we knew would train the algorithms better.

2.2.5 Other transformations

After everything, we also made special transformations for the *Effort Estimator*. To be able to use machine learning algorithms we encoded categorical features to numbers and normalized everything to a $[-1, 1]$ interval using algorithms a MinMaxScaler and an OrdinalEncoder [39][36].

2.3 Proposed solutions

Multiple people such as Niels Bohr mentioned "prediction is very difficult, especially if it's about the future!" [21] that is why we have designed three innovative solutions. Each solution will give us a different point of view and will enable us to increase our confidence in the future. Each solution provides multiple outputs of different types (e.g., flow analyzer provides both a flow diagram and a velocity summary). Each of these solutions will be applied on a project, program and team level.

2.3.1 Flow Analyser

From the cumulative flow makes predictions using an Autoregressive Integrated Moving Average (ARIMA). Based on how tasks are created and closed in time. This solution provides a perspective on how will the products evolve in the future, it does not go into very details on which tasks and when are they completed but which is the closed/opened growth. The number of tasks that are created each day is a stochastic process $X(t, \omega)$ with the discrete state because the variable $X_t(\omega)$ is discrete at each time t and discrete-time because the set of times is discrete, that is, it consists of separate, isolated points [6]. But why do we want a stationary time series? Time series can be broadly classified into two types, stationary and non-stationary. Non-stationary are usually random walks [6][47] since the pattern over time changes randomly. As defined in [20]:

> A stationary time series is one whose properties do not depend on the time at which the series is observed. Thus, time series with trends, or with seasonality, are not stationary — the trend and seasonality will affect the value of the time series at different times. On the other hand, a white noise series is stationary — it does not matter when you observe it, it should look much the same at any point in time.

A stationary time series has special properties because since it does not depend on the time we can make predictions based on its previous value. On the other hand, random walks are unpredictable, a good example of random walks can be found in stock markets. This is similar to what we talk about stochastic processes earlier.

There are different ways we can prove seasonality, in our scenario, we tested the autocorrelation plot and the Augmented Dickey-Fuller test (ADF) and the Kwiatkowski–Phillips–Schmidt–Shin (KPSS) test. We are working with a lot of sets of data so we are only going to show a representative of the tests here. Figure 2.8 shows us how the time series tends to zero over time, we can also see that there are a few lags that go outside the threshold and will be representatives of the time series, and will be useful when we use the ARIMA function. The statistical test of ADF and KPSS are common tests used to find out if a time-series is stationary or not; for instance the ADF test the hypothesis that the unit root is present in the time-series, a property inherent in non-stationary time-series. tries to we have not included the results because there are different results for us to count, but we can say that they showed a significant value to rule out the null hypothesis. Sometimes the test did not rule out the hypothesis, depending on the data and the range of discretization we are using but we had assumed it showed good predictions because most of the time it worked; and this is not a scientific experiment, the purpose of this project is to get the best results as possible while sacrificing the minimum.

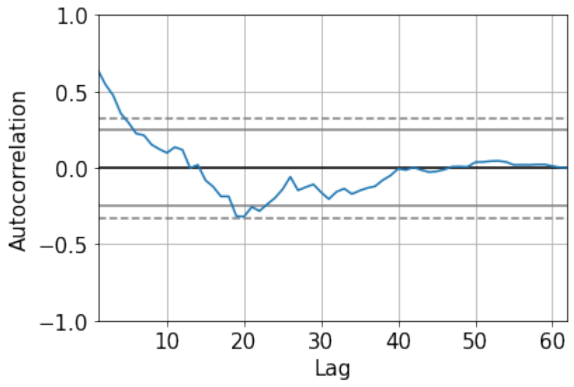


Figure 2.8: this is just an example on a specific dataset of how the auto correlation of a time-series. Every autocorrelation diagram is different but when we have enough data they look similar to this one. Knowing which lags are over the threshold will help us find out the p and d parameters in the ARIMA models [Own Compilation]

Once we tested the possibility of a stationary time series we were determined to run the ARIMA algorithm, an excellent algorithm used in time-series forecasts, since the data showed very promising features we didn't want to venture into more complicated algorithms such as neural networks or other non-linear models, moreover the results were very promising with these simple models. Since we could not fine-tune the ARIMA parameters for each of the datasets we created a train-test separation and made a simple grid-search where we looped between the three parameters and chose the ones that minimized the error, this search was always $O(n^3)$ in time complexity; figure 2.9 shows an example of this process, where we plot both the prediction and the ground truth, you can clearly see how great the results are. The final results can be visualized in figure 2.10, where we show 2 predictions, one prediction is on the issues created and the other on the issues closed, this way we can see both trends and try to find which grows faster and when will they meet. Since we could create issues all the way to eternity and the model does not know that we need to define a deadline where we supposedly stop creating issues, thus the final delivery date will be determined by the number of issues that are still open and the speed at which we can close them.

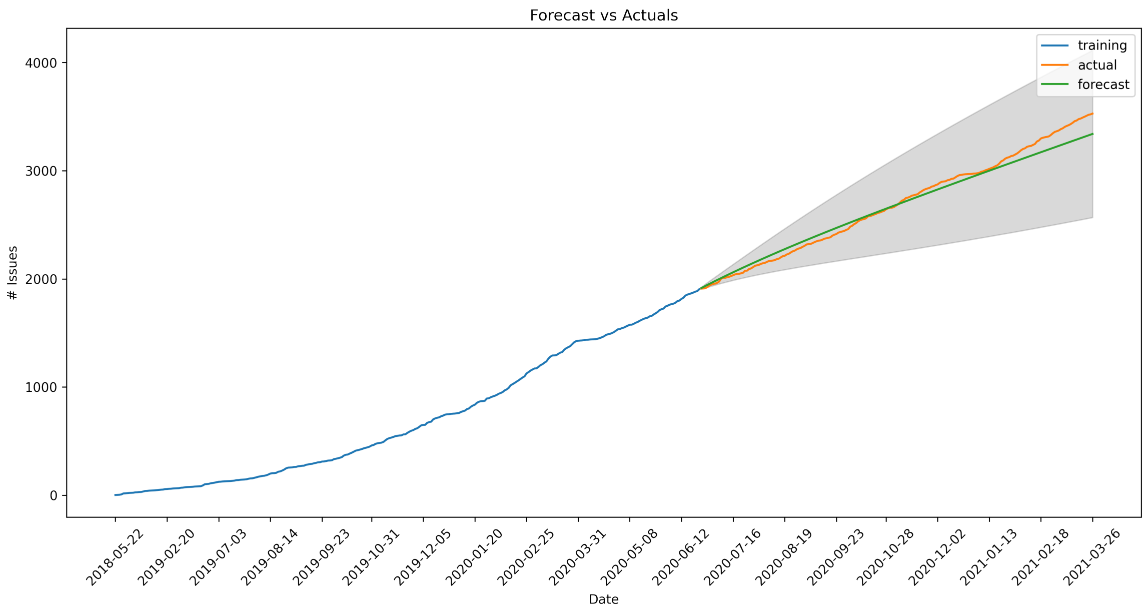


Figure 2.9: Shows a sample on how the algorithm tries to find the best parameters by separating the data into a train-test split, and making a prediction. The gray area shows the 95% confidence interval on the forecast [Own Compilation]

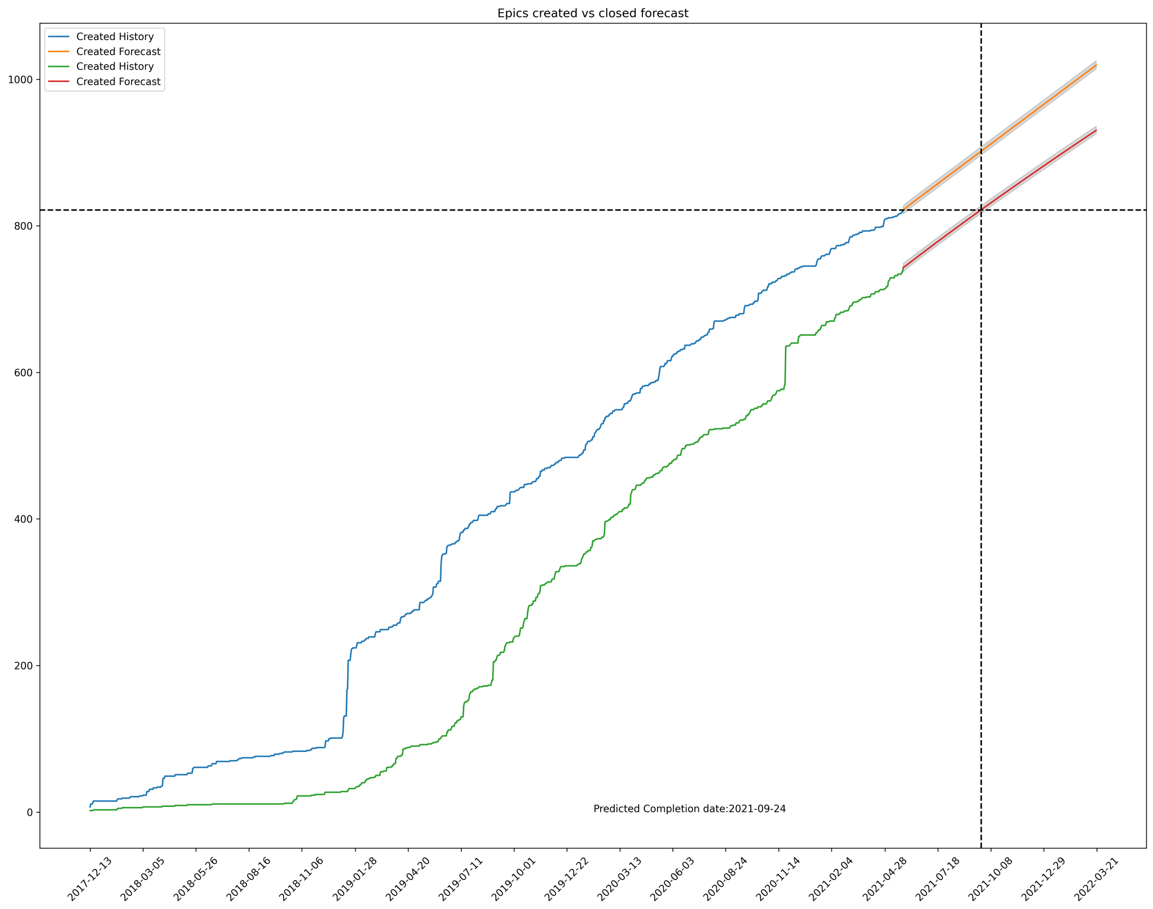
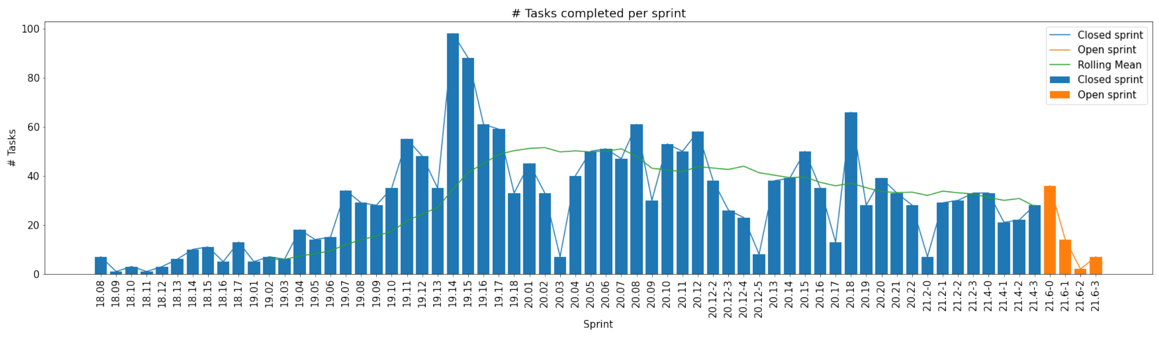


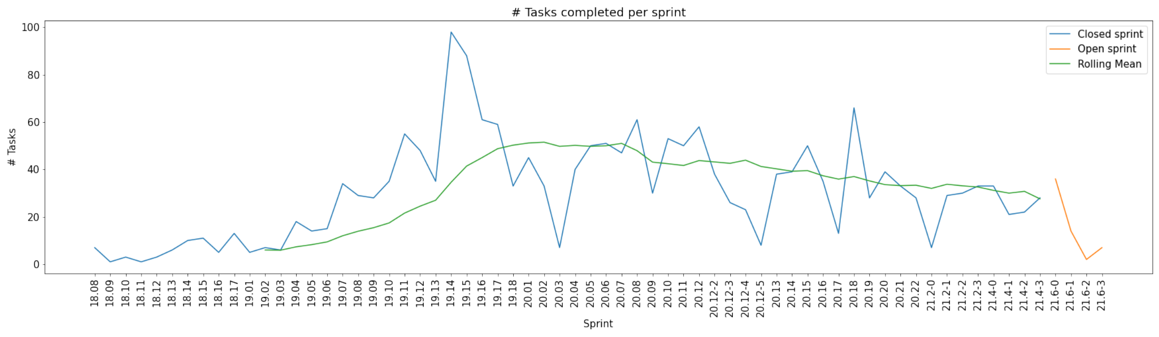
Figure 2.10: This is an example on a prediction of the *Flow Analyser* algorithm. It tries to predict when a specific project will finish, the predicted completion date is the 24 of September of 2021. We can say this prediction is extremely accurate, since enough time has passed to confirm our forecast [Own Compilation]

2.3.2 Monte Carlo Simulator

From the sprints and releases, we can make Monte Carlo simulations. Due to the fact that we cannot be completely sure which distribution or to which family of distribution our random variable follows, we can make use of a very common stochastic method called Monte Carlo simulations [37][18][6]; these simulations consist on sampling different values of the samples we have and make thousands of experiments until we get a reliable result. We don't need to know the distribution that random variables follow to understand their behavior. This type of simulation has been shown to have very good results in different areas such as reinforcement learning [47]. A typical scenario on how sprints evolve through time can be seen in figure 2.11, where you can see a little of seasonality which can be explained by workers vacations but the rolling mean is always the same which gives us an indication that tasks that are completed are represented by a stationary time series. The beginning of the plot can be ignored as it contains old data.



(a)

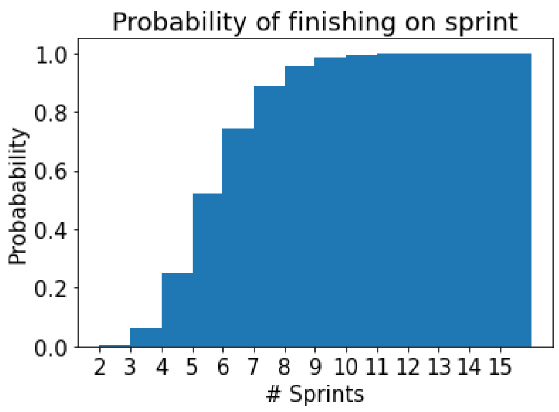


(b)

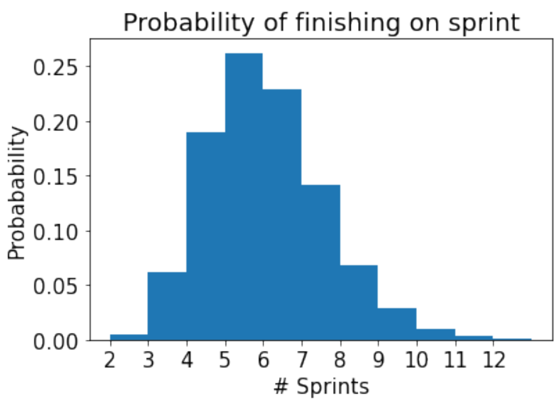
Figure 2.11: Shows the number of tasks completed in each sprint. Closed sprints (blue) are those that have already been completed and open sprints (orange) are those that have been planned for the future. The rolling mean (green) is always the same in the latest iterations. Both figures are the same with the only difference being the visualization [Own Compilation]

Based on how sprints evolve through time, our task is to find a probability distribution that explains how many tasks will be completed at a specific point in time. We could use the Poisson distribution to model this scenario since by definition it tries to explain the number of rare events in a fixed time length [6]. Although we preferred to generate Monte Carlo simulations because it would be difficult to find out the parameters of the distribution and we would need to model hundreds of different Poisson distributions (one for each team, project and program).

Based on the number of tasks we need to complete, how many sprints will it take us? So at each iteration of the algorithm we sampled a randomly closed sprint and used the tasks that number of tasks that were completed for our new tasks, we repeat this until we no longer have tasks left. All this process is then repeated thousands of times, the result is shown in figure 2.12 where we made a probability plot based on the number of occurrences, we can also see that probability of is normally distributed, which makes our forecasts more predictable.



(a)



(b)

Figure 2.12: Shows an example of MOnte Carlo simulations, figure (a) shows the cumulative distribution function (CDF) of finishing on at most that sprint, so in the end, the probability of finishing in at most 15 sprints is 100%, which gives use useful confidence interval. Figure (b) shows use of a probability density function (PDF), which gives us different information [Own Compilation]

Another forecast of this solution is that we can evaluate iteration plans; this means that if we represent an iteration or sprint as a tuple $I = (Stories, Bugs, Spikes)$ where each element is the number of tasks with their respective issue type, we can compare the iteration I_t at time t with other iterations at time $< t$. For each iteration in the past, we already know its completion ratio (that is the percentage of tasks that were completed in respect to the ones that were planned, figure 2.13) which gives us a measure of the quality of the plan, thus we can use this measure as a way of comparing it with the current sprint and evaluating how good or bad it is.

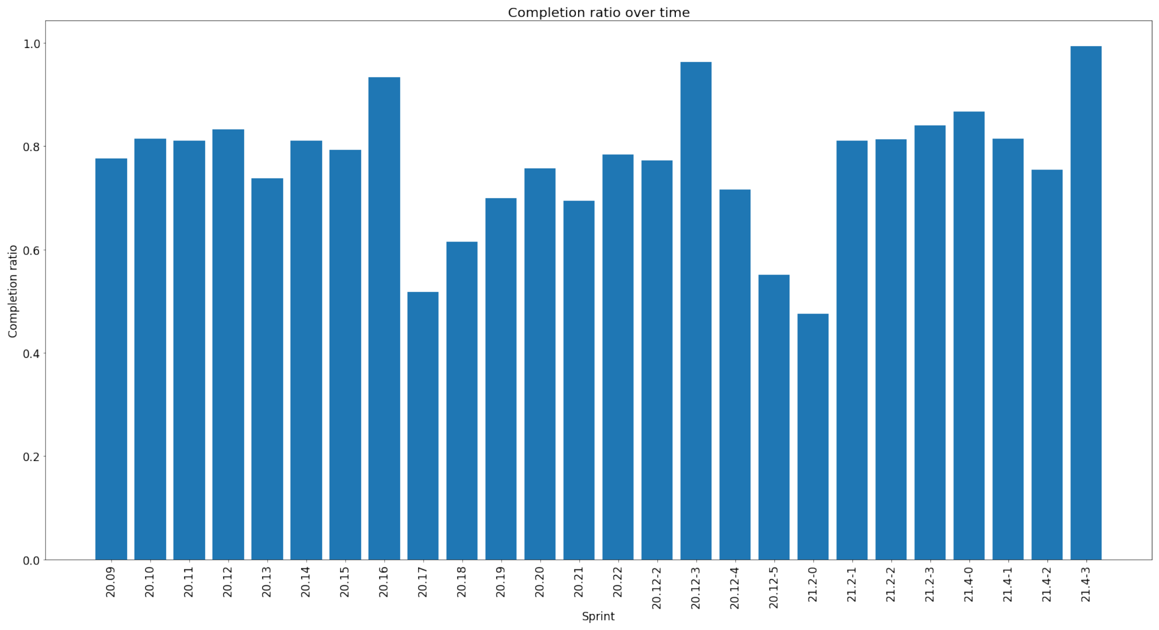


Figure 2.13: For every sprint we show the the completion ratio, a value between 0 and 1 which represents the percentage of tasks completed over the total planned [Own Compilation]

For this to work we analyzed the difference between sprints, and we identified three main clusters; we tried to show them inside boxplots in figure 2.14 but the separation between them is not clear enough compared to figure 2.15. We concluded that the clusters show the following things:

1. Sprints that do not contain bugs and contain few stories but a lot of spikes (blue, cluster 1). Points to sprints focused on research

2. Sprints that only contain stories and bugs of a similar magnitude (orange, cluster 2). Seems bug fixing + functionality developing

3. Sprints that only contain stories and spikes in similar magnitude (green, cluster 3). Functionality development + research

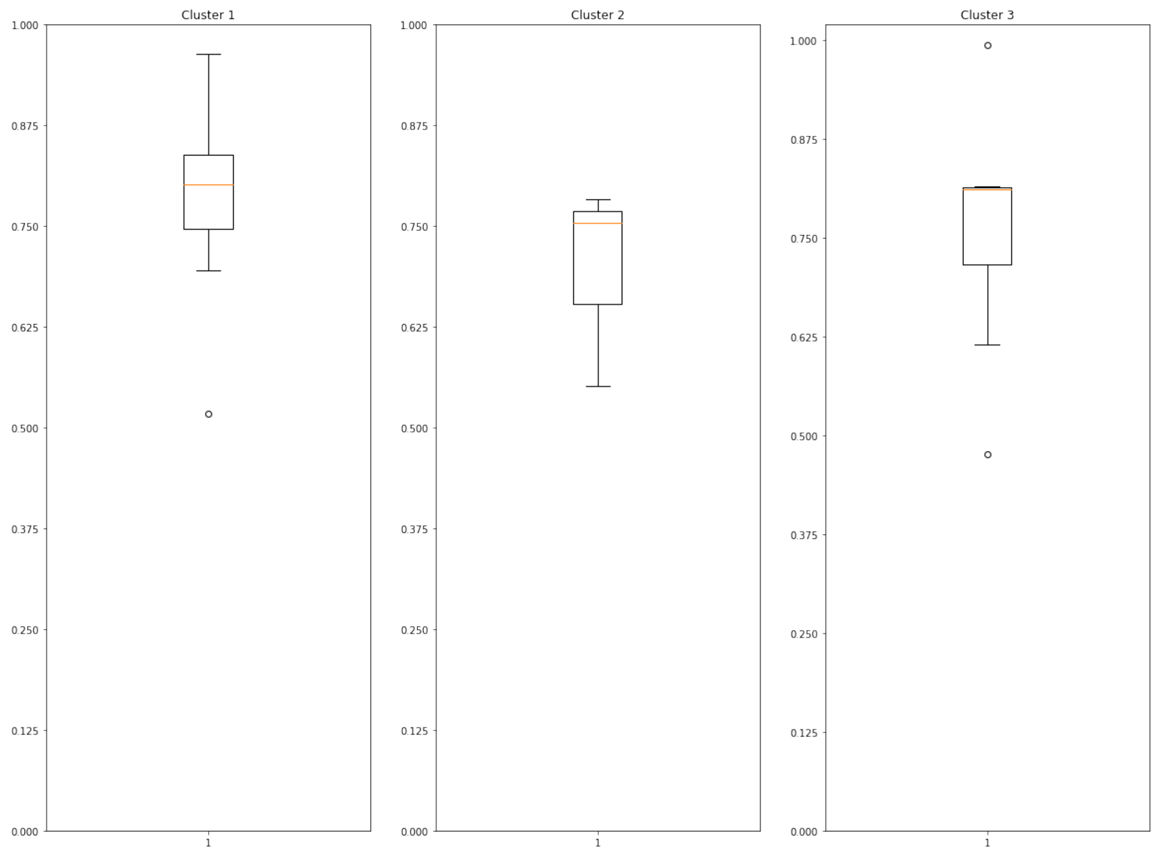


Figure 2.14: Shows three different boxplot, one for each cluster of iteration based on the number of tasks it contains. here are clearly some differences, specially on cluster 1 but too representative [Own Compilation]

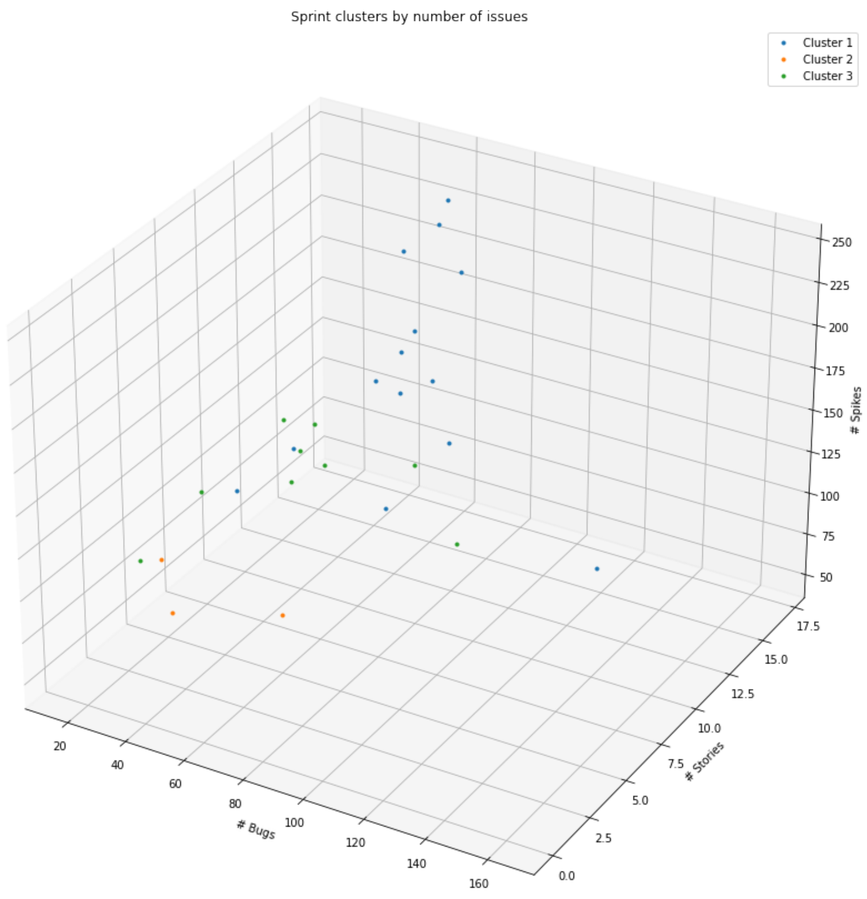


Figure 2.15: This 3-Dimensional plot shows how sprints are separated into different clusters based on the number of issues of each type, each cluster can be explained. More clusters did not seem to show any differences [Own Compilation]

The similarities between sprints were calculated in two different ways, the mean squared error and the cosine spatial distance, the MSE proved to show better results because it was not bounded this caused the cosine to dismiss the magnitude of the vectors which was extremely relevant. The idea is to calculate all the similarities and then select the 10-sprints most similar, this will give the user an idea of how great they plan their old and current sprint.

2.3.3 Effort Estimator

From open tasks estimate possible duration based on features and calculate overall duration. Applying machine learning algorithms to closed issues we can estimate the duration of unseen ones, once we have the duration we can apply more "accurate" simulations. The difference between the Monte Carlo Simulator and the Duration Estimator is that the latter does not take into account any random variable or its possible probability density function (PDF). Based on the different features a task might have, we want to estimate its possible duration, this duration will help us determine how much time it will take us to complete all the programs. The algorithm is a little complex so we have divided it into multiple steps:

1. **Preprocessing:** here we need to prepare the data for training, using machine learning models

with multiple features requires use to apply different techniques such as normalization and categorical encoding.

2. **Parameter tuning:** here we apply a grid-search to find the optimal parameters for a set of models. We can then use these parameters for the training and model comparison

3. **Training:** training is an important step in which we train multiple models cross-validation since we are comparing models and we want to reduce our bias we are using K-Fold cross-validation with $k = 10$, instead of the more basic hold-out cross-validation.

4. **Retrain** once we have the results we must elect the best model based on how much bias and variance it has, after that we retrain it with the whole dataset. An example on this results can be found on figure 2.17. We usually use the mean absolute error for visualization because it can be read more easily humanely but for model comparison and testing we use mean squared error since it is more robust to outliers.

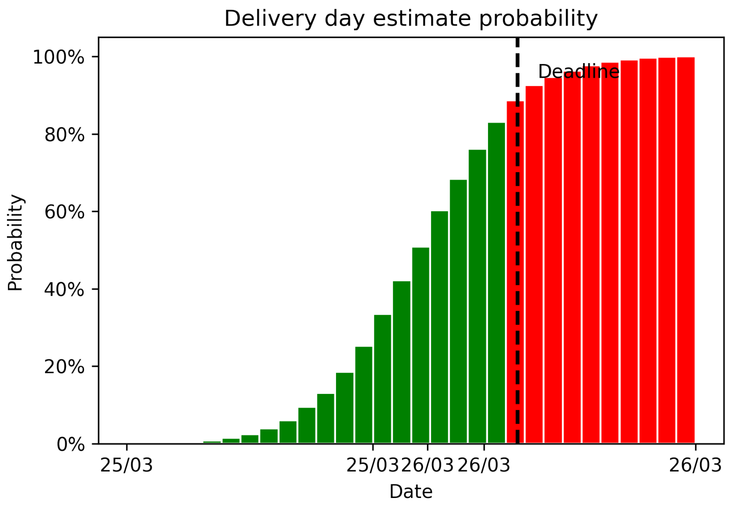


Figure 2.16: Visualization on how the predictions of the Effort Estimator work, for each date we have a probability, we marked the deadline with a dotted line. The area in green means are dates where we will finish on time and the area in red are dates that are over the deadline [Own Compilation]

5. **Predictions:** once we have the model, we select a set of issues and we make duration predictions on them.

6. **Calculate workers:** before we can go one we must also calculate the number of assignees or workers that might work in this particular set. This will help us determine how much people are working and how fast we can complete the tasks at hand.

7. **Monte Carlo simulations:** after that we make a Monte Carlo sampling of the issues so that they distribute randomly among the different employees. This will result in each assignee having a different set of issues, thus the duration is spread among the different workers and it is more realistic since you can not expect people to have always the same amount of work.

8. **PDF:** we then must run the simulation thousands of times to generate a probability density function on these simulations

9. **Probabilities:** finally, we transform the PDF into a cumulative density function to be able to estimate the probability more easily. These probabilities are displayed as in figure 2.16, where it can be easily seen how much time we have. This visualization is inspired by the ANDES algorithm [3].

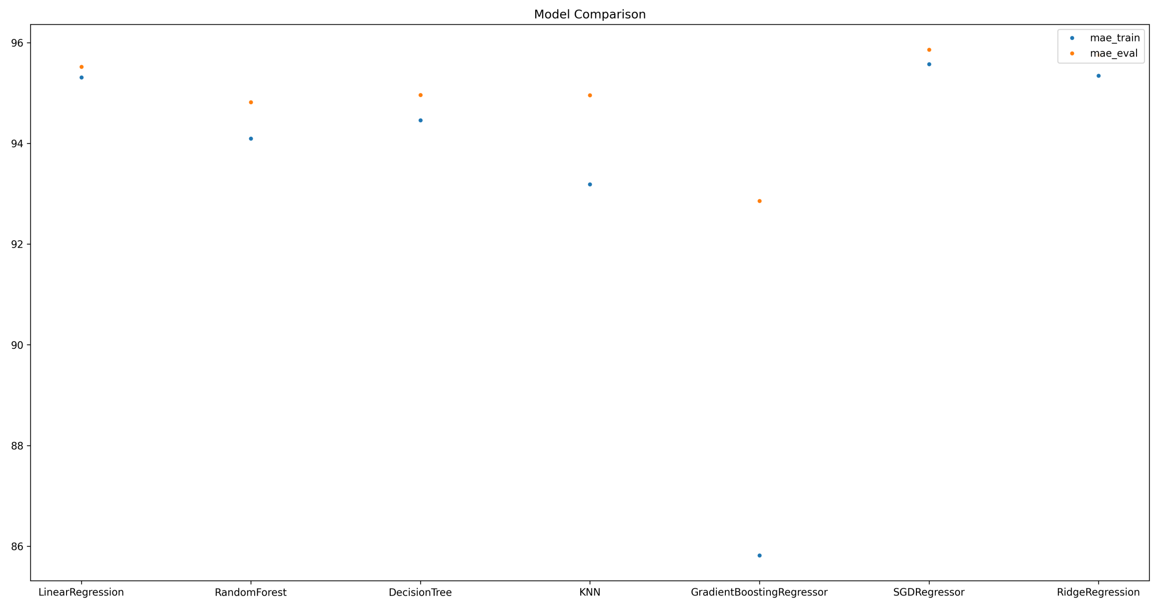


Figure 2.17: Shows the training and evaluation error on the duration prediction with different models, all models have a similar performance but we prefer models that are more stochastic and flexible, other models such as the Decision Tree are too rigid, non-parametric and thus always have the same list of results [Own Compilation]

2.4 Code

The code in the project is very complex because it has evolved a lot while we were working on it, thus we will not delve into all the specific details it has, we will only try to explain how is organized overall and some other details.

2.4.1 How are modules organized?

To organize the code we created different modules, each module contains multiple algorithms and scripts but they are all related to the same concept.

- **API:** the API module is in charge of connecting to Jira, downloading all the issues, making an initial cleaning and transforming them into a table.
- **Preprocessor:** the preprocessor contains further processing for specific algorithms, it contains text cleaning algorithms as well as preparation algorithms for ML models.
- **Core:** this is where most of the algorithms are in, here we have planning, model selection and data analytics algorithms.
- **Resources:** this is where we store all the resources as well as the Resource Manager (explained later on).

There are other scripts outside the modules but they are just entry points for the different algorithms. Their job is to connect everything. We also had many Jupyter Notebooks for data analytics but we cannot show them because they contain a lot of sensitive information, in fact, some information has been censored inside the code to be able to share it.

2.4.2 Resource Manager

As with every Data Science project, it is not a surprise that might get overwhelmed with experiments, after all, working with data means we need to test many alternatives and try lots of different parameters. Once we started having problems organizing the experiments we decided to implement a simple but very useful model called the **Resource Manager** (RM). The job of the RM was to manage every type of resource, the following are some examples

- **Logs:** every time we ran an experiment, especially when we wanted to analyze the data and train the algorithms we printed logs, but we also saved these logs to different files, this enabled us to reproduce old runs with the same parameters.
- **Data:** when we created a new experiment we usually tested the models with different data, but we don't want to replace old data, with our algorithm this can be easily done with a simple change of context.
- **Models:** every model we run in the past is tracked and saved in its own folder, saving models is very important because it is often difficult to reproduce old pieces of training.
- **Plots:** data analysis plots, as well as prediction plots, are also saved.
- **Predictions:** apart from the plots, we also save every prediction to a file, this enables us to recreate the plots differently with the same values.

The RM is in charge of saving and loading every resource, it was developed to solve two problems. First of all, it solved having to know the path to every resource and know every extension there is, finding where your resources are can be frustrating when you are coding and it is error-prone; with RM you only need to specify the type of resource and its name and it will find it for you. The second problem it solves is having to load the specific libraries or having to remember how to load a specific resource, with the RM, all resources are loaded the same way, this simplifies the code and reduces typos.

But how does it work? The Resource Manager contains a thing I like to call a context; a context is a directory where all the resources from a specific experiment are saved, the idea is that every time you run an experiment you tell the script which context you want to run in, and Resource Manager manages the rest. RM sorts all the resources in different sub-folders and organizes everything, so you don't have to worry about resources ever again. This algorithm has been extremely useful, especially when we wanted to reproduce old experiments, we only need to tell RM to get an old experiment context and all works perfectly.

Chapter 3

Conclusion and Discussion

Predictability is not easy, and many months have been spent on the development of this project. It has been both an exhausting and exciting experience, fortunately working at HP has been extremely rewarding. It also provided many areas of opportunity, such as being the only one with experience in Data Science, so I had to develop and research about the project all by myself. The only help I was provided with was domain knowledge on how the company works and how it is organized, but they did not have any technical experience as the people I had to work with specialized in designing firmware, that is another of the reasons why a project like this has never been developed, thus it was a very fruitful experience for both parties. I never thought I could learn so much or that I could even notice it, and I am grateful to have had this opportunity.

The duration of this project was marked by the number of stones we found in our path, since the beginning we had many problems that needed to be solved, which caused us to have a very diffuse path. Nevertheless, this caused us to find better, more innovative solutions. A thing that was a little frustrating was that every time we came up with a solution we always found new areas of improvement, in fact improving the algorithms and their usability was a daily topic, so we could have continued forever.

The algorithms we have shown have proven to have good results both in tests and in real life, which is more than we can expect. These algorithms will hopefully contribute immensely to each of the stakeholders, but just having good results is a great accomplishment. It is a must to continue improving our results in the future.

Appendix A

Planning

A.1 Tasks and Time estimation

Choosing a working methodology is very important, so as to learn more about it and to familiarize ourselves with how it works we choose an Agile methodology to track the project. We are going to use a Kanban methodology to track the progress of the work since Scrum is more team-centered and it relies on tracking techniques such as dailies that would prove useless in this project. We are going to use an iterative approach where after finishing a set of tasks we are going to review and analyze them.

Before continuing with the the planning I need to remark that the following is the initial planning, and in the end we explain the changes and deviations from the original plan have been done and why. The total estimated duration for the project is of 753 hours. The project can be divide into the following 7 blocks. Each block contains a list of tasks, subtasks, and even sub-sub tasks that correspond to a specific area of the project. For each block, we have a table with its tasks. Every task contains an **ID**, a short **name** or description, its estimated **duration** in hours, its **dependencies** and the most important **resources**. We need to note that a lot of dependencies are omitted because they can be inherited through the subtask system (i.e., if a task T1 is dependent on T2, then all subtasks T1.1, T1.2, ... T1.n and their children are dependent on T2).

Due to the number of tasks that are required to complete this project we can only describe them as a group, since describing each task individually would prove unnecessarily tedious, hopefully, the name of the task is descriptive enough.

A.1.1 Setting dates

To get a context on the period of time we are working, we must define some dates.

- **Start date:** 4 of November
- **Expected release:** 4 August
- **Deadline:** 4 October

All the work will be done on workdays, with a total of 20 hours a week. Due to the Covid-19 pandemic, we will work from home.

A.1.2 Resources

Considering the amount of work we need to do, we must use of various resources, some of the most important are listed as follow.

- **Overleaf:** LaTeX text editor required for writing the documentation.
- **Zoom:** Video conferencing software used on all HP meetings.
- **Jira:** Agile planning and tracking tool which contains all HP projects and their development.
- **Jira Python API:** Library required to access the information stored at Jira.
- **Python:** We will use this programming language for its ease of use, its familiarity, and because it is the only language that is supported by the JIRA API.
- **Evernote:** Note writing app used to keep track of daily progress and as a journal.
- **Todoist:** Task manager application used to organized the project tasks, meetings, notes, and everything related to my work at HP.
- **Powerpoint:** Presentation software used to make different presentations and proposals on the project.
- **Word:** Text editor used to track our weekly progress.
- **Python libraries:** This includes any python libraries used to train machine learning models, analyze the data, and track to project (i.e., Keras, scikit-learn, pandas)
- **Code editor:** As a code editing tool we have used both Jupyter notebook and Atom.
- **Wandb:** Developer tools for the tracking of machine learning projects.

A.1.3 Documentation and Project Management

Managing the project and writing its documentation is as important as the development of the project itself, that is why we have grouped all tasks related to tracking and planning the project in this block. Besides writing the documentation, we have developed a tracking system similar to scrum where we record what we do each day, this way we don't miss anything; apart from that we also do a weekly review where we evaluate our current overall progress and watch an eye for any problems that might occur. Finally, since we are working telematically with HP, we have daily meetings, where we discuss the project and review how well it goes. All of these tasks as well as their attributes can be seen in table A.1.

ID	Name	Duration (h)	Dependencies
D	**Documentation and Project Management**	**255**	
D1	Context and Scope	15	
D2	Planning	20	D1
D3	Budget and Sustainability	15	D2
D4	Final documentation review	8	D3
D5	Project development documentation	35	D4
D6	Meetings	115	
D7	Daily development tracking	20	
D8	Weekly development tracking	12	
D9	Code documentation	15	

Table A.1: Task list for documentation and project Management [Own Compilation]

A.1.4 Project Review

This block is very similar to documentation tracking, in fact, they are extremely related (see A.2), the difference is that Project Review tasks do not concern with progress, or timings but with performance metrics. The main purpose of this group of tasks is to evaluate the performance and error of the prediction models we are using. Other tasks include the analysis of the problems and solution proposals.

ID	Name	Duration (h)	Dependencies
PR	**Project Review**	**81**	
PR1	Problem Analysis	8	
PR2	Solution Requirements	4	PR1
PR3	Solutions Proposals	8	PR1
PR4	Results Review	61	IP, FW, TS
PR4.1	Basic model and features review	32	
PR4.2	Natural Language Model review	12	
PR4.3	Future Work review	12	
PR4.4	Final project review	5	

Table A.2: Task list for project review [Own Compilation]

A.1.5 Issue duration prediction

This is one of the fundamental task groups of the project. As the name indicates, these tasks are related to the prediction of the duration of each issue; every task from data analysis to model training and evaluation is included here. Tasks that are included here might overlap or at least be useful for other parts of the project, that is because connecting to *Jira* and familiarizing with the data we are working with is an implicit task that is involved in this group and that is related to each of the other groups. The objective of completing this set of tasks is to have an algorithm that can make smart predictions on the duration of individual tasks; the duration of an individual task is not useful by itself, but the duration of a set of tasks can give us a good approximation on the release date of a product.

ID	Name	Duration (h)	Dependencies
IP	**Issue Duration Prediction**	**194**	
IP1	Data Analysis	58	
IP1.1	Connect to JIRA API	2	
IP1.2	Fetch data from API	4	IP1.1
IP1.3	Initial data cleaning	4	IP1.2
IP1.4	Feature review and documentation	4	IP1.3
IP1.5	Analyse feature distributions	8	IP1.3
IP1.6	Study Natural Language features	30	IP1.3
IP1.7	Find correlations between features	4	IP1.3
IP1.8	Outlier detection	2	IP1.3
IP2	Data Preprocessing	54	IP1
IP2.1	Imputing missing values	4	
IP2.2	Data standarization	1	
IP2.3	Categorical and numerical features cleaning	5	
IP2.4	Natural Language cleaning	15	
IP2.5	Natural Language tokenization	15	
IP2.6	Natural Language parsing	4	
IP2.7	Natural Language word embedding	6	
IP2.8	Duration estimation	4	
IP3	Model Training	71	IP2
IP3.1	Model training for basic features	7	
IP3.2	Hyper-parameter tuning for basic models	1	IP3.1
IP3.3	Connect to wandb to track the experiments	3	
IP3.4	Model training neural networks for NLP	20	IP3.3
IP3.5	Hyper-parameter tuning neural networks	35	IP3.4
IP3.6	Research and apply possible transfer learning networks	5	IP3.5
IP4	Testing and Validation	11	IP3
IP4.1	Generate different comparison plots	6	IP4.2
IP4.2	Create a baseline model for performance measurement	1	
IP4.3	Test models on current issues	2	IP4.1
IP4.4	Write conclusions	2	IP4.3

Table A.3: Tasks necessary for the issue duration prediction [Own Compilation]

A.1.6 Task Scheduler

This group of tasks contains an essential part of the forecast algorithm. Once we have an accurate prediction of each tasks' duration, we need to introduce the concept of task concurrency. This notion indicates that tasks in a project are not done by a person, multiple developers in different teams work and interact on the tasks simultaneously. That is why we need a Task Scheduler or Task Dispatcher, its objective would be to make assignations of the tasks to the different people. For this, to work we must first query the workers that are on the current project and then create thousands of possible assignations.

ID	Name	Duration (h)	Dependencies
TS	**Task Scheduler**	**31**	**IP**
TS1	Research and brainstorming	4	
TS2	Development	27	TS1
TS2.1	Assignee Search	8	
TS2.2	Random Task Generator	12	
TS2.3	Connect to prediction model	1	TS2.2
TS2.4	Measure error on true labels	4	TS2.3
TS2.5	Test on current issues	2	TS2.4

Table A.4: Tasks list of the task scheduler [Own Compilation]

A.1.7 Future Work Forecast

Future Work Forecast encompasses tasks related to the prediction of future events, such as, how will the project evolve in the future? Will we have more tasks or less? Based on this estimation we can have an idea of the effort that will be required in the future to complete them.

ID	Name	Duration (h)	Dependencies
FW	**Future Work Forecast**	**97**	**IP, TS**
FW1	Data Analysis	21	
FW1.1	Research candicate projects for forecast	5	
FW1.2	Review past projects and their tasks	10	FW1.1
FW1.3	Research relationship between projects	6	FW1.1
FW2	Data Preprocessing	45	FW1
FW2.1	Fetch projects data	15	
FW2.2	Prepare and parse data	15	FW2.1
FW2.3	Research possible data augmentation	3	FW2.1
FW2.4	Research generative models	12	
FW3	Model Training	23	FW2
FW3.1	Train basic heuristic approach	15	
FW3.2	Train different machine learning models	8	
FW4	Testing and Validation	8	FW3
FW4.1	Validate model through mangement experts	3	
FW4.2	Validate model through error metrics	3	
FW4.3	Test model on current products	2	

Table A.5: Task list for the prediction of the future work [Own Compilation]

A.1.8 User interface

Creating the forecaster is not enough, we also need a way for people to interact with it (and a simple CLI does not do the trick). We need to create a user interface where people can connect easily, get results, and interact with them. This is a complicated task because we have dependencies with the people that will use the application. Users must test the application and give feedback if necessary. One advantage of this group of tasks is that is not strongly dependent on other tasks so it can be developed concurrently.

ID	Name	Duration (h)	Dependencies
UI	**Prediction User Interface**	**95**	**IP**
UI1	Design	27	
UI1.1	Design basic functionalities	12	
UI1.2	Design UI layout	15	
UI2	Implementation	53	UI1
UI2.1	Implement basic functionalities	25	
UI2.2	Organize and apply layout	20	
UI2.3	Beautify interface	8	
UI3	Deploy	15	FW, UI2

Table A.6: Tasks related to the user interface [Own Compilation]

A.1.9 Gantt Overview

The project we are working on involves the interaction of multiple tasks connected through multiple dependencies. Having a large number of tasks does not help to represent them, thus for the completion of the project, we have created a Gantt Chart (see A.1) that represents a broad overview of how tasks are related and how they unroll through time. Unfortunately the tasks that are represented last days or weeks, we could not include the smallest tasks because they wouldn't provide any useful information and they would only make the visualization of the tasks worse.

Each row in the Gantt chart represents a task, and each group is assigned a color for readability. We've also created a simplified graph A.2 where the big groups can be visualized without all the subtasks clutter. Each square of the Gantt chart represents a week in the Gantt space, we are simplifying every month to 4 weeks too for simplicity.

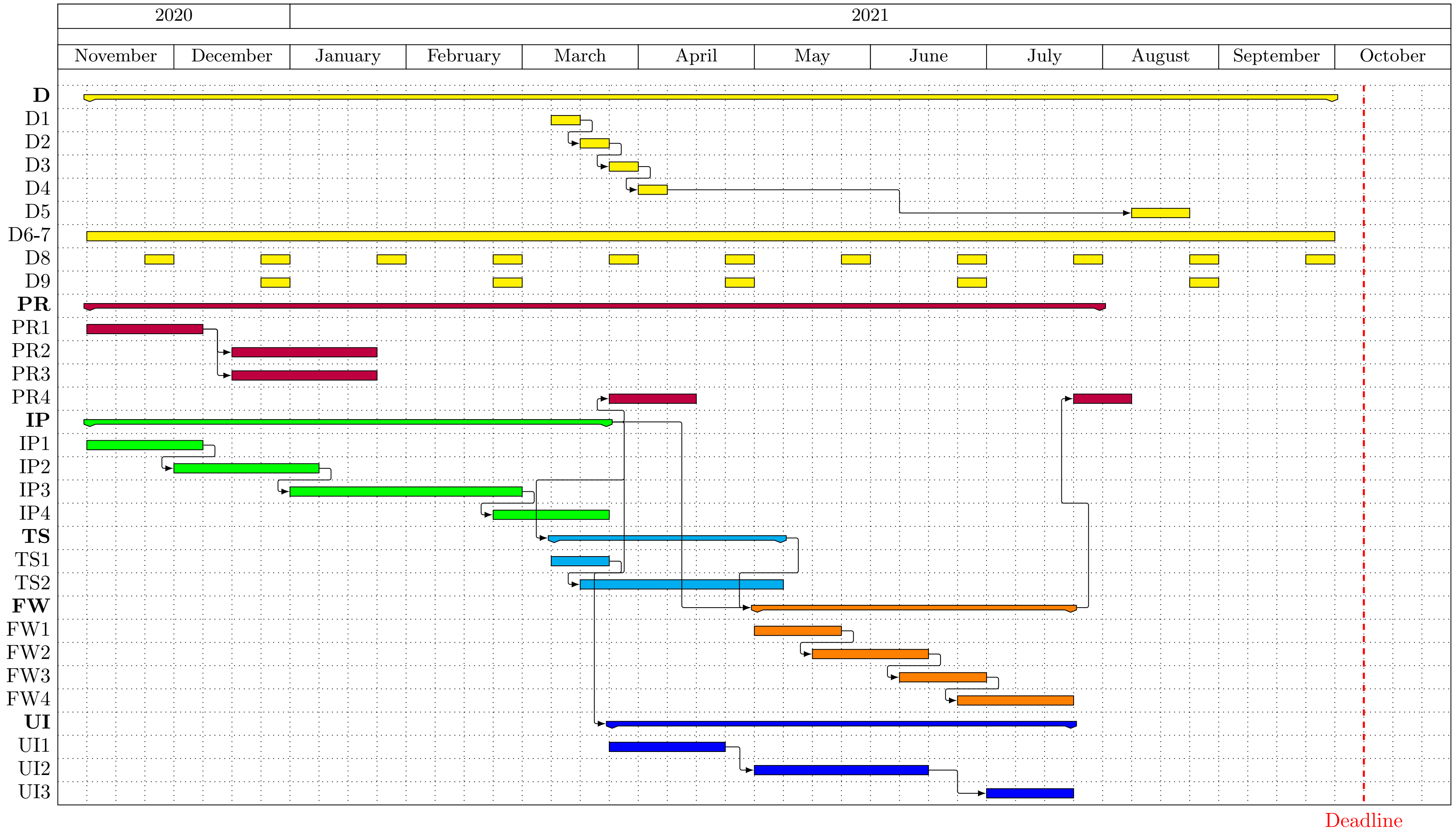


Figure A.1: Gantt Chart [Own Compilation]

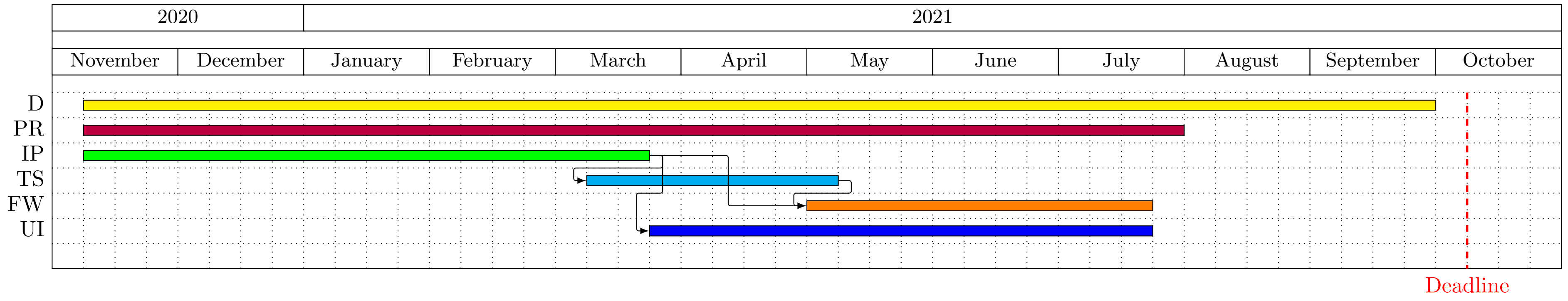


Figure A.2: Gantt Chart, simplified version [Own Compilation]

A.2 Risk management

During the development of this project we might encounter multiple obstacles, we cannot expect to know or prevent all of them, because we don't know the future, but we can list the most probable and impactful risks.

A.2.1 Deviation on original plan

This is possibly one of the most concerning risks that we might have, a change in the original plan can become a very big stone in the path. This change could happen if the current solution does not work as expected or for some reason we require a better one throughout the project.

- **Impact:** This has a high impact on the project because it can cause many problems since we might need to redo a lot of tasks.
- **Possible Solution:** We don't have a solution for this problem, the only thing we can do is apply preventive techniques. Improving the planning and thinking ahead can help us see into the future and prevent any changes. This is one of the reasons we track our progress daily.

A.2.2 Inexperience in project

One of the issues that might arise while developing the project is that I am young and don't have too much experience, this issue might cause problems with the project and we could get stuck in a specific task

- **Impact:** This has a medium impact because getting stuck means that we are spending more time on a task than necessary.
- **Possible Solution:** Even though this is a difficult problem, there are multiple solutions. We could spend more time learning and trying to keep developing the task or we could ask for help both at the University and inside HP, they both provide helpful resources and people that could give me a hand.

A.2.3 Computational Resources

Training machine learning models are known to be computationally expensive, this can cause two problems. It can make tasks last longer and it can cause a delay in tasks (e.g., if we don't have enough RAM, there is no way we can continue working).

- **Impact:** This has a low impact because it is possible but it can be fixed easily.
- **Possible Solution:** Some solutions to this problem might be to optimize the models to reduce resource consumption. Another approach would be to find an external computational resource (i.e., Cloud Computing).

A.2.4 Project deadline

Another problem we might have is that we might reach the deadline of the project before completing it. This might happen either because we underestimated the duration and quantity of the tasks we need to do, or because we encounter one of the other risks.

- **Impact:** The impact of this risk is low because based on the current planning, we have a lot of free time before the deadline.
- **Possible Solution:** Fixing this issue requires fixing other risks we might encounter, we also should adapt the planning of the project as we progress, this why we can have an updated forecast on the completion date.

A.3 Changes from the original plan

Here we will state which changes we had from the original plan. First of all we need to add additional resources

- **PowerBI:** an additional tool we used was PowerBI, this tool enabled us to visualize and share interactive plots with everyone in the company.
- **Microsoft Teams:** as an additional tool for planning and organization we also used Microsoft teams, mainly because it contains many plugins which proved useful.

Based on the tasks that were done, there are many similarities, the differences between the original plan and the actual methodology were more specific. What I mean by this is that we did the same amount of tasks with similar duration but with the difference being the name of the tasks and how they were organized. The three main tasks (**IP**, **TS** and **FW**) made their own contribution to the final three algorithms that we described in this paper. There are only some minor implementation differences. Moreover the User Interface task was simplified thanks to the use of PowerBI, we still needed to design the interface and beautify it, but there was no coding involved. Overall the original plan has been followed very well.

Regarding the final deadline, it has been delayed by about a month, so we finished in September instead of August as it was planned. But this was not a problem since we already have enough time left to deliver it.

Appendix B

Budget

The budget defined in the following sections is defined based on the original planning, the project has suffered a slight deviation but overall the impact to the costs might have been negligible.

B.1 Staff Costs

This is a very rich project, rich in the way that it encompasses a lot of areas, from machine learning to interface design. That is why there are multiple roles involved. In a more realistic environment, we would have these roles within separate people but here, we will assume that person can work in multiple roles. A summary of this relationship can be found on table B.1.

Role	Description	Person
Project Manager	Planning and verifies the correct development of the project	Pol, GEP, Tutor, Supervisor
Data Analyst	Parsing and analysis of all the data, creates statistics and conclusions	Pol
AI Engineer	Works with the training of machine learning models and their validation	Pol
UX Designer	Designs the user interface and its functionalities	Pol
UX Developer	Applies the interface designs and connect the app to the prediction model	Pol
Technical Writer	Writes all technical documents, presentations and documentations	Pol

Table B.1: Description and attributes of each role working in the project

Role	Tasks	Hours	Salary (€/hour)	Total Cost (€)
Project Manager	D6, PR	197	24 [46]	4728
Data Analyst	IP1, IP2, FW1, FW2, D6	293	18 [5]	3204
AI Engineer	IP3, IP4, TS, FW3, FW4, D6	260	21 [29]	3045
UX Designer	UI1, D6	142	18 [51]	486
UX Developer	UI2, UI3, D6	183	17 [52]	1156
Technical Writer	D1-D5, D6	208	19 [48]	1767
Total				25,081

Table B.2: Contains the salaries and total cost of maintaining the workers, it has been estimated based on the yearly salary and the total working hours at HP.

B.2 Generic costs

Apart from the human resources, we must also take into account software or hardware that is used in the development of the project. Software resources don't account for any cost since most are open sources and others are not related to our project (i.e., Jira cost is amortized by the developers that use it, I just happen to use the data they produce). Hardware resources can only encompass the use of the computer HP Elitebook, to stick to our previous scenario we will suppose we need to buy a computer for each employee. The computer has an approximate price of 1508.87€ [19]. Based on equation B.1 we can calculate the total amortization of the computer and their cost based on their usage hours. We have summarized this costs on table B.3. Note that the residual value is 0 because the usage of this computer can be amortized in the future by another HP employee.

$$\text{Ammortization} = \frac{\text{Initial value} - \text{Residual value}}{\text{Life expectancy}} \tag{B.1}$$

Employee	Hours	Life expectancy (years)	Ammortization	Cost (€)
Project Manager	197	8	188.60875	21.30
Data Analyst	293	8	188.60875	31.61
AI Engineer	260	8	188.60875	28.07
UX Designer	142	8	188.60875	15.30
UX Developer	183	8	188.60875	19.72
Technical Writer	208	8	188.60875	22.42
Total				138.42

Table B.3: This table summarizes the amortization and usage cost of each computer based on the employee's work hours. The cost is calculated by multiplying the amortization by the number of hours worked (assuming a year of work is equivalent to 1744 hours)

Other costs such as electricity, internet, and workspace are not accounted for because we are working inside a big company that will need to pay these expenses independently on the project, and extra space and electricity that might be used is negligible compared to what is paid for.

B.3 Total costs

As we have talked about before, there are some risks involved in this project, every risk has is set of problems and each can cause an economic impact. To reduce the impact of these unexpected events we will need to prepare contingency budgets based on their risks on the project. We will save 10% of the budget for staff and generic costs, this will help us prevent any change that might happen. Moreover, we will also reserve part some budget for the risks that might happen, this is summarized in table B.4, in this table we estimate how much would it cost us for a risk to happen and how will we reserve based on its probability.

Incident	Estimated cost (€)	Risk (%)	Cost (€)
Deviation on original plan	5,000	15	750
Inexperience in the project	500	40	200
Computational resources	500	25	200
Deadline	1,000	5	50
Total	7,000	-	1,200

Table B.4: This table summarizes the cost of each risk and the probability of happening.

Finally, based on the sum of all costs risks, and contingencies we can calculate the expected cost of the project (see B.5).

Subject	Cost (€)
Staff	25,081.00
Generic	138.42
Contingency	2,521.94
Possible incidences	1,200.00
Total	28,941.36

Table B.5: This table summarizes the total costs of the project.

Appendix C

Sustainability

C.1 Self-assessment

Sustainability is a very simple term, but it is often misunderstood and most of the time is not applied correctly. Social, economical, and environmental aspects are always implied in every project we work on, the problem is people don't usually think about them and don't try to change them, usually, the economic area is the one that has the most impact; this is usually caused by ignorance on the people that work with the project of because it is difficult to create e project that excels at three dimensions at the same time, that is because more often than not, there exists a trade-off between them. I must admit that, before the start of this project I haven't given much thought to the sustainability of the project, it was when I started working in HP where I encountered that their way of working the company culture is formed on different core values, one of which encompassed sustainability. From that point on, I realized that working in a conscious environment is as important as applying the concepts to the project. The poll we had to take is very important for our sustainability, getting conscious and inform about things is the first step to improve and navigate to a better future.

C.2 Economic dimension

Regarding PPP: Reflection on the cost you have estimated for the completion of the project

In chapter B we can find a comprehensive report on the economical costs of the project, it includes the human and material resources, as well as indirect costs and incidental costs. Based on that report we can assess that the cost of the project is very high, but the total costs of the product are completely justified because the main reason this project was created was to improve the productivity and predictability of products, this improvement will save us money in the long run.

Regarding Useful Life: How are currently solved economic issues (costs...) related to the problem that you want to address (state of the art)? How will your solution improve economic issues (costs ...) with respect to other existing solutions?

The problem that we want to address is highly related to our economy, in fact, it is one of the motivations behind it. The motivation for the project is to improve productivity and quality on projects, so indirectly we are making better products to the end-user, thus selling more. One of the problems with the product is that it would require multiple workers, each specialized in a different area, maybe we could reduce the economic impact by creating better planning, focused on optimizing the work hours and reducing meeting times since they represent a high percentage of the costs. Another solution would be to externalize the project to another company, depending on the case, this could provide us with a pretty good deal.

C.3 Enviromental dimension

Regarding PPP: Have you estimated the environmental impact of the project?

I have not estimated the environmental impact of the project but its impact is relatively low because we don't use material resources, the highest impact is in the use of electricity, mainly on the training of machine learning models which might require days or weeks of computer power.

Regarding PPP: Did you plan to minimize its impact, for example, by reusing resources?

We did not plan a way to minimize the impact because we don't have any material waste. We cannot reuse electrical resources because they are completely wasted (unless we wanted to create a system that converts the computer heat to electricity, but it is too far-fetched).

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)? How will your solution improve the environment with respect to other existing solutions?

This does not apply to our situation, since our environmental impact cannot be reduced without a drawback in the model performance. For example, if we reduce the size of the training set, or we make a more simple prediction model, the training time would be reduced, as a result, the use of electricity too. But this reduction would cost lost a lot of performance, which would not be helpful in the long run.

C.4 Social dimension

PPP: What do you think you will achieve -in terms of personal growth- from doing this project?

This project will help me grow in a lot of aspects. First of all, I have learned first hand, what is to work in a big company, this is an enormous experience that I will never forget. An advantage of working at HP is that they are motivated by their culture, this culture has a great impact on all three of the sustainability dimensions. Finally, I also learned a lot of things in terms of planning and time organization, I had never worked on a project this big, this has caused me a lot of frustrations and obstacles but it has also helped me learn.

Regarding Useful Life: How is currently solved the problem that you want to address (state of the art)?, and ... How will your solution improve the quality of life (social dimension) with respect to other existing solutions?

The problem that we are solving will increase the predictability of teams and products, this will be of a lot of help to everyone involved in them. Every developer will have higher productivity and project managers will be more confident on what can and cannot be released in time. As stated in the introduction, this solution has emerged from the need to improve old and unreliable solutions.

Regarding Useful Life: Is there a real need for the project?

As explained multiple times, this project emerged from a need for teams to improve their predictability in agile projects, so yes, there is a real need. In fact, this project will be of constant use for both section managers, project managers, and developers once it is completely functional.

C.5 Sustainability Matrix

In this final section, we are just simplifying the sustainability report into a matrix, so that it can be easily visualized. Each ranking is directly related to the answers on the other sections in this chapter.

	Environmental	Economical	Social
PPP	10	8	10
Useful Life	10	10	10
Risks	10	9	7

Table C.1: Sustainability Matrix rated from 0 to 10, where highest means we have achieved a better sustainability

Appendix D

Technical competences

- ***CCO1.1:*** *To evaluate the computational complexity of a problem, know the algorithmic strategies which can solve it and recommend, develop and implement the solution which guarantees the best performance according to the established requirements. [Enough]*

  We were not restricted to high-performance requirements because we did not create a user application, nevertheless, it was very important to develop fast algorithms because slow algorithms could make us waste a lot of time experimenting. If algorithms are slow then our experiments are slow, so we have very highly motivated to create our own performance requirements. Time complexities were too complex to analyze theoretically but it was very important to be careful with space complexities since we have had multiple *Out of Memory* issues. We need to be very careful with these so we created special algorithms, like a paginated data fetching query.

- ***CCO1.3:*** *To define, evaluate and select platforms to develop and produce hardware and software for developing computer applications and services of different complexities. [In depth]*

  Selecting the platforms to work with was somewhat difficult because there are a lot of platforms. For instance, we were not sure if we would use Julia or Python as a Data Science language. Designing the graphical interface was even more complex because there were many options and each of them had many problems. Most of the time we tended to select industry-standard platforms and libraries.

- ***CCO2.2:*** *Capacity to acquire, obtain, formalize and represent human knowledge in a computable way to solve problems through a computer system in any applicable field, in particular in the fields related to computation, perception and operation in intelligent environments. [Enough]*

  Many of the information that we used had very specific meanings that could only be understood with human knowledge, so we created a system that interprets this data and transforms it into something the algorithms understand. Some of the ways we tried to transform the information could not be used, showing that the data we are using has a very complex value and can only be understood by very specialized people. We managed to create systems that not only understood the data but that also made inferences with it.

- ***CCO2.3:*** *To develop and evaluate interactive systems and systems that show complex information, and its application to solve person-computer interaction problems. [In depth]*

  Our final graphical interface was made of a PowerBI dashboard. PowerBI enabled us to define highly interactive reports where a team could visualize their own personalized charts, make personalized filters, and even get a custom web URL. In the beginning, all of our plots were generated using Python but these were not interactive at all and could only be shared using images, this is the reason we needed to develop something more complex.

- ***CCO2.4:*** *To demonstrate knowledge and develop techniques about computational learning; to design and implement applications and systems that use them, including those dedicated to the automatic extraction of information and knowledge from large data volumes. [In depth]*

  This was one of the competences we needed to focus on the most. We needed multiple ways to extract and download large data volumes. First of all, we needed to download all the information, we also needed to clean it and extract everything important and discard anything useless. It was very important that this process could be automated because it needed to be executed daily in a virtual machine (new information is created daily).

- ***CCO2.5:*** *To implement information retrieval software. [In depth]*

  Most of our work here was to design information retrieval algorithms. The software we designed can retrieve information and adapt to future changes to the data, this step was critical because the data is always changing.

- ***CCO2.6:*** *To design and implement graphic, virtual reality, augmented reality and video-games applications. [A little bit]*

  To be able to visualize the final result and the predictions we needed to create and design a graphical interface. It does not consist of a virtual reality environment, nor a video game but we spent many hours trying to convey the information the best way possible.

- ***CCO3.1:*** *To implement critical code following criteria like execution time, efficiency and security. [Enough]*

  Execution time was of most importance because we are working with large data sets. We developed hardware-specific algorithms that improved their performance and we also tried to minimize the amount of processing required to do with the data. Moreover, HP has high security and privacy standards so we were always careful not to leak any personal or internal information.

- ***CCO3.2:*** *To program taking into account the hardware architecture, using assembly language as well as high-level programming languages. [A little bit]*

  During the project we worked with large amounts of data and corpora, this clearly caused a high computational overhead, so we needed to design algorithms that could adapt to our hardware requirements. Some machine learning algorithms like neural networks required us to use GPUs which provided accelerated training. We also tried to create multiple threads in order to speed up the data preprocessing and fetching. Other algorithms could be trained in parallel too. We didn't work with low-level languages but we did take into account hardware requirements.

Bibliography

[1] *Agile Alliance Velocity.* URL: https://www.agilealliance.org/glossary/velocity/.

[2] *Agile Predictability.* URL: https://www.agilealliance.org/resources/experience-reports/agile-predictability/.

[3] Evelyn Duesterwald Et Al. *Predicting the Likelihood of On-Time Delivery of Agile Projects Using ANDES.* Tech. rep. IBM, 2013.

[4] *Atlassian Scrum Guide.* URL: https://www.atlassian.com/agile/scrum#:~:text=Scrum%20is%20a%20framework%20that,and%20losses%20to%20continuously%20improve..

[5] *Average Data Analyst Salary in Spain.* URL: https://www.payscale.com/research/ES/Job=Data_Analyst/Salary.

[6] Michael Baron. *Probability and Statistics for Computer Scientists.* A Chapman & Hall Book, 2019. ISBN: 978-1-138-04448-7.

[7] Kent Beck Mike Beedle Arie van Bennekum Alistair Cockburn Ward Cunningham Martin Fowler James Grenning Jim Highsmith Andrew Hunt Ron Jeffries Jon Kern Brian Marick Robert C. Martin Steve Mellor Ken Schwaber Jeff Sutherland Dave Thomas. *Agile Manifesto.* URL: https://agilemanifesto.org/.

[8] Piotr Bojanowski et al. "Enriching Word Vectors with Subword Information". In: *CoRR* abs/1607.04606 (2016). arXiv: 1607.04606. URL: http://arxiv.org/abs/1607.04606.

[9] *Box Plot.* URL: https://en.wikipedia.org/wiki/Box_plot.

[10] Peter Bruce, Andrew Bruce, and Peter Gedeck. *Practical statistics for data scientists: 50+ essential concepts using R and Python.* O'Reilly Media, 2020.

[11] *Cron Job Man.* URL: https://man7.org/linux/man-pages/man8/cron.8.html.

[12] *Cummulative flow diagram.* URL: https://en.wikipedia.org/wiki/Cumulative_flow_diagram.

[13] Houtao Deng and George Runger. "Feature selection via regularized trees". In: *The 2012 International Joint Conference on Neural Networks (IJCNN).* IEEE. 2012, pp. 1–8.

[14] *Engineering Estimates.* URL: https://producthabits.com/engineering-estimates/.

[15] *Exploratory Data Analysis.* URL: https://en.wikipedia.org/wiki/Exploratory_data_analysis.

[16] Andrea Fryrear. *Benefits of Agile Marketing.* URL: https://www.agilesherpas.com/blog/benefits-of-agile-marketing.

[17] Aurélien Géron. *Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow: Concepts, tools, and techniques to build intelligent systems.* O'Reilly Media, 2019.

[18] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep learning.* MIT press, 2016.

[19] *HP Elitebook Price.* URL: https://bit.ly/2OpUo5t.

[20] Rob J Hyndman and George Athanasopoulos. *Forecasting: principles and practice.* OTexts, 2018.

[21] *It's Difficult to Make Predictions, Especially About the Future.* URL: https://quoteinvestigator.com/2013/10/20/no-predict/.

[22] *Jenkins.* URL: https://www.jenkins.io/.

[23] *Jira API.* URL: https://jira.readthedocs.io/en/master/.

[24] *Jira JQL.* URL: https://www.atlassian.com/blog/jira-software/jql-the-most-flexible-way-to-search-jira-14#:~:text=JQL\%20stands\%20for\%20Jira\%20Query,for\%20your\%20issues\%20in\%20Jira.&text=This\%20blog\%20is\%20intended\%20to,access\%20to\%20information\%20in\%20Jira..

[25] *Jira Web Site.* URL: https://www.atlassian.com/software/jira.

[26] Kenji Kira and Larry A Rendell. "A practical approach to feature selection". In: *Machine learning proceedings 1992.* Elsevier, 1992, pp. 249–256.

[27] Dean Leffingwell. *Agile Software Requirements: Lean Requirements Practices for Teams, Programs, and the Enterprise (Agile Software Development Series).* Addison-Wesley Professional, 2011. ISBN: 978-0321635846.

[28] Mary Lotz. *Waterfall vs. Agile: Which is the Right Development Methodology for Your Project?* URL: https://www.seguetech.com/waterfall-vs-agile-methodology/.

[29] *Machine Learning Engineer Salary.* URL: https://www.payscale.com/research/ES/Job=Machine_Learning_Engineer/Salary.

[30] *Machine Wisdom.* URL: https://machineswisdom.com/.

[31] Troy Magennis. "Managing Software Development Risk using Modeling and Monte Carlo Simulation". In: (May 2012). DOI: 10.13140/2.1.3708.1927.

[32] *Manifesto for Agile HR Development.* URL: https://www.agilehrmanifesto.org/.

[33] *Markov property.* URL: https://en.wikipedia.org/wiki/Markov_property.

[34] Roger S. Pressman / Bruce R. Maxim. *Software Engineering A Practitioner's Approach Eight Edition.* McGraw-Hill International Edition, 2015. ISBN: 978-1-259-25315-7.

[35] Morakot Choetkiertikul Hoa Khanh Dam Truyen Tran Trang Pham Aditya Ghoseand Tim Menzies. "A deep learning model for estimating story points". In: (2016).

[36] *Min Max Scaler Sklearn.* URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.MinMaxScaler.html?.

[37] Christopher Z Mooney. *Monte carlo simulation.* 116. Sage, 1997.

[38] *Open AI Better Language Models.* URL: https://openai.com/blog/better-language-models/.

[39] *Ordinal Encoder Sklearn.* URL: https://scikit-learn.org/stable/modules/generated/sklearn.preprocessing.OrdinalEncoder.html.

[40] *Pypi Google Trans.* URL: https://pypi.org/project/googletrans/.

[41] Jonathan Rasmusson. *The Agile Samurai.* The Pragmatic Programmers, 2014. ISBN: 978-1-934356-58-6.

[42] Jose Maria de los Santos. *Agile vs. Waterfall: Differences in Software Development Methodologies?* 2020. URL: https://project-management.com/agile-vs-waterfall/#:~:text=Agile%20is%20an%20incremental%20and,divides%20a%20project%20into%20phases.&text=Agile%20allows%20requirement%20changes%20at,changes%20once%20the%20project%20starts..

[43] Ken Schwaber and Jeff Sutherland. *Scrum Guide.* URL: https://www.scrum.org/resources/scrum-guide?gclid=CjwKCAiAv4n9BRA9EiwA30WND2QekiFhasysAVmJI8rukSdIr99cLftZKB0u70s_UPGUn3kjKYCiuRoCQQgQAvD_BwE.

[44] *Simple Imputer Sklearn.* URL: https://scikit-learn.org/stable/modules/generated/sklearn.impute.SimpleImputer.html.

[45] Guillem Hernandez Sola. *¿Cuál es la diferencia entre Scrum y Agile?* URL: https://www.scrum.org/resources/blog/cual-es-la-diferencia-entre-scrum-y-agile.

[46] *Sueldo para Project Manager.* URL: https://www.glassdoor.es/Salaries/madrid-project-manager-salary-SRCH_IL.0,6_IM1030_KO7,22.htm?countryRedirect=true.

[47] Richard S Sutton and Andrew G Barto. *Reinforcement learning: An introduction.* MIT press, 2018.

[48] *Technical Writer Salary.* URL: https://www.payscale.com/research/ES/Job=Technical_Writer/Salary.

[49] Ian Tenney, Dipanjan Das, and Ellie Pavlick. "BERT Rediscovers the Classical NLP Pipeline". In: *CoRR* abs/1905.05950 (2019). arXiv: 1905.05950. URL: http://arxiv.org/abs/1905.05950.

[50] John W Tukey et al. *Exploratory data analysis.* Vol. 2. Reading, Mass., 1977.

[51] *UX Designer Salary.* URL: https://www.payscale.com/research/ES/Job=UX_Designer/Salary/85647e62/Barcelona.

[52] *UX Developer Salary.* URL: https://www.payscale.com/research/ES/Job=Front_End_Developer_\%2F_Engineer/Salary.

[53] Raj Vardhman. *Agile Adoption.* 2020. URL: https://goremotely.net/blog/agile-adoption/#:~:text=The%20Agile%20failure%20rate%20is%208%25%2C%20while%20the,Waterfall%20failure%20rate%20is%2021%25&text=Let's%20look%20at%20it%20from,lower%20than%20the%20Waterfall%20one..

[54] *Velocity Chart.* URL: https://confluence.atlassian.com/jirasoftwareserver/velocity-chart-938845700.html.

[55] *Version Report Chart.* URL: https://confluence.atlassian.com/jirasoftwareserver/version-report-938845705.html.

[56] *View and understand the cummulative flow diagram.* URL: https://support.atlassian.com/jira-software-cloud/docs/view-and-understand-the-cumulative-flow-diagram/.

[57] *What is Agile for HR.* URL: https://www.agilehrcommunity.com/agilehr-community-blog/what-is-agile-hr-your-step-by-step-guide-and-handy-infographic.

[58] Julio Christian Young and Andre Rusli. "Review and Visualization of Facebook's FastText Pretrained Word Vector Model". In: *2019 International Conference on Engineering, Science, and Industrial Applications (ICESI).* IEEE. 2019, pp. 1–6.