Energy consumption of machine learning deployment in cloud providers

Bachelor’s Degree in Informatics Engineering

Specialization in Software Engineering

Author: Daniel Escribano

Director: Silverio Martínez-Fernández

Co-director: Xavier Franch

September, 2022

Abstract

Machine learning, a particular type of artificial intelligence, is increasingly being used in modern software systems. In addition to requiring a lot of energy, its use raises CO_2 emissions, has a high financial cost, and consumes many resources. Even while there are numerous studies that look at the possibilities of ML, very few of them concentrate on its green features, like energy consumption or its carbon footprint.

This bachelor thesis tries to increase understanding of the costs associated with deploying an ML system into some of the state-of-the-art cloud hosting platforms. In order to do this, we conduct an extensive empirical study to assess and contrast the emissions and latency of four different cloud providers: Amazon Web Services, Azure, Heroku, and Railway.

Resumen

El aprendizaje automático, un tipo particular de inteligencia artificial, se utiliza cada vez más en los sistemas de software modernos. Además de requerir mucha energía, su uso aumenta las emisiones de CO_2, tiene un elevado coste financiero y consume muchos recursos. Aunque existen numerosos estudios que analizan las posibilidades del ML, muy pocos de ellos se centran en sus características ecológicas, como el uso de energía o su huella de carbono.

Este estudio trata de aumentar la concienciación de los costes asociados al despliegue de un sistema de ML en algunas de las plataformas de alojamiento en la nube más avanzadas. Para ello, llevamos a cabo un amplio estudio empírico para evaluar y contrastar las emisiones y el la latencia de cuatro proveedores de nube diferentes, Amazon Web Services, Azure, Heroku y Railway.

Resum

L'aprenentatge automàtic, un tipus particular d'intel·ligència artificial, s'utilitza cada vegada més en els sistemes de software modern. A més de requerir molta energia, el seu ús augmenta les emissions de CO_2, té un elevat cost financer i consumeix molts recursos. Encara que existeixen nombrosos estudis que analitzen les possibilitats del ML, molt pocs d'ells se centren en les seves característiques ecològiques, com l'ús d'energia o la seva petjada de carboni.

Aquest estudi tracta d'augmentar la conscienciació dels costos associats al desplegament d'un sistema de ML en algunes de les plataformes d'allotjament en el núvol més avançades. Per a això, duem a terme un ampli estudi empíric per a avaluar i contrastar les emissions i la latencia de quatre proveïdors de núvol diferents, Amazon Web Services, Azure, Heroku i Railway

Index

1. Motivation and context

This project is a degree final project from the Informatics Engineering degree at the *Facultat d'Informàtica de Barcelona,* specifically one belonging to the specialization of Software Engineering. This project is co-directed by Silverio Martínez-Fernández and Xavier Franch Gutiérrez and it is an A modality project, without the aid of any external institution.

This chapter is divided into the following sections: the main concepts, the problems to be solved, the stakeholders involved, the project justification, and the structure of this document.

1.1 Concepts

To better understand the project, the reader should be acquainted with the following concepts:

- **Green AI:** Term initially employed by Schwartz et al.[1] to define the investigation on AI that provides new breakthroughs, and at the same time takes into consideration the computational cost of the study, pushing for the reduction in the resources employed and trying to establish a better relationship between performance and efficiency. In contrast, there exists the so-called "Red AI", research that promotes the improvement of the precision of AI at any cost, neglecting the impact and cost of reaching that kind of results
- **Machine learning:** According to IBM [2], machine learning is a branch of artificial intelligence and computer science that focuses on the use of data and algorithms to imitate the way that humans learn, gradually improving its accuracy in different fields, such as image recognition.
- **Training and inference:** These can be considered the two phases that a machine learning model goes through. The training phase is the first phase and in this, the model receives a set of training data so that it can learn everything about the data that it should analyze in the future. In the inference phase, the model is considered to have learned enough and it can be used to make predictions about new data that it has not seen before.
- **CO_2eq emissions [3]:** Standardized measure used to express the global warming potential of various greenhouse gases: the amount of CO_2 that would have the

equivalent global warming impact as the amount of gas measured. For computing, carbon emissions are measured in kilograms of CO_2-equivalent emissions.

- **Carbon Intensity[4]:** The number of carbon dioxide kilograms that it takes to make one kilowatt-hour of electricity. This carbon intensity tends to be calculated as the weighted average of the emissions from the different energy sources that are used to generate electricity in the energy grid of a geographic area.

1.2 Problem to be solved

Even though there have been a lot of studies under the “Green AI” field, not so many have been done analyzing the configuration of the deployment of the models, especially in terms of the inference phase. If the model is deployed on the cloud or the edge can be a critical change in terms of how the model can take advantage of the resources at its disposal on the machine where it is located. Adding additional configurations to the model, like containerization, can also impact the performance of the system, in terms of run-time performance and energy consumption

On the other hand, in few cases the number of emissions generated by an ML system is measured, and in even fewer these emissions are measured on the inference phase. Usually, the majority of studies where energy efficiency is measured are related to the training phase of the model and not focusing as much on the emissions, giving more attention to energy consumption.

To provide some new insights on not so explored metrics, the chosen measurements for this study are the CO_2eq emissions resulting from the energy consumption of a model and the latency in the inference of the model.

The energy consumption of a model tends to be measured in terms of the energy consumed by the computation performed, but when a system is deployed into a certain configuration (be it on the cloud or on-premise deployment), the underlying infrastructure has a certain environmental impact based on the carbon intensity of its energy grid (the type of energies employed for the generation of electricity). Choosing the correct configuration could make a certain amount of energy consumed more or less harmful to the environment and, because of this, developers and researchers should be aware of the sustainability benefits a certain deployment configuration can provide when compared to others.

On the other hand, latency is an essential point that developers and DevOps engineers need to consider and manage when deploying their systems. Network latency, in the context of a computing system, refers to the delay in network communication. It shows the time that data takes to transfer across the network. Networks with a longer delay have high latency, while those with fast response times have low latency[5].

When it comes to comparing trade-offs in ML studies, researchers tend to focus on metrics such as the precision or the accuracy of the model, but in the context of this study, latency is a much more interesting metric. As the inference of the model tends to have a fixed precision (as the precision is determined by how the model is trained), examining how the latency of a system containing an ML model varies when using different deployment configurations can be more useful to the community

For these reasons, we believe a study that experiments on different ML configurations and reports the results obtained on emissions and latency is necessary and could be of use to future researchers who work in the green AI field.

1.3 Stakeholders

According to the benefits that they can obtain from this project, we can divide the stakeholders into two different groups.

The ones that can obtain a direct benefit, and that are directly involved in the project, are the director Silverio Martínez-Fernandez and me, Daniel Escribano. As the director has previously worked on the domain [6], he can be looking forward to the correct and proper development of this project. On the other hand, the researcher, Daniel Escribano, is the one responsible for designing, developing, and documenting the project, carrying out the experiments, and drawing conclusions

On the other hand, there are those who are indirectly involved with the project. These would be the future researchers, students and developers who can access this project and use the information and conclusions obtained to expand onto these and use them in future studies and projects.

Also, finding a deployment configuration that proves to be energetically efficient can prove of use for businesses and start-ups. These could use this configuration as a template for their future products, be it because of a necessity of reducing the consumption of their systems or to scale their systems from an efficient and green base configuration.

1.4 Project justification

In recent years, there has emerged a notable increase in the number of studies with the objective of evaluating the energetic efficiency inside the ML field.

García-Martín et al.[7] submitted a study where they expose some of the guidelines one should follow when implementing energy efficiency as a metric to consider when designing machine learning systems.

Other studies have been conducted to try to define the most fitting configuration for each stage of the development process for ML systems. For example, Georgiou et al.[8] discovered that one model can show differences in its energy efficiency and execution time depending on the framework chosen for its development. Other studies like Hampau et al.[9] focus on the different containerization strategies available in a development environment. In these, different metrics of energy performance and efficiency are studied in contrast to the computational resources the device employs to perform the experiment. Strubell et al. [10] also report on the CO_2 emissions of different successful models, to shed light on how costly the training of large models really is.

These studies provide valuable results for future researchers who take into consideration energy sustainability when designing new investigations, besides proposing new guidelines to follow when designing similar experiments as the ones performed on their work. Optimizing models that have a high resource demand is an important step forward to battle the current tendency inside the field.

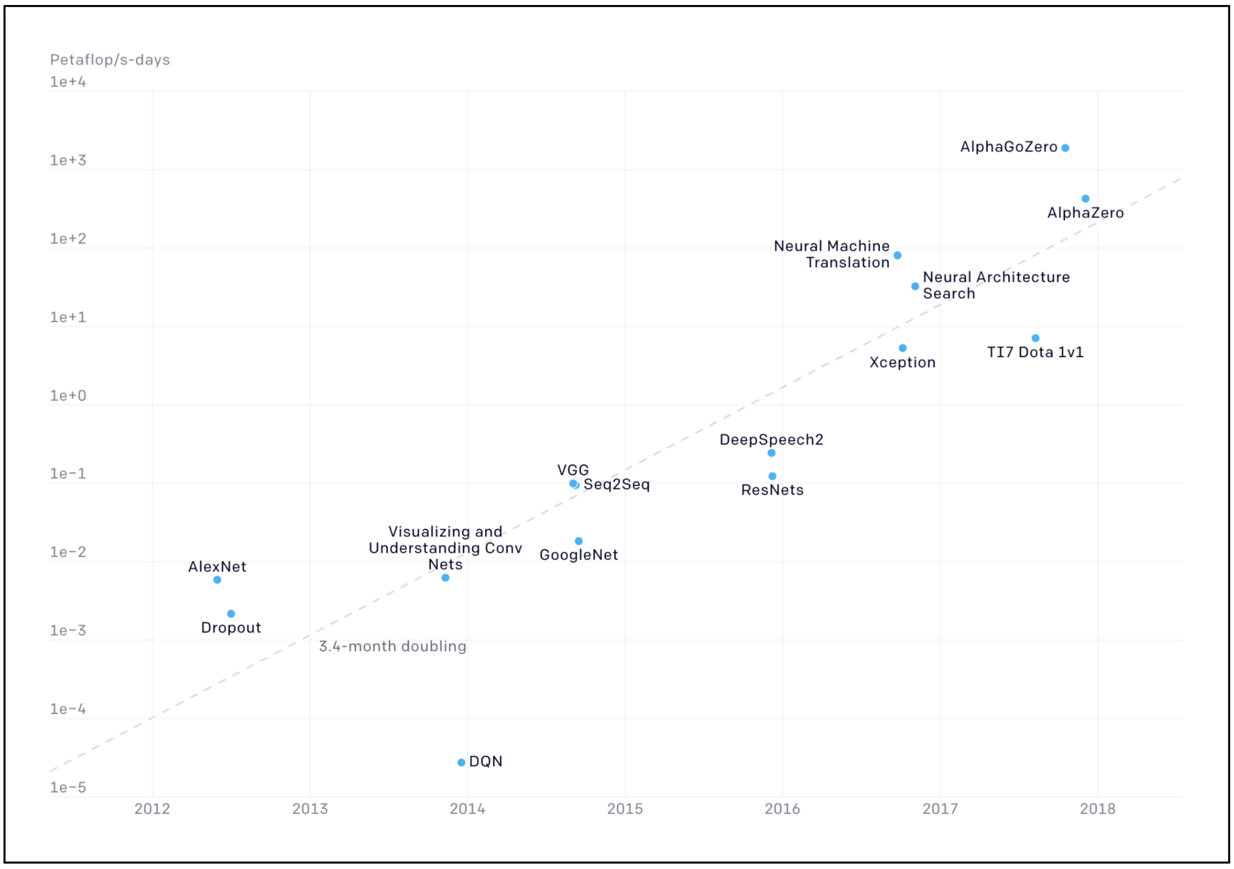


Figure 1: The amount of compute used to train deep learning models between 2012 and 2018. Figure taken from Amodel et al.[https://openai.com/blog/ai-and-compute/]

Because of these reasons, it is important to persist on the tendency exposed by the studies falling in the “Green Software Engineering” field. And to be consistent with the previous statement and with the goal of creating a new study that can contribute, that is why the project was started in the first place. A new study that explores the possibilities of the deployment of an ML model can be of great utility to the community, as there aren’t many works that explore the tradeoff between emissions and latency in contrast to the deployment configuration of the system.

1.5 Structure of this document

This document is structured as follows.

Section 2 shows the objectives and requirements set for this project. Section 3 explains the work methodology used. Sections 4 and 5 explain the logistic side of the project: how it is divided into tasks and the budget needed to perform these. Section 6 provides a sustainability report of the project.

Section 7 explains the design of the system which would be deployed and the chosen platforms of deployment. Section 8 explains how the experiment was designed, how the measures were retrieved, and how the data collected was analyzed. Section 9 shows the results obtained from the experiments, and Section 10 discusses the results obtained. Section 11 gives an overview of how the development process ended up being, and Section 12 describes the deviations from the initial planning and budget.

Section 13 provides the conclusions to this project, along with showing the knowledge integration of different subjects from the degree and what work could be done in the future. And to conclude, Section 14 contains some acknowledgements the researcher wanted to give.

2. Scope

This section describes the objectives and sub-objectives set to accomplish in this thesis along with the requirements the software system needs to meet and the potential risks the project can face.

2.1 Objectives

The main objective of this study is:

To study and analyze the different configurations of ML deployment, study the energy efficiency of the system in terms of CO_2eq emissions, and observe the relation these emissions have with the latency of the deployed system.

Thus, the project will be centered around studying the effect that deployment configurations have on the tradeoff between carbon emissions and performance parameters such as the latency of the model.

To reach this goal the project can be split into several sub-objectives.

2.2 Sub-objectives

- **SO1. Establish energy consumption and model performance parameters that make room for a reliable study.**
- **SO2. Explore and select measuring tools for the chosen energy consumption metric that provide rigorous and valid data.**
- **SO3. Design an experimentation methodology that enables the discovery of differences between deployment configurations, on the subject of the previously mentioned parameters.**
- **SO4. Search for a configuration that proves a good trade-off between energy efficiency and latency.**

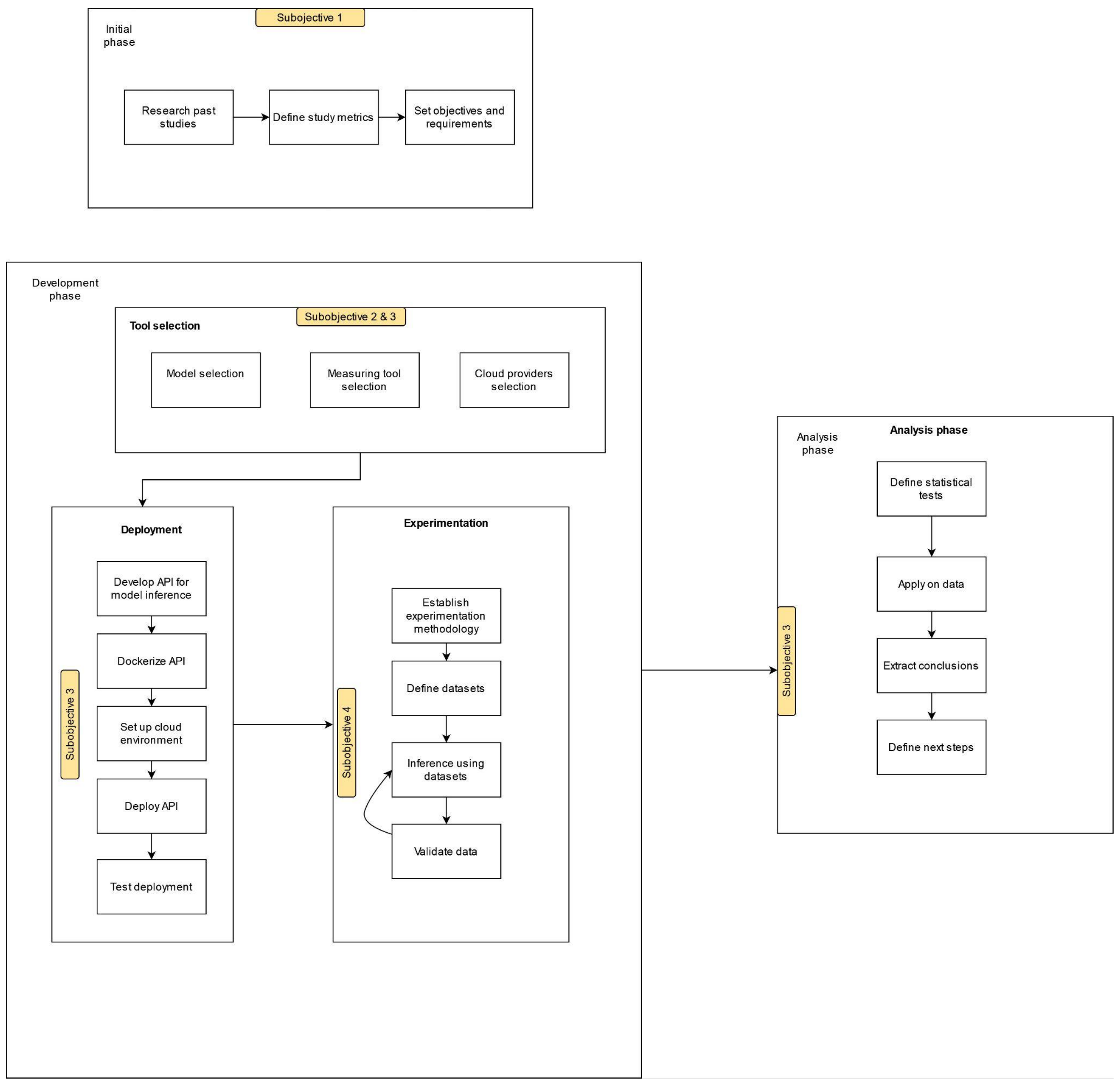


Figure 2: Diagram of tasks and objectives of this project

2.3 Requirements

In this section, the functional and non-functional requirements that the system has will be displayed.

These requirements have been specified using Volere [11] and each individual requirement will show: the type of requirement according to Volere, description, justification, and acceptance criteria.

2.3.1 Functional requirements

- **Type of requirement**: 9 -Functional requirement
- **Description:** The user wants to perform inference on the deployed model and obtain results from specific data
- **Justification:** Needed in order to accomplish SO3
- **Acceptance criteria:**
    - The user will be able to perform requests on the model deployed via an interface (API) across the Internet
    - The system will process the data sent by the user through the model and return its output

2.3.2 Non-functional requirements

- **Type of requirement**: 14c -Adaptability
- **Description:** The system needs to be adaptable enough to make room for future studies
- **Justification:** Needed in order to accomplish SO3
- **Acceptance criteria:**
    - The system needs to support the change of the model used and the requirements needed to run the model without the need of extra configuration

- **Type of requirement**: 15b -Integrity
- **Description:** All of the data processed by the system needs to be correct
- **Justification:** Needed in order to provide reliable results and accomplish SO4
- **Acceptance criteria:**
    - The data sent to the deployed model will be validated, both by restricting the type of data a user can send and by validating the data at the moment of performing inference on the model

- **Type of requirement**: 12f - Capacity
- **Description:** The system needs to be available and stable during the execution of the experiment
- **Justification:** Needed in order to properly execute the experiment and accomplish SO4
- **Acceptance criteria:**
    - The system needs to be able to withstand at least 40 consecutive inference requests without any error.

- **Type of requirement**: 11c - Learning
- **Description:** The system needs to have a small learning curve for developers not familiar with it
- **Justification:** Needed in order to accomplish SO3
- **Acceptance criteria:**
    - The system's code needs to be easy enough to read and understand to someone unfamiliar with it that wishes to use it in the future.

2.4 Drawbacks and potential risks

Throughout the development of the project, it is possible that risks and drawbacks arise. Some of these are

- **Lack of experience in the research field:** The author of the project has no previous experience in the research area, and this may result in dedicating additional resources to learning the technologies used.

- **Lack of measuring tools:** Given the nature of the energy metric that the study intends to measure(carbon dioxide equivalent emissions) it is possible that the available tools do not provide valid measurements.
- **Unattainable resources:** It is possible that some of the deployment platforms or tools needed to conduct the study aren't available due to financial issues or because of a lack of compatibility between the different components of the experiment.
- **Deadlines:** Considering that the project has a deadline for its delivery and that there are several deliveries before the final one, time is a factor to take into account when taking decisions on the development of the project. It is possible that certain changes have to be applied to fit the time constraints.

3. Methodology

The development of this project ties to a study about the possibilities in the deployment configuration, therefore it is important to use a work methodology that allows testing these configurations,making room for possible changes and discarding options for new ones which prove to be of more interest. The agile methodology employed in software development [12] is a good candidate for this, as defining short development cycles makes it possible to perform these tests.

Although the methodology used is an agile methodology, certain aspects of these are omitted for the project. To better profit from this methodology, only the sprints and periodic meetings with the directors will be used. On each sprint, certain goals will be fulfilled and at the review meeting at the end of the sprint, these will be evaluated. Based on the work done in the sprint and on the general development of the project, new goals and tasks to complete will be defined for the next sprint.

4. Temporal planning

The following chapter describes the tasks that will be done during the evolution of the project. Along with these descriptions, the resources needed to perform the tasks will also be described, together with a Gantt chart to illustrate the time evolution of the tasks. To close the chapter, the strategies set to manage the potential risks described in Section 2.4 will be explained.

4.1 Tasks description

Down below there is a description of the tasks performed throughout the project's development. For each one there is a brief description along with an identifier, its hourly dedication, and the dependencies to other tasks. Table 1 shows this information in a more compact manner.

4.1.1 Project planning

Given the importance of the tasks in the project, it is important to properly define them in order to not miss out on anything. In these tasks, periodic meetings with the director of the project are also included.

- **PP1 - Scope and context:** Definition of the scope of the project, the context which surrounds it, and the justification which proves the need for this study. It also includes the objective, the risks, and the methodology to follow through with the development.
- **PP2 - Temporal planning:** Planning of the tasks to be done through the project development. It also includes the graphic representation of the temporal planning via a Gantt diagram, a description of the resources needed, and a description of how these will be used and managed
- **PP3 - Budget, financial viability, and sustainability:** Analysis of the sustainability and environmental impact of the project, along with the planning of the monetary resources which will be directed towards the development of the project.
- **PP4 - Final deliverable:** Integration of the past documents in a single file, to present as the project definition document

- **PP5 - Periodic meetings:** Meetings done regularly with the director of the project. Each one has an approximate duration of 30 minutes and is done in order to analyze the evolution of the project.
- **PP6 - Documentation:** Documentation written throughout the project. In the course of the project events that arise will be written down and, once the project has finished and the results have been recollected, the final memory will be written.

4.1.2 Model research

In this project, as the objective is to experiment and study the inference phase of the ML system, there will be no special emphasis on the training phase of the models used. Because of this the models used will be pre-trained and extracted from online open repositories.

- **EM1 - Repository research:** Search for available repositories which offer trained machine learning models ready to use. Search criteria will be that the models offer detailed information on how they were trained, the purpose of the model, and how it can be used.
- **EM2 - Model selection:** Initial selection of models that can be candidates for posterior experimentation. The selection criteria consists in looking for models which report on the energy footprint of its training phase, along with being a light model in terms of memory size and providing interesting differences between each other.
- **EM3 - Data extraction and final selection:** After the candidate selection, a final decision will be made to choose the models to be used in the experiments, giving preference to the lightest ones. This decision will be made following the advice of the director.

4.1.3 Deployment configuration research

As this project has a strong emphasis on research, it is of the utmost importance to research and study thoroughly each of the components which will be used prior to experimenting. As the deployment configuration is one of the most important points of the study (if not the most), a large amount of time will be allocated to learning and researching all of the options one can modify in the deployment of ML systems.

- **EC1 - Study of deployment architectures:** Machine learning models can be deployed in different computational environments such as the cloud or the edge. Because of this, it is relevant to understand how every environment works to make an educated decision.
- **EC2 - Study of additional tools for inference:** Besides choosing an architecture, the use of additional tools can lead to developing a relevant study that can be of interest. Such tools can be, for example, containerization of the application or using inference servers.
- **EC3 - Study of measuring tools for energy metrics:** In order to compare the inference of the different configurations, it is important that the tools which measure the energy consumption are valid and produce reliable results
- **EC4 - Performance tests:** Once the tools and the models have been selected, brief tests will be performed in order to better understand the utility of the tools and to check that there aren't compatibility issues.

4.1.4 Experimentation

Once the study to select the components of the application has been done, the experimentation on the system will begin. In order to obtain quality data, it is crucial to define a rigorous experimentation methodology.

- **EX1 - Dataset selection:** To test the models, we need to feed them with data they can perform the inference on. These datasets must be representative of the model functionality, in order to showcase the working of the model
- **EX2 - Definition of the benchmarks:** Once the datasets have been established, the methodology to follow when performing the experiments must be defined.
- **EX3 - Experimentation:** Once the previous parameters have been set, the models have to be deployed and experimented on, collecting the data generated for the posterior analysis.

4.1.5 Conclusions and analysis

After experimenting and collecting data, statistical analysis must be performed to draw accurate conclusions.

- **AC1 - Definition of statistical tests:** Investigate which statistical tests can be valuable for the data obtained from the experimentation.
- **AC2 - Statistical tests appliance:** Apply statistical tests over the experimentation data, in order to evaluate the results and draw conclusions from them
- **AC3 - Conclusions:** Analyze both the results obtained from the experimentation and the results from the statistical tests to evaluate if the objectives set have been accomplished or not.

4.2 Resources

To be able to perform the previous tasks we will need two types of resources: human resources and material resources.

4.2.1 Human resources

The main asset when it comes to human resources is the researcher itself. As this is who will work on the development of the project, he will be responsible for the project's success. Another important asset to keep in mind is the director of the project, as he will be in charge of guiding the less experienced researcher and will be able to give the project a proper structure. Lastly, the GEP tutor will also be important, as he will be in charge of reviewing and giving the feedback necessary for establishing sensible initial planning of the project.

4.2.2 Material resources

Like any project heavily based on research, there is a great need of consulting previous studies done on the same matter, and because of this, it is essential to be able to access books and academic papers. Furthermore, to be able to do the practical part of the project both hardware and software resources will be needed.

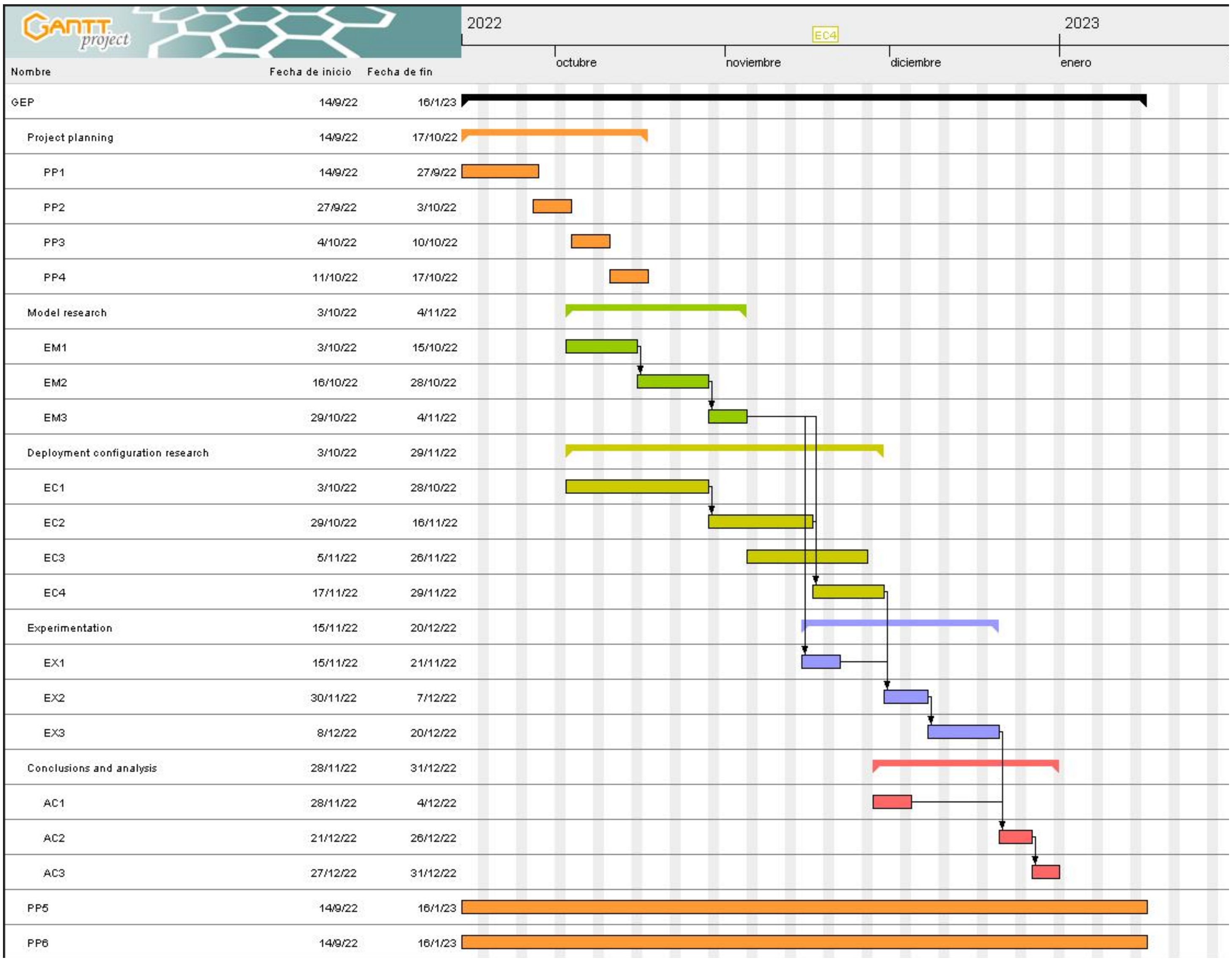
- **“El Racó”:** Official website belonging to the *Facultat d’Informàtica de Barcelona.* Needed in order to communicate with the GEP tutor and to be able to deliver the project’s memory
- **Github:** As an Internet hosting service for software development and version control using Git, Github will be used in order to store the developed code in a safe and structured manner.
- **G-Suite:** Tasks such as the redaction of the memory, online meetings with the director, and setting up future meetings with the director, among others, will be done using the tools available in this suite.
- **Gantt Project:** For the design of the Gantt diagram that shows the timeline of the tasks to perform, we will use this free software application.
- **Personal computer:** As the project won’t be sponsored by any company, the researcher will have to use his personal computer to work on the project.

4.3 Task summary

ID	Name	Time (h)	Dependencies
PP	Project planning	100	
PP1	Scope and context	15	
PP2	Temporal planning	9	
PP3	Budget, financial viability, and sustainability	8	
PP4	Final deliverable	8	
PP5	Periodic meetings	20	
PP6	Documentation	40	
EM	Model research	75	
EM1	Repository research	30	
EM2	Model selection	30	EM1
EM3	Data extraction and final selection	15	EM2
EC	Deployment configuration research	165	
EC1	Study of deployment architectures	50	
EC2	Study of additional tools for inference	40	EC1
EC3	Study of measuring tools for energy metrics	50	
EC4	Performance tests	25	EC2,EM3
EX	Experimentation	75	
EX1	Dataset selection	15	EM3
EX2	Definition of the benchmarks	20	EX1,EC4
EX3	Experimentation	40	EX2
AC	Conclusions and analysis	45	
AC1	Definition of statistical tests	20	
AC2	Statistical tests appliance	15	AC1,EX3
AC3	Conclusions	10	AC2
Total		460	

Table 1: Task summary of the tasks to perform

4.4 Gantt



Nombre	Fecha de inicio	Fecha de fin
GEP	14/9/22	16/1/23
Project planning	14/9/22	17/10/22
PP1	14/9/22	27/9/22
PP2	27/9/22	3/10/22
PP3	4/10/22	10/10/22
PP4	11/10/22	17/10/22
Model research	3/10/22	4/11/22
EM1	3/10/22	15/10/22
EM2	16/10/22	28/10/22
EM3	29/10/22	4/11/22
Deployment configuration research	3/10/22	29/11/22
EC1	3/10/22	28/10/22
EC2	29/10/22	16/11/22
EC3	5/11/22	26/11/22
EC4	17/11/22	29/11/22
Experimentation	15/11/22	20/12/22
EX1	15/11/22	21/11/22
EX2	30/11/22	7/12/22
EX3	8/12/22	20/12/22
Conclusions and analysis	28/11/22	31/12/22
AC1	28/11/22	4/12/22
AC2	21/12/22	26/12/22
AC3	27/12/22	31/12/22
PP5	14/9/22	16/1/23
PP6	14/9/22	16/1/23

Figure 3: Gantt's diagram

4.5 Risk management

As explained in Section 2.4, there are some risks and obstacles that can arise through the development of the project. In this section, there is an explanation of how these will be handled, if it's possible to do so, or what alternative plans will be applied in case the problems can't be solved. A level of Low, Medium, High, or Extreme will be assigned to each problem to represent the impact it would have.

4.5.1 Lack of experience in the research field

- **Risk level:** Medium
- **Proposed solution:** In order to mitigate this risk, more hours can be allocated to the research previous to the experimentation, delaying the time schedule of the tasks. These hours will be drawn from other tasks such as AC1, EC4, or EX1, as they can be deemed "less" important.

4.5.2 Lack of measuring tools

- **Risk level:** High
- **Proposed solution:** Given the nature of the metrics (CO_2eq emissions) it is possible that the available tools to measure energy consumption do not have access to these metrics or aren't capable of calculating the metrics by themselves. Given this scenario, the metrics to measure could be replaced with others that can be obtained in order to be able to perform a comparison. This would extend the duration of task EC3, adding a margin of up to 20 additional hours.

4.5.3 Resource unavailability

- **Risk level:** Medium
- **Proposed solution:** It is possible that some of the resources that initially are proposed as study objects can't be accessed due to a subscription paywall. If the project encounters this scenario, a meeting would be held with the project director in order to discuss the financial viability of paying the subscription for the platform or product. If the product is deemed unworthy of payment, an alternative product would be used instead. To properly assess and search for a correct replacement, an increment in

hours for tasks EC2 and EC4 would be necessary, setting a maximum limit of 30 hours

On the other hand, it could be possible that, because of the nature of the chosen tools and configurations, some measuring tools aren’t compatible with certain environments. Given this scenario, an extraordinary meeting would be held with the director to handle this obstacle

4.5.4 Deadlines

- **Risk level:** High
- **Proposed solution:** In the case that the amount of hours estimated is incorrect (probable, given that the lack of experience is important) and the project looks like it won’t be able to finish, the remaining tasks would have to be replanned in order to give a relevant conclusion for the problem exposed in the scope section. If the replan is not enough, the number of hours would be incremented.

5. Budget

In this section, the economic cost of the project will be described. To properly determine the cost, the cost of the staff involved will be described, specifying the necessary roles along with the generic and indirect costs. The procedures employed to manage the deviations from the budget will also be explained.

5.1 Personnel cost per activity (PCA)

In this section the total cost for each task defined in the previous section will be calculated. Adding the cost of each staff member associated with the task will result in the total cost for the task. The cost of each involved worker will be obtained by multiplying the hours dedicated to the task per the hourly cost.

Different roles can be assigned to the project to handle the different tasks presented (if the project was developed in a real company). These roles have different hourly rates (see table 2). The first to mention is the project manager. This role will be in charge of managing the planning of the project. On the other hand, there also exists the researcher, in charge of conducting the previous study of the models and configurations to use, along with writing down the results and experiments done.

On the technical side, the project will also need a DevOps profile to manage and implement the different deployment strategies established. The benchmarks for the experiments will be designed by a tester, so this role will also play a part in the development of the project. Finally, a data analyst will be needed to design the statistical tests that will be used to draw conclusions from the data obtained from the execution of the experiments.

Given this definition of roles, we can create a relationship between the Gantt chart and the established roles, to visualize the hours dedicated to each role and the cost per task.

Role	Yearly cost(€)	Cost per hour(€)
Researcher (RS)	22,192 [13]	12,66
Project manager(PM)	45.683 [14]	26
DevOps Junior (DO)	38.341 [15]	21,87
Software Tester (ST)	20.347 [16]	11,6
Analyst (AN)	28.990 [17]	16,53

Table 2: Yearly cost and cost per hour of different roles. The number of hours per year is estimated to be 1753 hours [18].

Task	Hours	RS	PM	DO	ST	AN	Cost (€)
Project planning	100	40	60	0	0	0	2066.4
Scope and context	15	0	15	0	0	0	390
Temporal planning	9	0	9	0	0	0	234
Budget, financial viability, and sustainability	8	0	8	0	0	0	208
Final deliverable	8	0	8	0	0	0	208
Periodic meetings	20	0	20	0	0	0	520
Documentation	40	40	0	0	0	0	506.4
Model research	75	75	0	0	0	0	949.5
Repository research	30	30	0	0	0	0	379.8
Model selection	30	30	0	0	0	0	379.8
Data extraction and final selection	15	15	0	0	0	0	189.9
Deployment configuration research	165	90	0	75	0	0	2779.65
Study of deployment architectures	50	20	0	30	0	0	909.3
Study of additional tools for inference	40	20	0	20	0	0	690.6
Study of measuring tools for energy metrics	50	50	0	0	0	0	633
Performance tests	25	0	0	25	0	0	546.75
Experimentation	75	0	0	0	75	0	870
Dataset selection	15	0	0	0	15	0	174
Definition of the benchmarks	20	0	0	0	20	0	232
Experimentation	40	0	0	0	40	0	464
Conclusions and analysis	45	0	0	0	0	45	743.85
Definition of statistical tests	20	0	0	0	0	20	330.6
Statistical tests appliance	15	0	0	0	0	15	247.95
Conclusions	10	0	0	0	0	10	165.3
Total	460	205	60	75	75	45	7409.4

Table 3: Cost and time estimated per role for each task

To finish, in the following table we can see the personnel cost per activity.

Role	Hours	Cost (€)
Researcher	205	2595.3
Project manager	60	1560
DevOps Junior	75	1640.25
Software Tester	75	870
Analyst	45	743.85
Total	460	7409.4

Table 4: Total cost of the personnel

5.2 Generic costs

5.2.1 Amortization

To fully describe the economic cost of the project, the amortization of the employed resources has to be taken into consideration.

Each day of work will average 3 hours, throughout the length of 124 days. All of the work will be performed on a Huawei laptop, using a secondary monitor. To calculate the amortization of this laptop, the following formula will be used:

$$Amortitzation = Resource\ cost \times \frac{1}{Usage\ years} \times \frac{1}{Working\ days} \times \frac{1}{Hours\ per\ day} \times Total\ hours$$

Hardware	Cost (€)	Years of usage	Amortization(€)
Huawei laptop	700	3	288.53
Secondary monitor	125	6	25,76
Total	825		314,29

Table 5: Amortization of the employed hardware

5.2.2 Indirect costs

To create a realistic budget, the indirect costs of the project have to be defined correctly. As the project will be done primarily on the student's residence, the costs of the electricity bill among others have to be taken into account.

- **Electricity bill:** Currently, the kWH price [19] is at .1736 €/kWh.To get the cost of the electricity: power(in kW) * consumption hours * kW/h price.

Device	Power	Hours	Cost(€)
Laptop	65 W	460	5,19
Monitor	42 W	460	3,35
Total			8.54

Table 6: Electrical cost of the hardware used in the project

- **Internet cost:** The contracted fee costs 67 € per month. The total cost of the internet can be calculated by: 67 €/month * 4 months * 3 working hours/24h = 33,5 €
- **Working area:** As the project will be developed at the student's residence, the rental cost has to also be taken into account. The rental of the apartment costs 1400€/month. As the apartment is shared among 4, the price for the working area falls to 350 euros per month. Considering that the project will last 5 months, the price rises to 1750 €.

5.2.3 Total generic costs

Concept	Cost (€)
Amortization	314,29
Electricity	8,54
Internet	33,5
Working area	1750
Total GC	2106.33

Table 7: Total generic costs

5.3 Other costs

5.3.1 Contingencies

As every other project, some unexpected events can pop up throughout the course of the project. For this reason, some resources have to be kept apart to palliate the effects of these events and not alter the initial planning of the project.

Following the market's standards, a contingency percentage of 15% will be used.

To obtain the cost of this contingency the total PCA (7409.4) has to be multiplied by the percentage (0.15)

5.3.2 Risks

Estimating the cost that the previously mentioned risks can have on the project, the cost of the alternative plans have to be taken into consideration. These alternative plans can be seen in the previous sections. In the following table, the estimated costs of the execution of the alternative plans can be seen. These costs can be calculated by multiplying the cost of solving the problem (in the worst case scenario) by the chance of the risk actually happening .

Incident	Estimated cost(€)	Risk(%)	Cost (€)
Lack of measuring tools (5 - 20 hours)	253,2	40	101,28
Resource unavailability (10 - 30 hours)	656,1	20	131,22
Deadlines (10 - 25 hours)	316,5	20	63,3
Total	1225,8	-	295,8

Table 8: Cost of alternative plans for each risk

Each alternative plan is assigned to the role in charge of executing the task. For the risk related to resource unavailability, the role in charge of executing the alternative plan is DevOps. The other two alternative plans will be executed by the researcher.

5.4 Total cost

In the following table, the cost for the whole project is displayed.

Activity	Cost (€)
PCA	7409.4
GC	2106.33
Contingency margin	1111,41
Risk management cost	295,8
Total	10922,94

Table 9: Total cost of the project

5.5 Management control

With projects such as this, sometimes it is hard to give a correct estimate of the time and budget necessary because of the problems that can arise during development. For this reason, a method will be used to control the possible deviations in the budget. After completing the project, the time deviation will be computed, with the following formula.

$$Total\ hours\ deviation\ (€) = (Estimate\ hours - dedicated\ hours)$$

The deviation in terms of the cost of the personnel employed can also be obtained in the following way:

$$Total\ PCA\ deviation\ (€) = (Estimated\ PCA\ cost - Real\ PCA\ cost)$$

The deviation in terms of the generic costs can be obtained in the following way:

$$Total\ GC\ deviation\ (€) = (Estimated\ GC\ cost - Real\ GC\ cost)$$

Finally, to see if the project has spent more than the initially planned budget, the contingency deviation will be calculated.

$$Total\ contingency\ deviation\ (€) = (Estimated\ contingency\ cost - Cost\ of\ contingencies)$$

6. Sustainability

The following chapter provides a report on the sustainability of this project, on three different dimensions: the economic dimension, the environmental dimension, and the social dimension.

6.1 Self-evaluation

Thanks to the survey I have realized that, despite having heard on many occasions the problems we currently have in terms of sustainability, few times we have been told about the possible solutions. After answering the questions I have seen that at this point I would not know how to solve many of the problems that I can find towards the social sustainability of my projects. In addition, I have also learned that there are many indicators to evaluate the different dimensions of sustainability of a project.

Given the nature of my final degree thesis, environmental sustainability is already taken into account, but I have realized that I cannot neglect the other two dimensions of sustainability. Making a project more sustainable in all three dimensions should be one of the goals to be pursued by all students and professionals who may make decisions regarding products and services with an impact on our society.

6.2 Economic dimension

About PPP: Have you estimated the cost of the project (human and material resources)?

In the previous section, the reader can find an estimate of the monetary impact of the project, considering material and human resources expenses, also including costs in respect to deviations (contingencies and unforeseen costs).

On life cycle: How is the problem you want to address currently solved (state of the art)? How will your solution economically improve on existing solutions?

Currently, the lack of research into the energy consumption of deploying a machine learning system means that other researchers and practitioners do not know the optimal configuration. One of the current solutions may be to try to reduce energy consumption by training models more efficiently, reducing the time it may take for a model to make a prediction. Another way to solve the problem is to spend many hours searching for a configuration that can be moderately efficient for a given study.

My solution can economically improve existing solutions by finding a generic optimal configuration within a particular domain. By generating a generic configuration, other researchers and practitioners will be able to build on it to carry out their studies, saving many hours of study time.

In relation to the PPP, have you quantified the cost (human and material resources) of carrying out the project? What decisions have you taken to reduce the cost? Have you quantified these savings?

In section 12 of this document, the deviation in terms of budget is described.

Although the personnel cost is taken into account in the budget, in reality all of the work has been done by a single person, the author of this thesis. This means that the costs related to human resources is inexistent. Regarding the material costs, the amortization for the laptop and the monitor used has been of 316,69 euros (see Table 20)

In relation to the PPP, has the expected cost been adjusted to the final cost? Have you justified the differences (lessons learned)?

As the estimated cost took into account the hiring of different professionals to perform the role that the author of the project ended up developing, those weren't really necessary in the end. One cost that wasn't planned and that ended up being necessary was the payment of the Heroku subscription for the three months of the project development.

In relation to the useful life, what cost do you estimate the project will have during its useful life? Could this cost be reduced to make it more viable?

In the hypothetical case that future researchers continue this study with more cloud platforms, the cost of those additional cloud platforms would be an extra to the costs estimated for this project, not taking into account that probably there would be a need to actually hire the roles specified in this project. The cost could be reduced by following the same principles as this project and going for the cheapest options available from the chosen providers

In relation to the useful life, has the cost of adjustments/upgrades/repairs during the useful life of the project been taken into account? / repairs during the useful life of the project?

As the project upgrades depend on the purpose of the future study, these cannot be properly estimated, as these upgrades can vary greatly from one project to another.

In relation to risks, could there be scenarios that could undermine the viability of the project?

As the purpose of the project is orientated toward research, the system doesn't have a focus on being viable economically.

6.3 Environmental dimension

About PPP: Have you estimated the environmental impact of the project and have you considered minimizing the impact, e.g. by reusing resources?

Since the main objective of the project is to study the environmental impact under certain parameters, it can be considered that the environmental impact of the project has been estimated. In order to minimize the impact of the project, it has been decided to reuse resources from the study area, in this case using already trained models.

On life cycle: How will your solution be environmentally better than existing solutions?

As the aim of the project is to find a deployment configuration that is energy efficient while maintaining a standard of quality and meeting certain requirements, it can be considered that the environmental improvement would be the discovery of this optimal configuration.

In relation to the PPP, have you quantified the environmental impact of carrying out the project? What measures have you taken to reduce the impact? Have you quantified this reduction?

Yes, as the purpose of this thesis is to measure and examine the environmental of a certain configuration of software tools, the environmental impact of carrying out the project can be seen throughout this document.

In relation to the PPP, if you were to do the project again, could you do it with fewer resources?

No, as the project already uses a small number of resources, reducing these is not possible.

In relation to the life cycle, what resources do you estimate will be used during the useful life of the project? What will be the environmental impact of these resources?

The resources used in the life cycle of this project would be the machines where the application is deployed further in the future. The environmental impact of these resources will vary from different providers, as each one has its machines in different geographical locations and the energy grid of each region makes the environmental impact different

In relation to life cycle, will the project reduce the use of other resources? Overall, will the use of the project improve or worsen the ecological footprint?

Hopefully, the results of this project raise awareness about the emissions generated by popular cloud hosting platforms and more developers and researchers consider the decisions in regards to the hosting they use in their systems.

6.4 Social dimension

About PPP: What do you think the realization of this project will bring to you personally?

First of all, carrying out this project will give me knowledge on how to carry out a research project and I will be able to learn how to manage them correctly. On the other hand, I believe that I will acquire a technical organizational improvement and I will be able to better plan my future projects. Finally, being a project where I am not familiar with the technologies in question, this work will allow me to learn about them and whether in the future it could be a branch that I would like to continue studying.

On life cycle: How will your solution be socially better than the existing ones?

By exposing an optimal deployment configuration, it may be useful for students who do not have as many resources and want to perform similar experiments in the future. This will allow them to spend more time on other important tasks instead of searching for an optimal solution.

On life cycle: Is there a real need for the project?

As discussed in the contextualization section, today's machine-learning solutions seek accuracy above all else. This will mean that future solutions will continue to pursue these objectives and, therefore, will still cause the same problems as the solutions that exist today. Finding an energetic solution to a part of the systems using these technologies can be very helpful to improve the environmental sustainability of these studies.

In relation to the PPP, the realization of this project has implied significant reflections at a personal, professional or ethical level on the part of the people involved?

Yes, mainly on an ethical level. Realizing that the knowledge available to the community is so limited in terms of knowing how your application's energy is truly created and the impact that running your applications has on the environment has made me reflect on how much there is left to improve. Although some information is provided on certain platforms, in general users are quite blind to the impact they provoke on the environment.

In relation to the useful life, who will benefit from the use of the project? Is there any group that could be harmed by the project? To what extent?

The main beneficiaries would be the community of researchers and developers. As the project intends to improve the environmental impact of computing, not many groups could be harmed by this project, maybe companies that benefit from the use of fossil fuels.

In relation to the useful life, to what extent does the project solve the problem initially raised?

Completely, the results of the project provide insight into the environmental impact of the chosen configurations.

In relation to risks, could scenarios occur that would make the project harmful to a particular segment of the population?

No, as the purpose of the project is to improve the current environmental situation

In relation to risks, could the project create some kind of dependency that would leave users in a weak position?

This project doesn’t offer any kind of service which could be harmful to any kind of user.

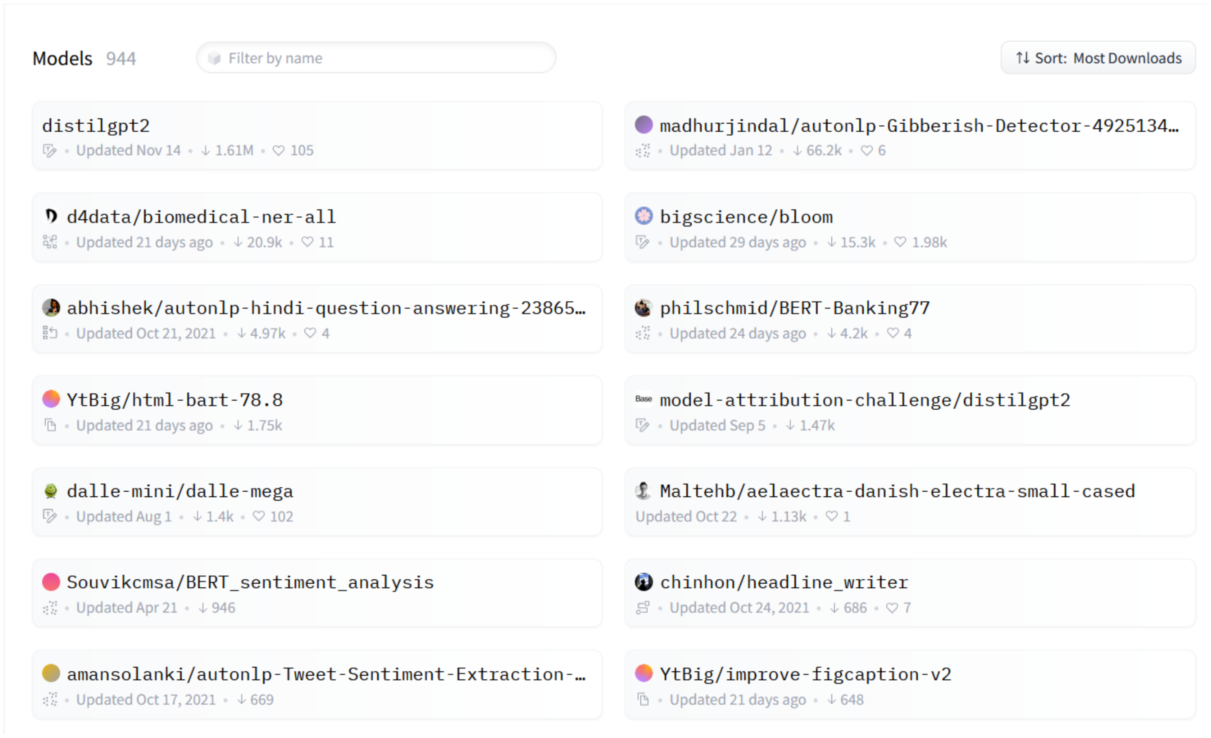
7. ML deployment (SO1)

The following chapter describes the deployment process followed in this project. Firstly, the development evolution of the deployed application is described and afterwards, the chosen deployment platforms are listed

7.1 Hugging Face model selection

Before deploying our systems onto several deployment platforms, we first need to select a model to start developing the application to launch. The selection criteria for this process were unclear at first, given that the lack of experience was a limiting factor to assess the requirements properly. At first, the main criteria to select these models were that the model itself was highly rated and downloaded on the huggingface model repository, to have some sort of validation that the model was valuable and properly developed, and also that the model presented information about the carbon footprint generated by its training.

To begin with the experiments the type of model selected was an NLP model, as they were easier to set up and test for someone unfamiliar with machine learning and also because the tasks to perform were straightforward, which would simplify the experiments to be performed later on. Following these, the selected model at first was DistilGPT2 [20].


Models 944 | Filter by name | Sort: Most Downloads

distilgpt2 · Updated Nov 14 · 1.61M · 105
madhurjindal/autonlp-Gibberish-Detector-4925134… · Updated Jan 12 · 66.2k · 6
d4data/biomedical-ner-all · Updated 21 days ago · 20.9k · 11
bigscience/bloom · Updated 29 days ago · 15.3k · 1.98k
abhishek/autonlp-hindi-question-answering-23865… · Updated Oct 21, 2021 · 4.97k · 4
philschmid/BERT-Banking77 · Updated 24 days ago · 4.2k · 4
YtBig/html-bart-78.8 · Updated 21 days ago · 1.75k
model-attribution-challenge/distilgpt2 · Updated Sep 5 · 1.47k
dalle-mini/dalle-mega · Updated Aug 1 · 1.4k · 102
Maltehb/aelaectra-danish-electra-small-cased · Updated Oct 22 · 1.13k · 1
Souvikcmsa/BERT_sentiment_analysis · Updated Apr 21 · 946
chinhon/headline_writer · Updated Oct 24, 2021 · 686 · 7
amansolanki/autonlp-Tweet-Sentiment-Extraction-… · Updated Oct 17, 2021 · 669
YtBig/improve-figcaption-v2 · Updated 21 days ago · 648

Figure 4: Top 14 models sorted by downloads and filtered by reporting of carbon emissions in huggingface.co

At first, this looked like a good choice but after beginning to deploy the application, its size proved to be a limiting factor for the deployment. Because of this, and given that other models that fit some of the previously established criteria failed to accomplish the other, these criteria changed, replacing the necessity of presented carbon emission data for a smaller model size. By maintaining the criterion of the model necessarily being one with a high number of downloads, if needed, the model emissions can be calculated using hugging face-supported emissions calculator [21].
After these changes, the selected model was Google's t5-small. By using this, we obtained a model small enough to run on the majority of deployment platforms (as they often share similar specs in regard to their machines on the free tier, more on that later).

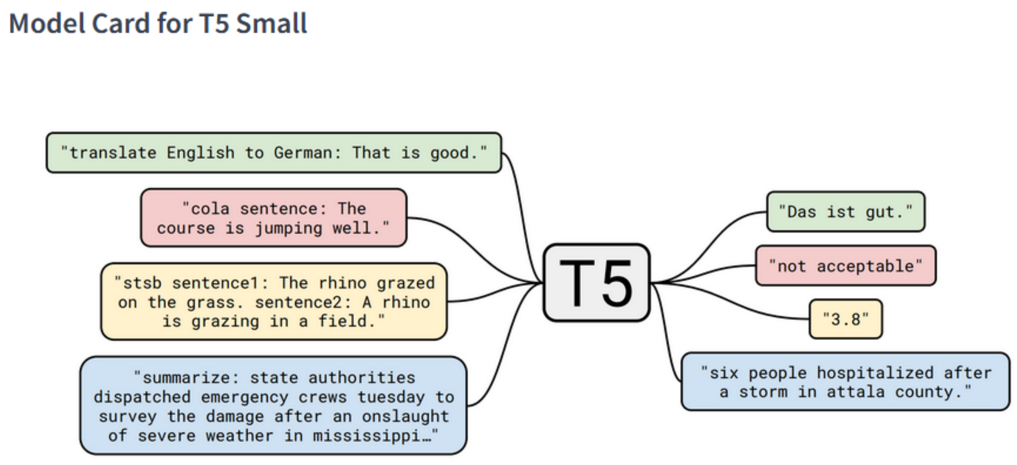

Model Card for T5 Small
"translate English to German: That is good."
"cola sentence: The course is jumping well."
"stsb sentence1: The rhino grazed on the grass. sentence2: A rhino is grazing in a field."
"summarize: state authorities dispatched emergency crews tuesday to survey the damage after an onslaught of severe weather in mississippi…"
T5
"Das ist gut."
"not acceptable"
"3.8"
"six people hospitalized after a storm in attala county."

Figure 5: Diagram showing the main usages of the t5-small model.

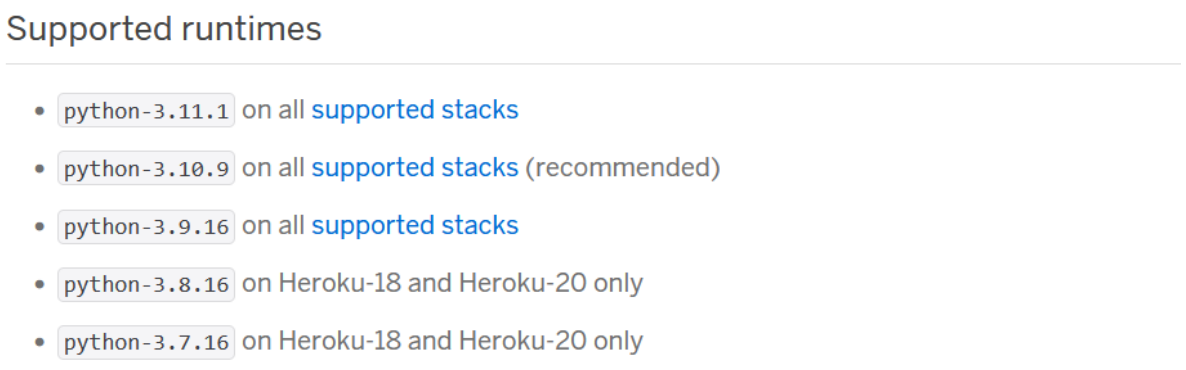
7.2 Application to deploy

After selecting a model, we need to design the application to be deployed. As the main purpose of these experiments is to test the inference emissions of the model, we need to make the application as simple as possible, avoiding external overhead affecting the calculations. Because of this, the selected design for the program is to use a python framework to facilitate the creation of an API which will be called to test the inference of the model. The chosen framework was FastAPI as it provides a small learning curve for beginners and offers an easy setup, along with powerful features for input and output validation.

After selecting the framework to use, the chosen structure for the application is rather simple, dividing the program into two parts, the main script, and an inference script. To better isolate the work of the model, the model inference is performed separately from the API, avoiding the initialization consumption that could interfere with the power calculation.

7.3 Containerization of the app

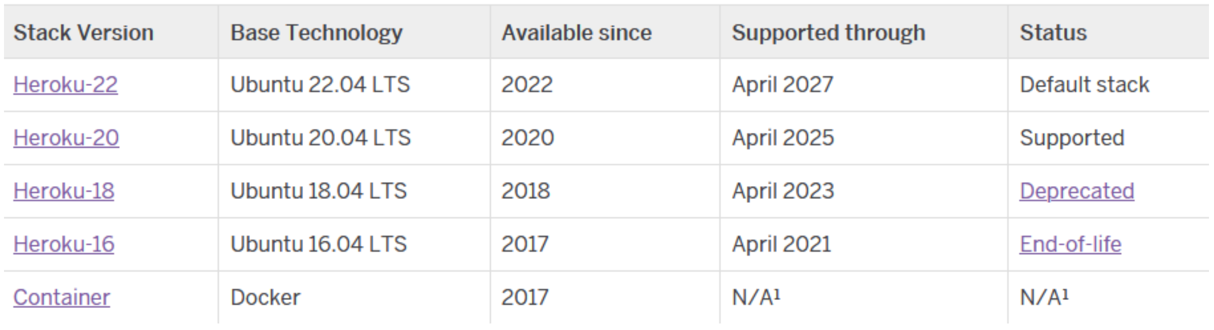
After researching different cloud platforms, one thing that needed to be addressed was the difference in production environments once the app was deployed. For example, Heroku only supports certain python versions [22] for different Heroku versions (the Heroku-22 stack only provides support to python 3.11.0, 3.10.8, and 3.9.15), which can prove troublesome in regards to matching with other constraints, such as package requirements.



Supported runtimes

- `python-3.11.1` on all supported stacks
- `python-3.10.9` on all supported stacks (recommended)
- `python-3.9.16` on all supported stacks
- `python-3.8.16` on Heroku-18 and Heroku-20 only
- `python-3.7.16` on Heroku-18 and Heroku-20 only

Figure 6: Supported python runtimes for different Heroku stacks



Stack Version	Base Technology	Available since	Supported through	Status
Heroku-22	Ubuntu 22.04 LTS	2022	April 2027	Default stack
Heroku-20	Ubuntu 20.04 LTS	2020	April 2025	Supported
Heroku-18	Ubuntu 18.04 LTS	2018	April 2023	Deprecated
Heroku-16	Ubuntu 16.04 LTS	2017	April 2021	End-of-life
Container	Docker	2017	N/A[1]	N/A[1]

Figure 7: Different Heroku stacks

To address this issue, wrapping the application in a Docker container was the selected solution. By using a Docker container, the app is ready to be deployed to different cloud providers.

Besides this main benefit, by using Docker the project facilitates the reproducibility and repeatability of the experiments, providing a template environment for future researchers.

On the other hand, by using a Docker image the researcher becomes responsible for selecting and maintaining the base layer on which containers are deployed, something to keep in mind when scaling the experiment.

7.4 Research criteria for providers

As these will be the subject of comparison, a previous exhaustive study of the available cloud providers is needed to choose the best study subjects.

As mentioned in the previous section dedicated to the objectives of the project, one of the objectives of this study is to produce an experiment that can be easily replicated and/or expanded upon. Because of this, one of the main criteria to have in mind when looking for platforms to deploy our model is the availability of low-end tiers. Meaning that these providers should offer their services for a lower price or, ideally, for free, at the expense of providing less powerful machines.

Besides offering these low-end tiers, another important feature required is compatibility with Docker applications. As our model will be deployed on a Docker image, these providers should offer proper integration between the machine which will run the application and the system which will store the image that contains the app.

Luckily[23], a considerable number of state-of-the-art providers fulfill the previous expectations. Even if a majority of the providers match the requirements, providing free services with appropriate Docker support, the free services usually have a limit on usage. This means that running our application on their machines without appropriate monitoring of the uptime of the deployments and the requests performed onto these deploys could easily consume all of the free resources, and in consequence require payment from the researcher.

In the sections below we will list the selected cloud providers, explaining how they meet the requirements .

7.5 Chosen providers

7.5.1 Amazon Web Services

As the largest cloud provider in terms of market share [24] and one of the pioneers in cloud computing, AWS needs to be in the comparison. Offering a myriad of over 200 services, AWS presents the ones needed to deploy a simple Docker application to test their machine's energy efficiency.

Regarding the requirements mentioned before, AWS offers a vast selection of machines one can use to deploy their projects, including some “free” alternatives. These are the t2.micro and the t3.micro, machines which are on the lower end of the hardware specifications of the computer, in comparison to the other options offered. These are free to use, although not without a catch, as this free usage only extends to the first 12 months of using AWS and with a limitation of 750 hours of use per month.

Because of the needs of the application, these instances are good enough, being the t2.micro instance the one selected for the deployment. Even if both instances are similar in performance, with t3 having a slight edge [25], the selected t2.micro comes at a lower price per use, and thus being a more affordable option, something to keep in mind if the experiment is to be reproduced.

On the other hand, as far as Docker compatibility is concerned, AWS offers a registry service (ECR) that also provides a free tier of use. This registry allows storage of up to 500 MB worth of Docker images, a quantity that suffices for the Docker image of the application.

The deployment process to follow can be split into two separate parts: uploading the Docker image to a registry service and generating the container instance inside the machine which will serve as the host for the application.

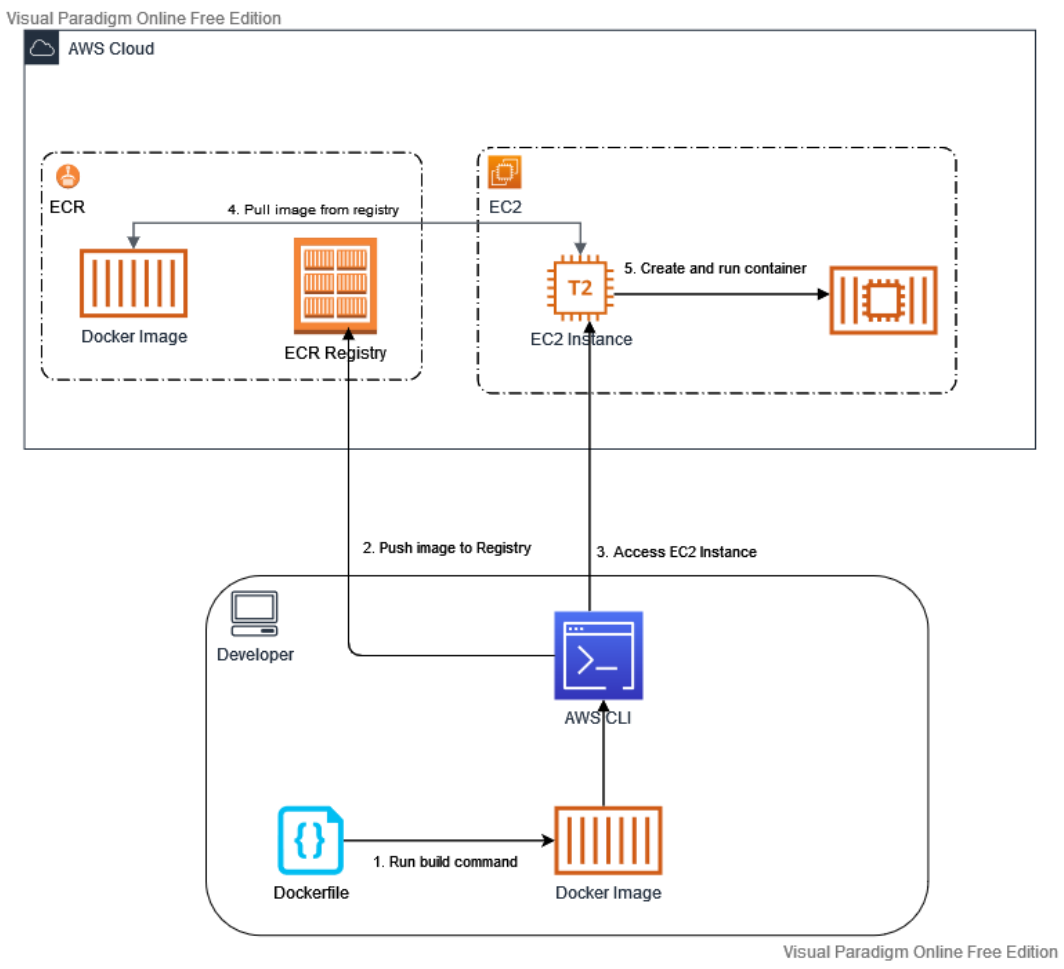

Visual Paradigm Online Free Edition
AWS Cloud
ECR
4. Pull image from registry
Docker Image
ECR Registry
EC2
T2
5. Create and run container
EC2 Instance
2. Push image to Registry
3. Access EC2 Instance
Developer
AWS CLI
1. Run build command
Dockerfile
Docker Image
Visual Paradigm Online Free Edition

Figure 8: Process followed to upload the Docker Image in AWS

For a more detailed explanation of the steps to follow to deploy the application, see Annex 1.

7.5.2 Azure

As the second largest cloud provider, following AWS, the platform from Microsoft also is an interesting study subject. In a similar fashion to AWS, Azure Cloud offers a vast catalog of services one can use for its cloud computing needs, and because of this same reason, Azure contains the services necessary to deploy and run the application.

In a similar way to AWS, the services offered by Azure are available at a free tier, with the slight difference that Azure offers this free option for an indefinite period of time, although with the same monthly limitations regarding usage. This affects the App Service, the equivalent to AWS EC2 instances and the one in charge of running the application, as well as Azure Container Registry, the resource in charge of storing the Docker image.

As per the deployment process, in the same fashion as AWS, the process is split between the upload of the image to a cloud registry and the creation of the container in the instance in charge of running the application. One thing that sets Azure apart from AWS is that in this case, the developer doesn't need to build the image on its local machine. Instead, the image is built by cloning the project containing the source code along with the Dockerfile necessary to build the Docker image. Because of this, it is essential that the source code is uploaded to a code hosting platform, such as Github.

7.5.3 Heroku

A hosting platform known and deeply rooted in the web development community, Heroku offers an easy and ready-to-use environment that allows developers to deploy their code without thinking much about the infrastructure side of their system. Because of the simplicity of its ecosystem, Heroku is better suited for small businesses and independent developers, who don't have a DevOps team who can handle complex infrastructures such as the ones presented by platforms such as Azure or AWS.

Although at the start of this project, Heroku was a platform that fit the requirements of having a free tier of their services and a registry capable of storing Docker images, throughout the development of the project Heroku changed the pricing of their plans, removing the free options. Because of this Heroku fails to meet the first requirement. Even with that, Heroku will be used in the comparison, as its relevance as a smaller cloud provider is noteworthy, along with its acceptance and popularity in the web development community. The selected option out of the Heroku plans is the Eco plan [26]

One peculiarity of Heroku is that once the application container is deployed, Heroku runs this in a special container called *dyno*. This is performed on any kind of software deployed in Heroku, as wrapping the applications in these lightweight containers allows Heroku to provide all of its services, such as the console capable of accessing the deployed instance or the monitoring metrics. Because of this, Heroku's solution also proves to be an interesting case of study, as wrapping the already containerized application could have some interesting effects on the energetic efficiency.

The following figure illustrates how the deployment is handled by Heroku. The second step, where Heroku selects the buildpack to use, and the following step, using the buildpack to transform the source code of an application into a *slug* [27], a compressed copy of the application optimized for the dyno manager, are what makes Heroku interesting from an energetic efficiency point of view.

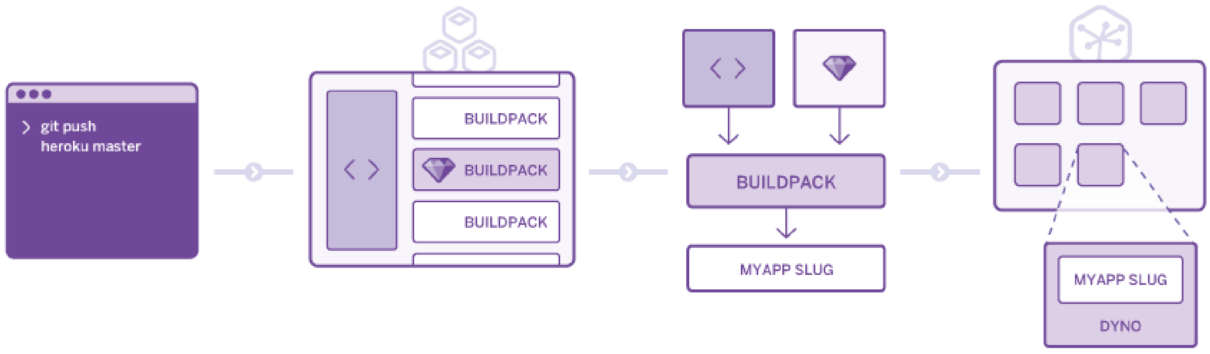


Figure 9: Deployment flow in Heroku

As per the previous providers, Heroku also follows a process of uploading images and deploying them onto the instances.

7.5.4 Railway

Another hosting platform following the guiding principles of Heroku, Railway offers an easy solution for the deployment part of the development lifecycle. As the company emerged as an alternative to Heroku, they are similar in some ways, such as providing a CLI tool for developers, along with built-in databases and GitHub repo deployments.

Nonetheless, Railway differs from Heroku in several ways, such as providing resource-based pricing (thus meeting one of the requirements set before), a stronger focus on support, and a better developer experience in terms of how to handle builds, lifecycle management, etc.

In their pricing plans, they also offer a free monthly quota of credits to use on their service. On the other hand, Railway also presents excellent Docker support, as with their GitHub repo deployment they can also detect whether the repo contains a Dockerfile or not, and in consequence, determine if it is necessary to build a Docker image and run it.

8. Experimental setup (SO2,SO3)

We first present the research questions we aim to answer, then describe the experiment's design. The design description is divided into the variables of the experiment, how the systems will be warmed up, and how the experiment will be executed. Additionally, the tools and tests employed for the data collection and analysis are described.

8.1 Research questions

We break down our goal (see Section 2) into two research questions (RQ):

- **RQ1: Do the free plans of popular cloud providers have a different energy consumption in terms of emissions for the inference of an NLP model?**
  With this RQ we examine the emissions of some of the most popular cloud providers at the moment right now, aiming to empirically assess the possible differences between them.

- **RQ2: Which is the provider that offers the best emissions/latency tradeoff for the inference of an NLP model?**
  With this RQ we examine the tradeoff between the emissions and the latency in the inference of an NLP model, aiming to assess which of the chosen providers offers the best result.

8.2 Experiment design

The experiment will consist of the following: using the API exposed in the application, a certain number of HTTP POST requests will be executed, in an automated fashion and with a certain delay between each request. After all of the requests have been executed, there will be a final HTTP GET request in order to collect the logs containing the energy consumption generated by the previous requests.

Some of the experiment design guidelines have been inspired by the blog post by Luis Cruz [28], such as the warm-up steps or some of the data analysis tests to use.

8.2.1 Variables and statistical hypotheses

The *independent variable* (input) of the measurements is the cloud provider and has four possible treatments: aws (Amazon Web Services), azr (Azure), hrk (Heroku), and rail (Railway).

The *dependent variables* (output) are *energy consumption* (CO_2eq emissions) and *latency* (seconds).

To answer RQ1, we construct the following null hypothesis:

- **Energy consumption:** Considering $\bar{x}$ as the mean of the emissions generated by the cloud provider, the null hypothesis is formulated as follows

$$H_0^{emissions}: \bar{x}_{aws} = \bar{x}_{azr} = \bar{x}_{hrk} = \bar{x}_{rail}$$

The corresponding alternative hypothesis is formulated as:

$$H_1^{emissions}: (\bar{x}_{aws} \neq \bar{x}_{azr}) \vee (\bar{x}_{aws} \neq \bar{x}_{hrk}) \vee (\bar{x}_{aws} \neq \bar{x}_{rail}) \vee (\bar{x}_{azr} \neq \bar{x}_{hrk}) \vee (\bar{x}_{azr} \neq \bar{x}_{rail}) \vee (\bar{x}_{hrk} \neq \bar{x}_{rail})$$

8.2.2 Warm-up

To obtain the data from where we will base the project's results and conclusions, we will be executing a number of API requests to the deployed models in the different platforms.

The temperature of the workstation is an important parameter to manage before performing an experiment aimed to measure energy consumption.
As the machine warms up, the temperature will rise, increasing the resistance of electrical conductors inside the components, leading to higher dissipation and, consequently, higher energy consumption. Because the energy consumption is measured taking into account the computer's underlying hardware, if we begin collecting measurements with a relatively cool machine, these will have less energy consumption than measurements that are retrieved later on.

To mitigate this, before starting with the experiment, we will start the deployed model instance and let it run for 20 minutes, in an attempt to warm up the remote machines. As we do not know the temperature of these machines, the only thing we can do is give some time to the machines and then execute a small number of calls and afterward start our experiments, to ensure that the machines are in the correct state.

Another important parameter that could affect the results obtained is the state of the system. Ideally, all other tasks in the machine would need to be stopped to not interfere with our measurements. Practically this is impossible as all computer systems need at least some tasks running to make the system work properly. Although some other tasks can be stopped, in the context of this study this can't be done, as access to the remote machines is very limited.

8.2.3 Experiment execution

In order to be able to perform statistical hypothesis testing, we need to choose a sample size that contains enough data to ensure that the analysis performed is reliable. The chosen sample size will be a number of 100 API calls. This number of requests considers the possibility of failed executions, making room for these and allowing posterior analysis without needing to repeat the experiment again, being able to discard the errors.

The calls will be performed using data obtained from the WMT'14 English-German [29] data set, more specifically the newstest2014.en file. The data set contains 100 sentences, to be translated into German from English, used in every deployment platform, so as not to introduce variability between experiments. Although the model is capable of translating into other languages such as French or Romanian, the sentences will be translated only to a single language, not to introduce a possible bias in the computational cost of the translating operation, as it may be possible that the difficulty of translating into a language changes.

The requests are executed on a Huawei Matebook D14 (8 GB RAM, AMD Ryzen 5 3500U). The calls are executed under the Windows 10 OS and wired through an Ethernet cable, with a download speed of 91.5 Mb/s and an upload speed of 90.3 Mb/s, and a latency of 10 ms.

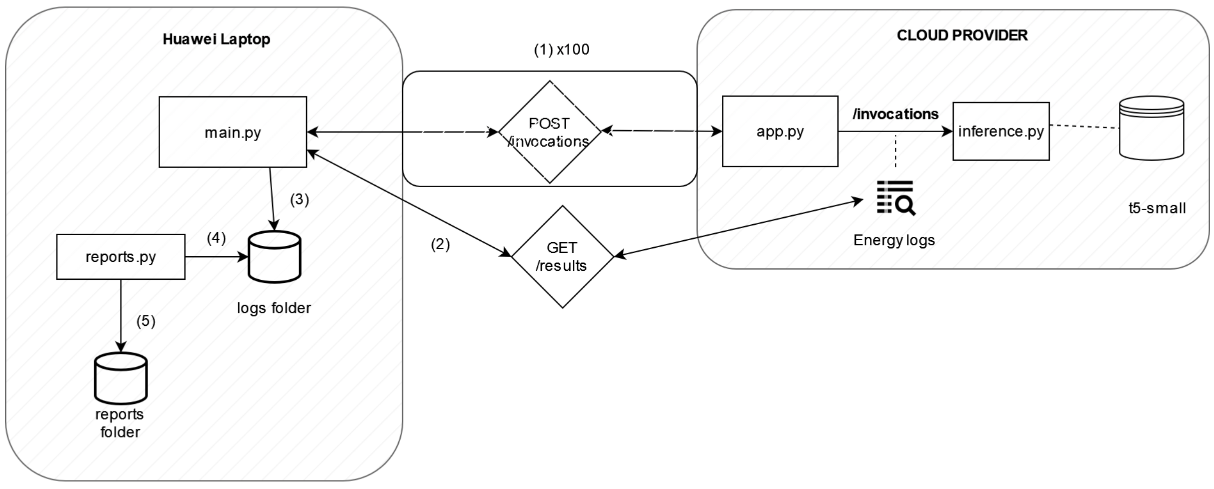


Figure 10 : Overview of the experiment infrastructure

One iteration of the experiment goes through the following steps, pictured in Figure 10:

- 1: The main.py script will begin performing the API POST requests to the deployed API, sending the data to the /invocations endpoint. The API will return the status code for the request.
- 2: After all requests have been performed, the main.py script makes one last GET request to retrieve the energy logs stored in the deployed application. The application will return these in a .csv format.
- 3: The main.py stores the retrieved energy logs in a local folder logs folder. Additionally, the latency (calculated by the main.py script) of each request is also stored along the energy logs.

Additionally, the script used to generate the data analysis (reports.py) figures and tables perform

- 4: Retrieve the energy logs from the logs folder and process them.
- 5: reports.py stores the figures and tables in a reports folder.

8.2.4 Data collection

After running our tests, we need to collect the results generated throughout the experiment runtime. To generate these results we will use a python tool, CodeCarbon.

Developed between four different organizations [30], CodeCarbon intends to provide developers with a tool capable of properly estimating the amount of carbon dioxide emissions produced by the cloud or personal computing machine used to execute a piece of code, specifically ML processes. By using the power consumption of the programs and retrieving information based on the machine's location, CodeCarbon aims to provide solid estimations useful to developers in search of lessening their energy footprint.

To facilitate its adoption by the developer community CodeCarbon was developed as a lightweight python package that is easily integrated into a developer's codebase. It also provides a minimal learning curve, as after installing the package, you can start benefiting from its functionalities by either instantiating an EmissionsTracker object or by placing a decorator on the function you wish to track.

After tracking the corresponding piece of code, CodeCarbon will generate a log containing the emissions generated by the execution of the code, along with other metrics such as the power consumption of different components of the machine, the region where the machine is located, and several others. In this study, only a selected group (see Table 10) will be used in the data analysis, but a full description of all the metrics provided by CodeCarbon can be found on their documentation webpage [31].

CodeCarbon data field	Description
cpu_model	Cpu model used in the machine
cpu_count	Number of CPU
duration	Duration of the compute, in seconds
emissions	Emissions as CO_2-equivalents [CO_2eq], in kg
energy_consumed	Sum of cpu_energy, gpu_energy and ram_energy (kW)
region	Province/State/City where the compute infrastructure is hosted
country_name	Name of the country where the infrastructure is hosted

Table 10: CodeCarbon measurements used in the study

To ensure that the energy metrics provided by CodeCarbon are trustworthy, the CodeCarbon results will be compared against other energy measuring tools, before executing our experiments.

PyJoules provides the user with a toolkit to measure the energy footprint of a program's execution, similar to CodeCarbon. Besides performing a similar task as CodeCarbon, PyJoules also estimates the power consumption by profiting off of Intel CPU's"Running Average Power Limit" (RAPL) feature. Because CodeCarbon also uses this feature to generate its estimations, we can make a correct comparison between the two, ensuring that the result will be meaningful regarding the validity of the data obtained by CodeCarbon.

Even if other energy measuring tools exist (such as Linux perf or RRZE-HPC's likwid), just comparing the tools mentioned above is deemed necessary, as other tools may be troublesome to integrate into the project and a comparison between energy measuring tools lies outside of the scope of this project.

As per the latency measurements, the python requests API is used [32]. The use of this library facilitates the obtention of the elapsed amount of time between sending the request and obtaining the response. As described in the documentation, this property begins measuring the time as soon as the first byte of the request is sent and finishes when the response headers are parsed, not accounting for the time spent consuming the response content[33].

8.2.5 Data analysis

As the researcher has little experience conducting experimental studies, the way to go in terms of deciding how to analyze the collected data is to follow the guidelines set by past studies and researchers. The main references for how to analyze these results are the blog post by Luis Cruz [28] and the previously mentioned study by Hampau et. al. [9], although other studies have also been consulted

Cruz argues that the common tendency for energy consumption measurements is that these follow a normal distribution. He explains that were the results not to follow this distribution, the experiment should be repeated fixing mistakes such as having tasks running in the background, the computer used to perform the experiment entering a different power mode, etc. As this is not possible in the context of this study (as there is no way to access the cloud machines), and given that the only other method Cruz shows to attain normality in the data is to remove outliers from this data (which could be argued that is done in order to fish for specific results), if the data obtained does not follow a normal distribution, other statistical tests will be done. These will be of a nonparametric type.

So, to analyze the outcome of the experiment, the following set of statistical methods are applied.

We begin by performing preliminary observations on the measures using a series of plots to assess the distribution of the data along with performing the Shapiro-Wilks test [34] to determine the normality of the sample. After assessing the normality of the samples, statistical significance tests are performed in order to see if there really is a difference in terms of energy consumption, parametric or non-parametric based on the distribution of our data.

If the dataset follows a normal distribution, a one-way ANOVA test [35] is to be performed in order to determine if there is a significant difference between at least two of the group's mean energy consumption. After this, a Tukey HSD (Honestly Significant Difference) test is performed to examine the differences between each cloud provider's mean energy consumption[36]. And to conclude an effect size test, namely a Cohen's-d test will be performed to understand if the mean difference is small, medium, or large between each group, as the comparisons are performed pairwise (in a similar manner as the ANOVA and Tukey HSD tests)

On the other hand, if the results yield a non-normal distribution, studies like Hampau et. al. [9] suggest using non-parametric tests like Kruskal-Wallis [37] to assess if there is a significant difference in the medians in cases where the independent variable has more than two levels, which fits our study. After performing this test, a Dunn test paired with a Bonferroni [38] correction is done in order to examine the differences between each cloud provider. Finally, a Cliff delta [39] effect size test is used to provide a better understanding of the differences between the cloud providers.

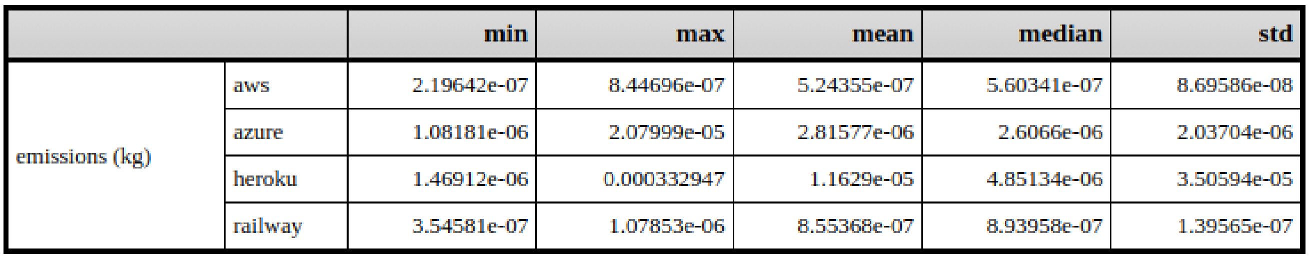
Both the Shapiro-Wilks test and the test that determines if there exists a significant difference (ANOVA or Kruskal-Wallis) will be compared against a significance level of $\alpha = 0.05$

9. Results (SO4)

We first present an overview of the raw measures, and then we present the results of the normality assessment, together with the statistical significance test results. Finally, we will answer the RQs.

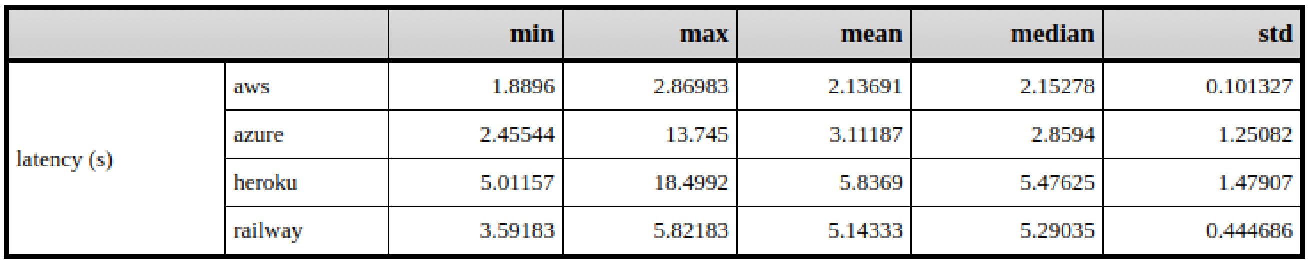
9.1 Overview of collected measures

Tables 11 and 12 provide an overview of the obtained data, following the process described in Section 8.2.3. Specifically, we can look at the median and mean columns to see how each platform performed compared to the others.



		min	max	mean	median	std
emissions (kg)	aws	2.19642e-07	8.44696e-07	5.24355e-07	5.60341e-07	8.69586e-08
	azure	1.08181e-06	2.07999e-05	2.81577e-06	2.6066e-06	2.03704e-06
	heroku	1.46912e-06	0.000332947	1.1629e-05	4.85134e-06	3.50594e-05
	railway	3.54581e-07	1.07853e-06	8.55368e-07	8.93958e-07	1.39565e-07

Table 11: Summary statistics for requests emissions



		min	max	mean	median	std
latency (s)	aws	1.8896	2.86983	2.13691	2.15278	0.101327
	azure	2.45544	13.745	3.11187	2.8594	1.25082
	heroku	5.01157	18.4992	5.8369	5.47625	1.47907
	railway	3.59183	5.82183	5.14333	5.29035	0.444686

Table 12: Summary statistics for requests latency

Looking at the previous tables, some preliminary observations can be made. Starting with the emissions Table 11, the cloud platform which provides the lowest emissions result is AWS and the one with the highest is Heroku.

As per the latency, AWS also seems to yield the best result, providing the lowest mean latency. Mirroring the emissions results, Heroku also provides the worst result in terms of latency

9.2 Results on energy consumption

9.2.1 Normality assessment

After an initial overview of the results, the normality of each provider's results is assessed.

By taking a look at each cloud provider's emissions histogram (see Figure 11), the normality of each distribution can easily be disproven. AWS shows the most "normal" of the four providers, although a lack of measures to the right of the mean is missing to be assessed as normal. Azure and Heroku both show histograms with skewed distributions to the right. On the other hand, Railway also shows skewness, although to the left.

Additionally, by looking at Figure 12, we can see a violin plot comparing each provider. By looking at this figure, it is hard to tell the differences between each provider, as the results from Heroku heavily affect how the other plots are viewed. By looking at Figure 12, the magnitude of the outliers found during the experiment can easily be observed, as the other providers display an almost flat plot in comparison to the Heroku plot.
Because of this, Figure 13 shows each platform's violin plot individually. Each violin plot's width represents the frequency of samples in a specific region, so by looking at each one, the tendency of where the measures reside can be interpreted. Both Heroku and Azure show that there is a tendency for the measures to reside around the median value (white dot in the plot), although both of them show some outliers which are far larger than the median value.

On the other hand, AWS and Railway show similar distributions, as the measures also seem to reside around the median value, but not on the same level as Heroku and Azure, and with some of the measures residing below the median value in a similar pattern. Both show some outliers, although not of the same magnitude as the Heroku and Azure plots.

Besides these histograms and violin plots, Table 13 shows the results of the Shapiro-Wilks test. To disprove that the distribution is normal, the p-values coming from the test are

8.86e-11 for AWS, 1.51e-19 for Azure, 1.19e-20 for Heroku, and 1.82e-8 for Railway. As all of them are lower than the chosen significance level of $\alpha = 0.05$, the data is not normally distributed.

As the normality of the distribution has been disproven, the statistical significance tests to apply have to be of non-parametric behavior. The results of these can be found in the following section.

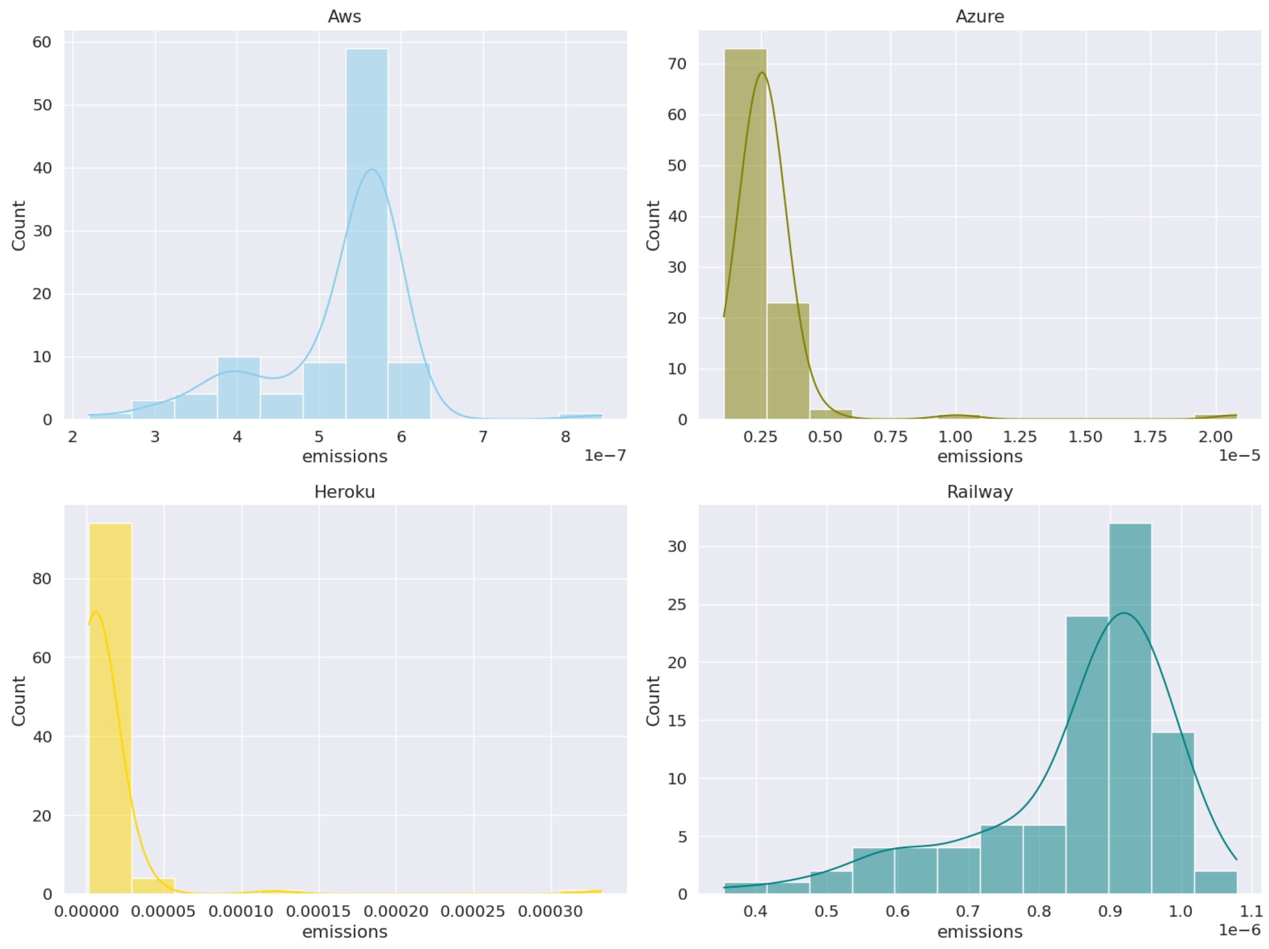


Figure 11: Histograms of each cloud platform. From top to bottom, left to right: aws, azure, heroku, railway

	test_stat	p-value
aws	0.785113	8.86104e-11
azure	0.310005	1.51276e-19
heroku	0.218821	1.19703e-20
railway	0.855043	1.82954e-08

Table 13: Shapiro-Wilks test results for each cloud provider's emissions

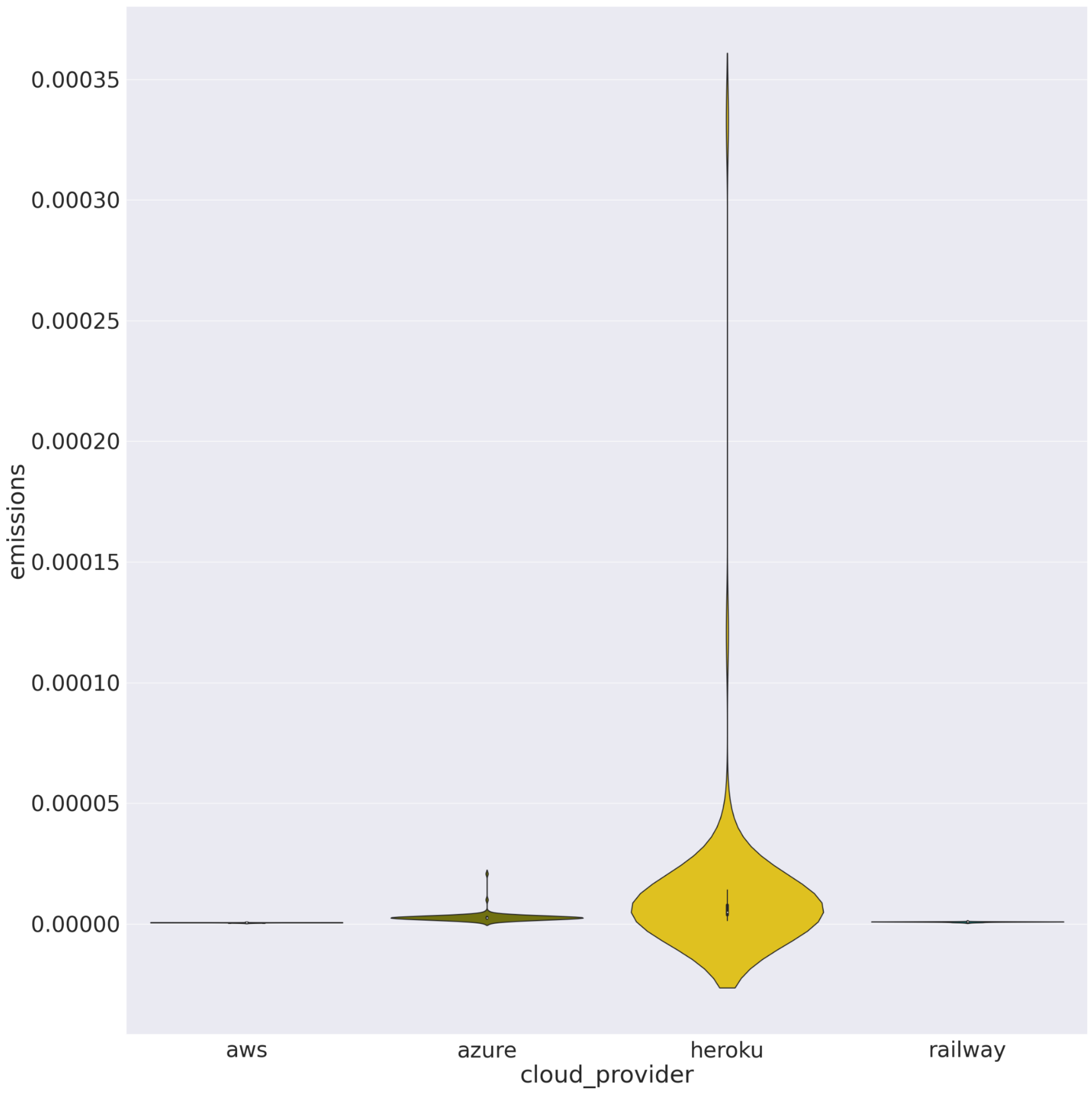


Figure 12: Violin plot of each cloud provider’s emissions

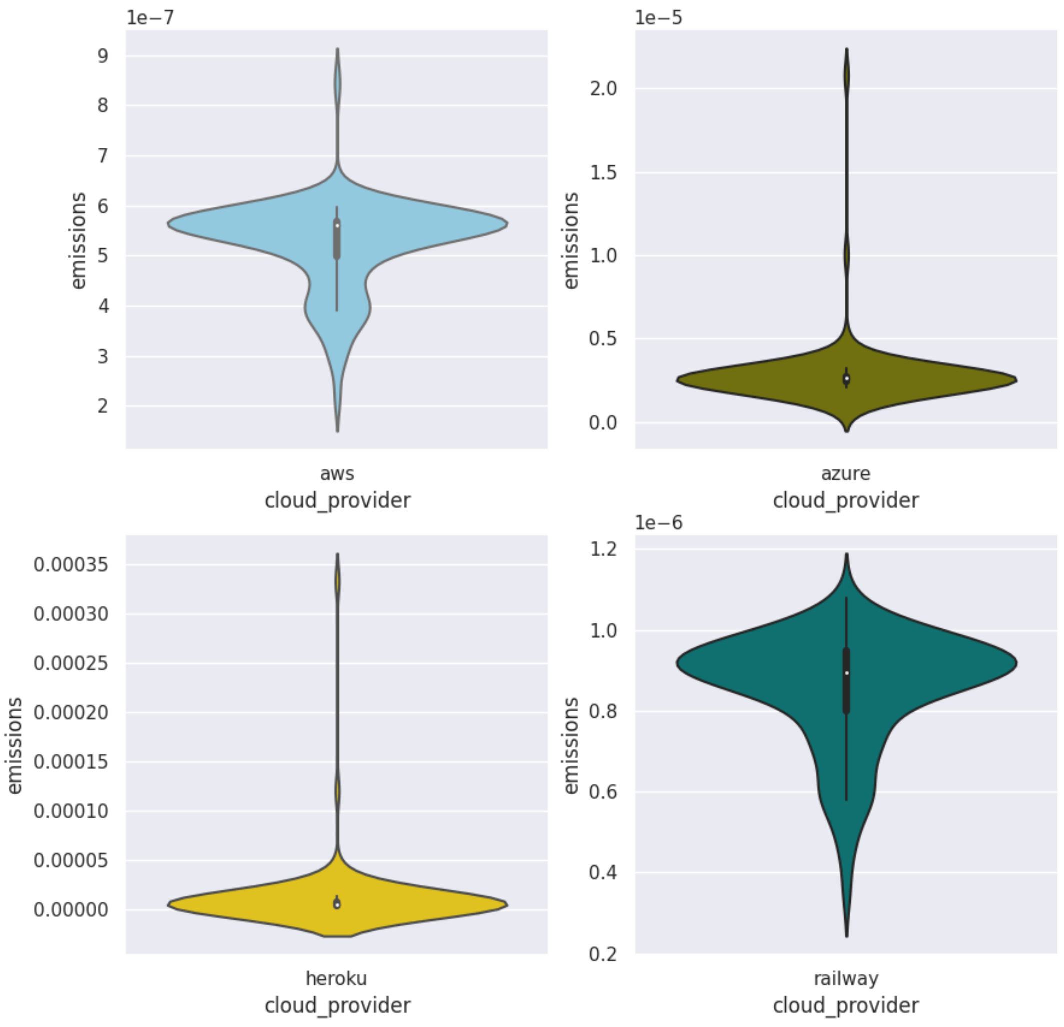
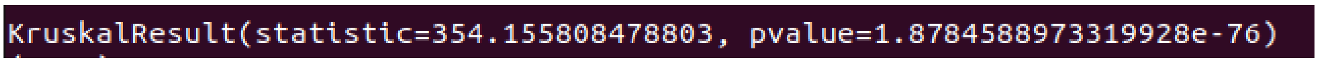


Figure 13: Violin plots for each cloud provider's emissions. From top to bottom, left to right: aws, azure, heroku, railway

9.2.2 Statistical significance tests

To start with the significance testing, the Kruskal-Wallis test needs to be performed in order to assess if there exists a significant difference between the cloud providers.

As seen in Figure 14, the p-value of the test falls below the significance threshold of 0.05, and thus allows to reject the null hypothesis, $H_0^{emissions}$

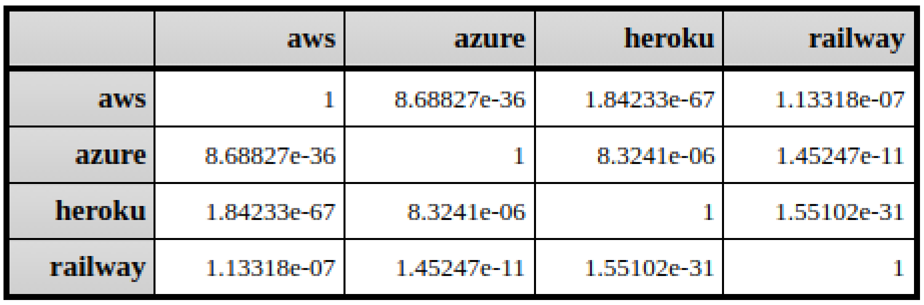


```
KruskalResult(statistic=354.155808478803, pvalue=1.8784588973319928e-76)
```

Figure 14: Results of the Krusk-Wallis test

After confirming that there exists a significant difference between the cloud providers emissions in the results of the experiment, the Dunn test paired with a Bonferroni correction is performed to view which groups are different. Table 14 shows these results.

As seen in Table 14, pairwise comparisons using Dunn's test indicate that each group's scores show statistically significant differences from the other groups. Matching with the observations drawn from the violin plots, AWS and Railway show more similarities in terms of distribution (the comparison performed in the Dunn test) as their difference is smaller than with the other two providers. The same can be said for Heroku and Azure



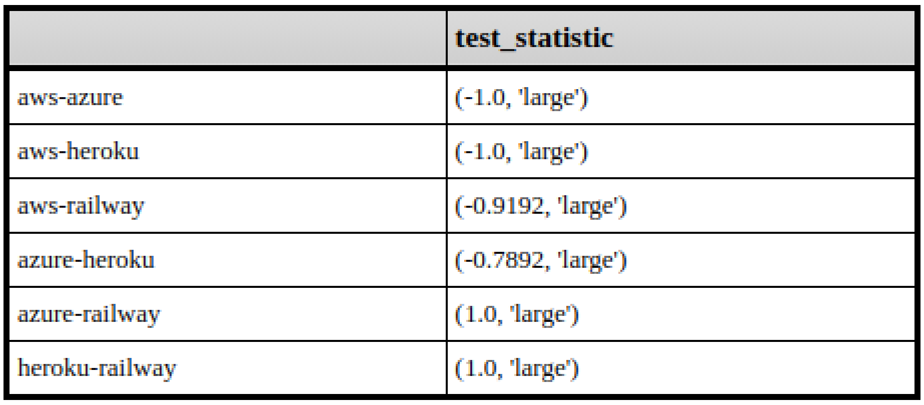
	aws	azure	heroku	railway
aws	1	8.68827e-36	1.84233e-67	1.13318e-07
azure	8.68827e-36	1	8.3241e-06	1.45247e-11
heroku	1.84233e-67	8.3241e-06	1	1.55102e-31
railway	1.13318e-07	1.45247e-11	1.55102e-31	1

Table 14: Dunn test with Bonferroni correction results

9.2.3 Effect size estimation

Finally, a complementary analysis is done by performing an effect size estimation using Cliff’s delta effect size test. As the python package already provides the effect magnitudes values after performing the calculations, there is no need to use others such as Hess and Kromrey’s values [40].

Table 15 presents the interpretation of the results based on the obtained values from the test computation. Negative values for the test statistic imply that, in general, samples from the distribution on the left member of the pair had lower values.



	test_statistic
aws-azure	(-1.0, 'large')
aws-heroku	(-1.0, 'large')
aws-railway	(-0.9192, 'large')
azure-heroku	(-0.7892, 'large')
azure-railway	(1.0, 'large')
heroku-railway	(1.0, 'large')

Table 15: Cliff delta effect size test results for each cloud provider pair

9.2 Answer to RQ1

After performing the significance tests and examining the plots, the null hypothesis can be disproven, and the existence of a difference between the cloud's provider emissions can be assured. To compare the results we can use the following equation:

$$p \, / \, p_{min}$$

Where p is the mean CO_2eq emissions of a cloud provider and p_{min} is the minimum mean value for CO_2eq emissions of all of the platforms. By applying the equation with AWS as the lowest mean emissions (see Table 11), Azure emits 5x more emissions than AWS, Heroku emits 22x more emissions than AWS and Railway only emits 1.6x more emissions than AWS.

Answer to RQ1: We find that there exists a difference in the CO_2eq emissions of the cloud providers selected for this study. More specifically, AWS shows the best results in terms of kg of CO_2eq emissions and Heroku shows the worst results.

9.3 Results on energy tradeoff

The following abbreviations are used in Table 16:

- **cpu_m**: cpu model.
- **cpu_c**: cpu count.
- **reg:** region.
- **cntr**: country.
- **em:** mean of the emissions.
- **en:** mean energy consumed.
- **comp:** mean of the computation time.
- **lat:** mean of the latency time.

prov	cpu_m	cpu_c	reg	cntr	em (kg)	en(kW)	comp (s)	lat (s)
aws	Intel(R) Xeon(R) CPU E5-2686 v4 @ 2.30GHz	1	paris	France	5.24355e-07	4.99386e-06	0.419905	2.13691
azure	Intel(R) Xeon(R) CPU E5-2673 v4 @ 2.30GHz	2	north holland	Netherlands	2.81577e-06	8.85461e-06	0.738743	3.11187
heroku	Intel(R) Xeon(R) Platinum 8259CL CPU @ 2.50GHz	8	leinster	Ireland	1.1629e-05	3.96893e-05	1.11405	5.8369
railway	Intel(R) Xeon(R) CPU @ 2.20GHz	32	Oregon	USA	8.55368e-07	6.63076e-05	2.75306	5.14333

Table 16: Summary statistics with hardware specifications and location

After answering RQ1, to examine which provider offers the best trade-off between the emissions and the latency we need to look again at Table 16. As AWS has shown the best results in terms of emissions and also displays the lowest mean and median values in terms of latency, the provider which ends up offering the best relation between emissions and latency is AWS. Similarly to the emissions results, Heroku also offers the worst results in terms of latency.

9.4 Answer to RQ2

By using the same equation that was used to answer the first research question, we can compare the results in terms of latency:

$$l / l_{min}$$

Being l the measurement of the mean latency of a cloud provider and l_{min} being the minimum mean value for all the latencies of the cloud providers. By applying the equation with AWS as the lowest mean latency (see Table 13), Azure is 1.4x slower than AWS, Heroku is 2.7x slower than AWS and Railway is 2.4x slower than AWS.

Answer to RQ2: The collected results of emissions and latency suggest that AWS offers the best emissions/latency trade-off and that Heroku is the worst cloud provider of the selected options.

10. Discussion

10.1 Discussion

On CO_2eq emissions: Starting by taking a closer look at the emissions results, the results obtained happen to be quite surprising.

Although initially, the prediction for the results was that the bigger cloud platforms should end up being the most efficient in terms of emissions , Azure ended up presenting worse results than Railway. After examining the energy logs obtained from Railway, the cloud provider column indicated that the machines were actually provided by gcp (Google Cloud Platform) and the hosting location resided in Oregon, USA. This actually proved to be in favor of the results provided by Railway as the CodeCarbon documentation offers some references as to from where they base their emissions calculations.[41, 42] From these reference pages, the Google Cloud Platform Oregon region is one of the best in terms of carbon intensity (providing a 12.9 value) and this is reflected in the obtained results.

On the other hand, Heroku yields the worst results and by examining the previously mentioned reference pages, this comes as no surprise as the carbon intensity of the Leinster region in Ireland is 617 g CO_2eq/kWh. By looking at the cloud providers that have machines located in this region, Azure is the only one which is available and as Heroku doesn't provide a value in the column that indicates the name of the cloud provider, one could think that this is Azure. By taking a look at the OS column something strange pops up as the distribution used by Heroku happens to be one containing an AWS tag in its name. Maybe Heroku just happens to be using a Linux distribution based on the ones that AWS uses on their machines, but as Heroku doesn't offer any information regarding their usage of external cloud providers some clarification is needed in order to avoid drawing incorrect assumptions.

On Energy Consumption: By taking a look at the mean energy consumed by the cloud platforms some conclusions can be drawn. By taking a look at Table 16, one can see the mean in terms of energy consumption for all providers. By applying the previously used equation to compare the means, we can see how little is the difference.

$$en / en_{min}$$

Being en the measurement of mean energy consumption and en_{min} the lowest value of mean energy consumption. Using AWS as the minimum value, Azure shows that it only consumes 1.77x more than AWS, Heroku consumes 7.95x more energy than AWS, and Railway consumes 13.2x more energy than AWS. By looking at Table 16, these results shouldn't be surprising as the CPU count for each provider increases as the mean energy consumption increases. Although this seems to provide some reasoning for the results, it seems unlikely that a small cloud hosting platform like Railway would provide the computation power of 32 CPUs at a free tier for a large number of users. As CodeCarbon doesn't specify if that is the number of CPU cores used or simply the number which is available in the machine[43], a user can only speculate on the true meaning of that number. The same can be said for Heroku, as 8 CPUs seem a little too much for the most economic solution, even if the user is paying for the service.

On Carbon Intensity and its impact on emissions: After performing all comparisons on the measurements taken, one can observe how truly important the carbon intensity of the consumed electricity is. Although Railway had the worst results by far in terms of energy consumption, its results in terms of real CO_2eq emissions were just slightly worse than AWS, only 1.6x worse at that. Azure although just consuming 1.77x more energy than AWS, ended up emitting 5x more emissions than Amazon's solution. Heroku, after emitting 22x what AWS emits, only consumes 7.95x more energy.

These results show that the truly important part when it comes to reducing carbon emissions is selecting a proper infrastructure that utilizes a cleaner energy grid, which bases its energy consumption on energies with low-carbon impact or renewable energies. Doing this can prove challenging, as not all providers offer detailed documentation on where their cloud regions are deployed and what the carbon intensity is like in that area (unlike Google Cloud Platform which offers thorough documentation [44]). Some don't even provide any information at all about where their machines are located, like Heroku or Railway which we only found out after performing the experiments and obtaining the CodeCarbon results.

On the trade-off between Emissions and Latency: When it comes to the trade-off between the emissions and the latency of a specific cloud provider, as the best option in terms of emissions also happens to be the best in terms of latency, developers shouldn't have to compromise one quality for the other. As the study has been restricted to the region where the researcher resides, a single option has ended up being the best, but if the study was replicated in multiple regions spread out geographically, perhaps the results could vary and tradeoffs between emissions and latency could arise.

Key Takeaways: To summarize, this project offers the following takeaways to researchers and practitioners that use cloud computing hosting platforms:

- The biggest cloud providers out of the selected group offers the best option, both in terms of emissions and latency. As its resources are larger, and its responsibility to society and the environment is bigger, it comes as no surprise that it offers the most sustainable options.
- Improvement in measuring tools is needed to improve the quality of future studies. Although CodeCarbon has proven to be a detailed and satisfactory tool for the context of this project, improvement is in need for the measuring tools, as certain aspects of the measurements provided by these tools can be uncertain to developers
- Cloud providers should provide better documentation on their energy footprint, especially smaller ones. Although the bigger providers in this study offer some information regarding their energy footprint, this has proven to be lacking in certain aspects, such as providing detailed information on how each region handles its carbon intensity. The smaller providers also should improve their documentation on the sustainability of their solutions, as it is almost non-existent.
- As for the emissions generated by applications, the carbon intensity of the underlying energy infrastructure appears to be more impactful than the energy consumption of the application. One of the worst providers in terms of energy consumption in this study ended up being the second-best result in terms of emissions.
- AWS should be employed by developers who are concerned about the emissions of their systems and are capable of running a complex cloud platform.

  The ideal alternative is Railway if, on the other hand, the system or application is of a smaller scale (and, thus, the team responsible is small), as its emissions are slightly worse than AWS and the work required to operate and deploy applications is considerably lower (see Annex 1 for more information).

10.2 Threats to validity/Limitations

Internal Validity: Due to monetary constraints, the used configurations were the most basic and cheap out of the myriad of options available in the providers, both in order to give some equality between the providers (as the options available in smaller providers cant be as powerful as the ones available in the larger ones) and to fit the constraints. The reader should consider that the obtained results may vary significantly on the used CPUs and GPUs used in the machines. Also, to fit the memory constraints imposed by the selected configurations, the version of the ML framework used is one which doesn’t benefit from the computing potential of the GPU (as the python package that benefits from GPU computing is way too large).
However, the aim of the study was to analyze the emissions and latency on the cheapest options available of the chosen providers.
The energy of an application under test can be affected by many factors, such as running background processes, daemons, and so on. This issue is common in studies like this one, but in the context of this study, controlling these is difficult as users have very limited access to the cloud machines they are using to deploy their applications. Although some providers offer more control than others, no additional configuration has been done in order to avoid possible bias in the results obtained. To mitigate the effects of these factors, the only thing which could be done for all providers was to provide a warm-up time to allow the machines to heat up properly, as the temperature of the machine can impact the energy it consumes.

The latency of an HTTP request can also be affected by many factors, but mainly the stability of the network is the deciding factor that determines the latency. To mitigate the possible effects that could impact the latency, the requests are performed by plugging an Ethernet cable into the workstation (avoiding using Wifi which can be more unstable) and stopping all other devices connected to the network when performing the requests of the experiment.

External Validity: Regarding the generalizability of the findings, the results are deeply related to the chosen providers and model. This threat to the external validity of the study can be mitigated in the future by repeating the experiment on more models, specifically of different ML fields like computer vision, and more cloud platforms. In this study, the use of bigger and smaller providers was done in order to reduce the threat to external validity, by trying to provide a relevant and realistic measurement environment.

Reliability: For the reproducibility of this study, a replication package has been developed and made publicly available, as well as the inputs and outputs of the experiments (inside the public replication package). Additionally, in this document, a configuration annex has been provided to set up the providers chosen in this study.

11 Evolution of the agile development

In this chapter of the document, the development of each sprint of work done in this project will be described.

11.1 Sprint 1

This initial sprint's objectives were to get an initial understanding of how to structure the application that needed to be deployed and to start selecting the models to deploy.

For the selection part of this sprint, the first thing to do was to examine the available models in Hugging Face and choose some possible candidates to deploy. Initially, the plan was to first choose an easy model to deploy and after all the deployments on the selected platforms were done, then move on to a second model. This first model needed to be of the NLP field, as the configuration needed to run this type of model is minimal. This model ended up being the one used in the study, although other models were tested(see Section 7)
As for the other part of the sprint, the objective was to think about how the application could be designed in order to be able to perform the future experiments and extract measurements from these. The initial idea was to develop a REST API, which after the deployment of the application could easily be accessed via an API platform such as POSTMAN or directly performing the requests via a command-line interface. Throughout the sprint, this API was developed and properly tested.
Besides these objectives, the search for providers on where to deploy the app also started and Heroku was selected as the first provider on where to deploy the application

11.2 Sprint 2

This second sprint's objectives were to deploy the application in the first chosen provider and to search for more providers on where to deploy the application.

To deploy the application in Heroku, several modifications were performed in order to make the application fit the criteria established by the provider. First of all, the version of the ML framework(PyTorch) was changed to its CPU-only version, as the regular version which utilized GPU was too large for the Heroku machines to handle. The model's configuration was also tweaked using the hugging-face's transformers API to make the model less memory-consuming. After going through this process, the application was finally running on Heroku and could be accessed and used for inference.

After the Heroku deployment, AWS was the second provider chosen and on the same sprint, the deployment was performed. The deployment process onto AWS required that the application was inside a docker container, and this change remained for the rest of the project, as it eased the deployment process for the next providers and provided a controlled environment for the application's dependencies.

After both of the deployments, the deployment onto the third provider was started, but it did not finish in time for the end of the sprint.

11.3 Sprint 3

In this sprint the objective was to finish up the search of the providers on where to deploy and start designing the experiment to perform. As the researcher and the director agreed that deploying another model from another field of ML was not possible due to the time constraints of the project, the selected model ended up being the only one being tested.

After finishing the deployment on the Azure platform, three more providers were selected to deploy the application, Google Cloud Platform, Railway, and IBM Cloud. The first one chosen to begin the deployment was Railway, and after successfully deploying the application, the deployment process started in parallel for both Google Cloud Platform and IBM. The Google Cloud Platform proved to be more challenging than initially assessed and at the end of the sprint, as it was not viable to keep dedicating efforts to the deployment, the provider was discarded. IBM, on the other hand, seemed to fit the criteria established to be a good candidate, but after a more careful examination of the IBM platform, it was discovered that IBM’s solution was not compatible with the requirements of the project.

In parallel with the deployments, the experiment started to be designed. Initially, the idea was to create a database that could store the results of the experiments and using the same database, dashboards could be created to visualize the results. After careful consideration and consulting with the director of the project, this idea was discarded, as it wasn’t in the scope of the project to design this complex infrastructure. In the end, a simpler solution was chosen, by simply returning an energy log of the requests performed in the experiment via the API of the application. This implied that a refactoring of the application was needed and some of the providers would need to be redeployed.

11.4 Sprint 4

In this final sprint the objective was to wrap up the development process and finish up the experiment design.

After redeploying the application containing the endpoint necessary to retrieve the energy logs, the development of the application was finished. The experiment design was also finished, after choosing the data which would be used in the inference of the model and how the latency would be measured.

Besides that, the data analysis tests to use on the results were also researched and studied, in order to properly review the results and give conclusive answers to the research questions.

12. Deviations from initial planning

In this chapter the deviations from the initial planning are described. The effects that these deviations have on the budget of the project will also be described.

12.1 Changes in planning

Regarding the initial planning, the temporal planning for the majority of the tasks has been quite accurate, for the most part. The only deviations from the original planning were in terms of the experimentation and the definition of the statistical tests.
In regards to the experimentation deviations, initially, the experiments were to be performed using an Ethernet cable connected to the researcher's laptop using an adapter which made possible the connection (as the laptop lacked the specific ports to directly connect an Ethernet cable). During the initial tests, the wired connection to the laptop was troublesome, with multiple failed executions due to network connectivity problems. To test which component was faulty, tests were performed using the ethernet cable with a desktop computer. These revealed that the problem was related to the adapter, which was quickly replaced with a new one. These issues ended up costing 4 more hours (taking into account the time spent to go buy another adapter) than expected for the experimentation part of the project.

In relation to the definition of the statistical tests, these took more time than expected because of the wrong assumptions the researcher had regarding how to treat the results obtained from the experiments. After realizing this mistake, and before starting the data analysis, the researcher explored more statistical tests that could cover all of the possible results. This deviation ended up costing 7 more hours than expected.

ID	Name	Estimated time (h)	Real time (h)
EX3	Experimentation	40	44
AC1	Definition of statistical tests	20	27

Table 17: Tasks that deviated from the initial planning estimation

12.2 Changes in budget

Because of the deviations explained above, the total number of hours dedicated to the project has added up to 471 hours. The following sections describe how this increase in hours affects the initial calculations regarding the budget of the project

12.2.1 Changes in PCA

As some tasks deviated from the initial planning, these changes imply that the budget also ended up changing.
Because of the additional 11 hours, the hypothetical PCA is affected, adding 4 more hours to the researcher due to the experimentation task (EX3) and adding 7 more hours to the data analyst due to the definition task (AC1). Table 18 shows the total cost of the personnel after taking into account the additional hours.

Role	Estimated hours	Additional hours	Cost per hour (€)	Total cost (€)
Researcher	205	4	12,66	2645,94
Project manager	60	-	26	1560
DevOps Junior	75	-	21,87	1640.25
Software Tester	75	-	11,6	870
Analyst	45	7	16,53	859.56
Total	460	11	-	7575,75

Table 18: Total cost of the personnel

Role	Estimated cost (€)	Real Cost (€)
Researcher	2595.3	2645,94
Project manager	1560	1560
DevOps Junior	1640.25	1640.25
Software Tester	870	870
Analyst	743.85	859.56
Total	7409.4	7575,75

Table 19: Estimated PCA vs Real PCA

12.2.2 Changes in generic costs

Because of the additional work hours, generic costs such as electricity and the internet used also increased. Table 20 show these increases, using the equations shown in section 5.2.2

The internet cost and the working area cost do not increase as they are monthly costs and the number of additional hours does not involve working more months than expected.

$$Amortitzation\ PC = 700 \times \frac{1}{3} \times \frac{1}{126} \times \frac{1}{3} \times 471 = 290,74$$

$$Amortitzation\ monitor = 125 \times \frac{1}{6} \times \frac{1}{126} \times \frac{1}{3} \times 471 = 25,95$$

$$Total\ amortitzation = 290,74 + 25,95 = 316,69$$

$$Electricity\ bill\ PC = 0.065\ kW \times 471\ hours \times 0,1736\ kW/h = 5,314764$$

$$Electricity\ bill\ monitor = 0.042\ kW \times 471\ hours \times 0,1736\ kW/h = 3,434155$$

$$Total\ electricty\ bill = 5,314764 + 3,434155 = 8,748919$$

Concept	Estimated cost (€)	Estimated hours	Additional hours	Real cost (€)
Amortization	314,29	460	11	316,69
Electricity	8,54	460	11	8,749
Internet	33,5	460	-	33,5
Working area	1750	460	-	1750
Total GC	2106.33			2108,93

Table 20: Estimated generic costs vs Real generic costs

In addition to these adaptations of the generic costs to the additional hours, a cost which was not expected initially was added halfway through the project's development. This cost is the monthly rate of Heroku's Eco plan. This plan was paid for 3 months with a 5 € cost per month. Taking this additional cost into account the generic costs of the project, Table 21 show the total generic costs for the project.

Concept	Cost (€)
Amortization	316,69
Electricity	8,749
Internet	33,5
Working area	1750
Heroku subscription	15
Total GC	2123,93

Table 21: Total generic costs taking into account Heroku subscription

12.2.3 Total budget

To calculate the final budget only the PCA and generic costs will be taken into account, as the contingency margin and risk management cost is a established cost which does not vary when taking into consideration the deviations which happened during the project's evolution.

Activity	Estimated cost (€)	Additional costs	Real cost (€)
PCA	7409.4	166,35	7575,75
GC	2106.33	17,6	2123,93
Total	9699,68	183,95	9699,68

Table 22 : New PCA and GC of the project taking into account the deviations

As seen in Table 22, the additional costs of these deviations from the initial planning amount to 183,95. As the contingeny margin is 1111,41, these costs have been covered completely and the total cost of the project is not affected, proving that the contingency margin was well computed.

12.3 Management control

Using the formulae shown in section 5.5, the deviations of the project can be computed as:

- **Deviation in terms of total hours:**
  $Total\ \ hours\, deviation\, (h)\ =\ (460\ -\ 471)\ =\ -\ 11\, hours$
  An additional 11 hours have been spent.
- **Deviation in PCA:**
  $Total\ \ PCA\, deviation\, (€)\ =\ (7409.4\ -\ 7575,75)\ =\ -\ 166,35\ €$
  An additional 166,35 euros have been spent in the personnel of the project
- **Deviation in generic costs:**
  $Total\ \ GC\, deviation\, (€)\ =\ (2106.33\ -\ 2123,93)\ =\ -\ 17,6\ €$
  An additional 17,6 euros have been spent in the generic costs of the project
- **Deviation in contingency spending:**
  $Total\ \ GC\, deviation\, (€)\ =\ (1111,41\ -\ 183,95)\ =\ 927,46\ €$
  The contingencies of the project did not consume all of the contingency resources, meaning that the project stayed in budget, and ended up saving money.

13. Conclusions

To finish the project, the conclusions of this project are shown, together with how it fits within the specialization of the degree and what future work can be done.

13.1 Project conclusions

In this project, we have conducted a study with a small-scale application to measure the emissions of different cloud hosting platforms, attempting to determine which of the selected offers the best option in this regard. Besides looking at the emissions, we have also measured the latency of the requests performed to these providers, looking for the existence of a trade-off between the emissions and the latency, and in case there was one (which there wasn't), we would have exposed which one offered the best relation.

Performing this study also allowed us to examine the measuring tools available, and by examining these, some flaws were unveiled in terms of lack of proper documentation of some of the output given by the tool. These findings lead us to the documentation given by the providers, which also proved to be lacking and showed that the companies need to work on these aspects of their business.

The results obtained by this project should prove useful for researchers and developers interested in creating systems mindful of their impact on the environment and can serve as the basis of future studies interested in following the tendency of Green AI.

13.2 Knowledge integration

To show the knowledge integration of different disciplines in this project, a list of the subjects of the degree which have been useful for the development of this project is presented.

Operating systems

The deployment on all of the providers has required the use of different CLI based on Linux operating systems. To properly understand these the knowledge of Linux-based systems of this subject has been applied

Probability and statistics

To understand how to properly apply statistical tests and which ones to use, the knowledge gained from this subject has been applied in this project.

Web applications and Services

The creation of this application required the development of a REST API in order to be able to perform HTTP requests for the experiment. The experience and knowledge gained from this subject have been valuable for the success of this project.

Operating systems for Distributed Applications

The inspiration for using cloud providers as study subjects and the decision of using a Docker container to containerize the application come from the knowledge gained from this subject.

Software Project Management

The working methodologies shown in the subject, more specifically the agile methodology, have been applied in the evolution of this project.

Software Engineering Project

The experience gained from this subject has been useful in order to properly managing a small software project such as this one.

Besides the knowledge obtained from the subject in the degree, some of the knowledge needed to develop this project has been acquired in a self-taught way, mainly all the concepts regarding the deployment of cloud providers. Other concepts, such as knowledge of Python programming or Docker containers have been applied from work experience gained by the researcher.

13.3 Competence justification

CES 1.1: To develop, maintain and evaluate complex and/or critical software systems and services

One of the main points of this project has been to develop an application that could be deployed on several cloud providers, each one with its own specific constraints. Developing the application code and maintaining it throughout the evolution of the project and choosing and implementing the appropriate solution which allowed the code to be deployed in multiple deployments environments are tasks that fit this competence and because of that, we consider that this competence has been accomplished successfully.

CES1.2: To solve integration problems in function of the strategies, standards, and available technologies

The deployment on multiple cloud hosting platforms demanded that the application be adaptable to multiple standards imposed by the platforms. After applying a containerization strategy to the application, the code was ready to be deployed in all provides without much difficulty. For this reason, we consider that the competence has been successfully solved.

CES1.3: To identify, evaluate and manage potential risks related to software building which could arise.

After handling all of the problems that arose in the initial deployment in Heroku, the researcher was aware of all the risks that should be taken into account when selecting more providers to further the study. Those were evaluated in each provider and, if they could be managed, the application was deployed there.

CES1.7: To control the quality and design tests in the software production

Through the development of the project, the quality and the testing of the project were approached in a more functional way. After deploying the application on a new provider, a series of measures were taken from inside the console of the provider, controlling the correct functioning of the application. After that, some requests were performed in order to ensure that the application would run properly the experiment.

CES2.1: To define and manage the requirements of a software system.

The requirements initially set for the system were used to establish a baseline from where to develop the application. Those requirements, though, were evolving as the project matured, adapting to the necessities imposed by the strategies and standards used by the cloud platforms and the used tools to measure the metrics.

CES2.2: To design adequate solutions in one or more application domains, using software engineering methods which integrate ethical, social, legal and economical aspects.

As one of the main motivations behind the project was to follow a tendency in software development that calls for better reporting in terms of sustainability and energy efficiency, the solution was designed with this in mind and, thus, gives detailed information about its environmental footprint.

CES3.1: To develop multimedia services and applications.

Initially, the plan was to deploy more than one model into different cloud providers, ideally models from other fields of ML such as computer vision. As the project progressed, we realized that this could not be accomplished in time.

13.4 Future work

Even if the results obtained have been satisfactory and the main goals have been accomplished, there are improvements to be done.

Deployment in more cloud providers

The initial four providers selected provide a good perspective on the available state-of-the-art, but the addition of more providers to the study, and more importantly, different plans and machines inside these is important in order to validate the results obtained from this project.

Use of different models

The selected model in this project was useful in its context, as the researcher had no previous experience working in this field and because of the nature of the model, it was easy to set up and deploy. This, however, limited the results that could be obtained from the study, as restricting the models to those of one specific field (NLP) makes the results obtained more specific and less representative of all of the fields present in ML.

Acknowledgments

I would like to thank the co-directors of this project for giving me the opportunity to work on this project and for the guidance they have provided me throughout the development of this thesis. Thanks to their aid, the project has been molded into what it is today.

I would also like to thank my GEP tutor, Joan Sardà for helping me improve the initial planning of this thesis. This initial planning helped structure the work to be done in the months after GEP.

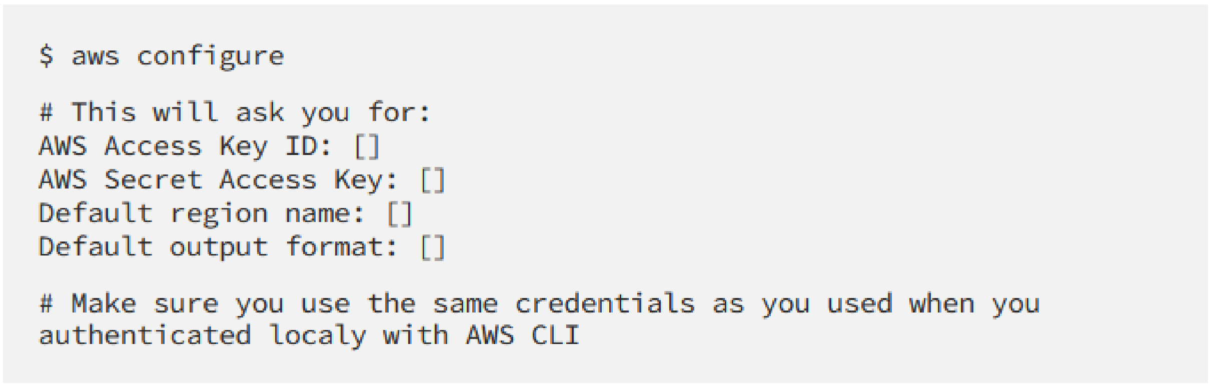
Annex 1: Deployment guidelines

The following annex gives some guidelines on how to deploy an application in the chosen cloud platforms for this project.

Amazon web services

Regarding the deployment process, AWS presents a common methodology applied on some of the other platforms we will see. This deployment process is mainly divided into two parts: the registry and the virtual server.

Before starting with any of these parts, we need to install the AWS CLI on our computer. Using the curl command provided by AWS we can download an installer, which we can later use to install the CLI tools on our machine.
As for the registry, this part involves the creation of a cloud registry where we store our locally generated docker images, pushing them there using docker CLI commands. Before being able to push our docker images onto the registry, we first need to authenticate our AWS credentials on the AWS CLI, previously creating an access key ID and secret access key pair.



```
$ aws configure

# This will ask you for:
AWS Access Key ID: []
AWS Secret Access Key: []
Default region name: []
Default output format: []

# Make sure you use the same credentials as you used when you
authenticated localy with AWS CLI
```

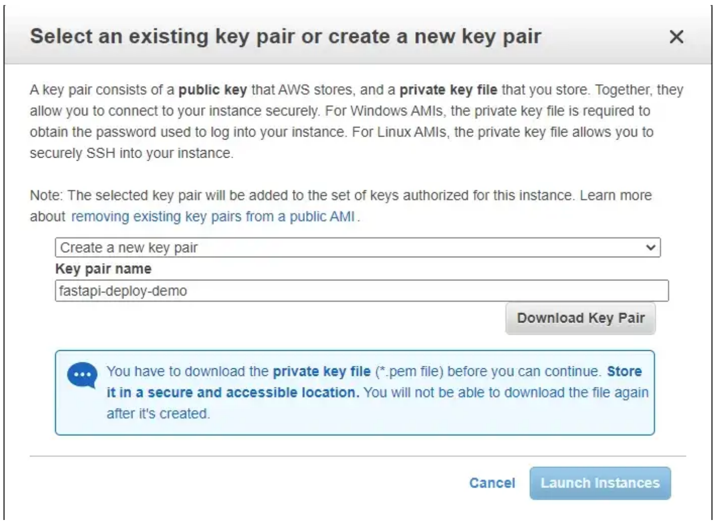
Figure 15: Output of the configure command of the AWS CLI

After doing this, we need to create an AWS ECR instance, i.e. the registry. After doing this, we just need to follow the steps provided by the registry itself to push our docker image on the registry:

- Authenticating the docker client to the registry.
- Creating the image (if it was not created previously)
- Tagging the image to the registry
- Pushing the image to the registry.

As for the virtual server, the first thing we need to do is create the instance of the EC2 server. The configuration dashboard regarding this step is somewhat complex so sticking to the recommended setup provided by Amazon is advised, selecting the free options when possible.

After creating the instance, AWS will ask the user to generate a new key pair needed to establish an ssh connection with the instance and remotely access to it.



Select an existing key pair or create a new key pair

A key pair consists of a **public key** that AWS stores, and a **private key file** that you store. Together, they allow you to connect to your instance securely. For Windows AMIs, the private key file is required to obtain the password used to log into your instance. For Linux AMIs, the private key file allows you to securely SSH into your instance.

Note: The selected key pair will be added to the set of keys authorized for this instance. Learn more about removing existing key pairs from a public AMI.

Create a new key pair

Key pair name

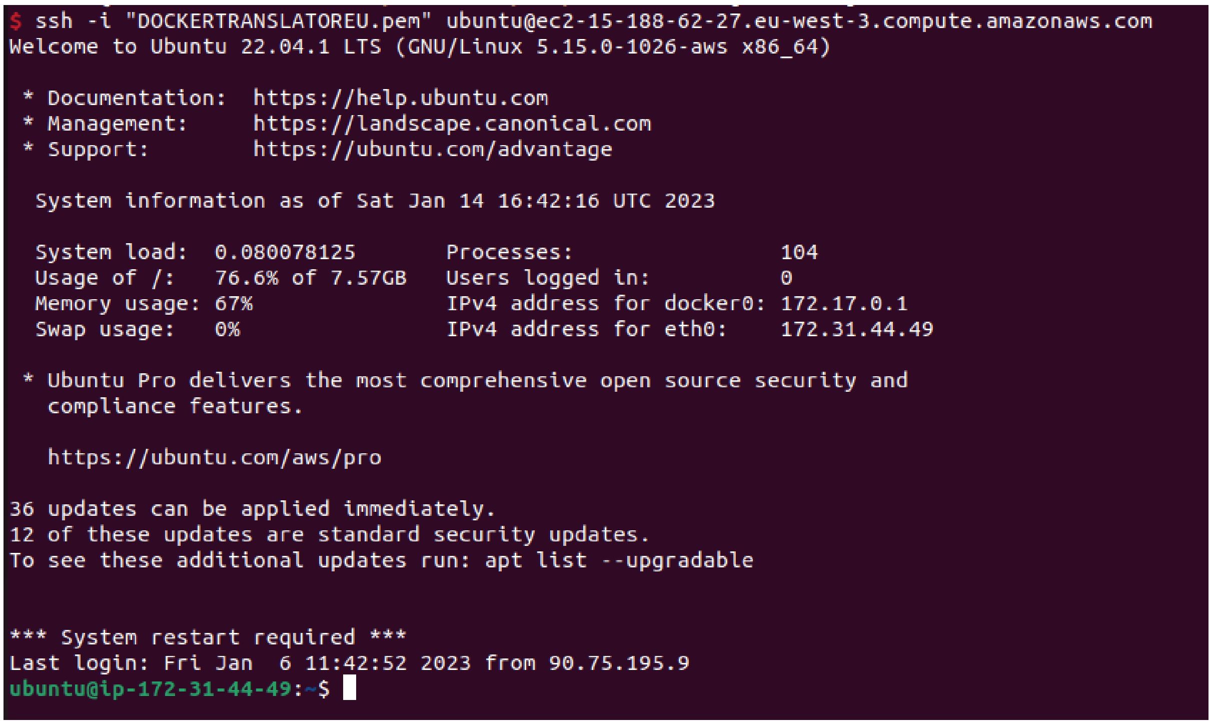
fastapi-deploy-demo

You have to download the **private key file** (*.pem file) before you can continue. **Store it in a secure and accessible location.** You will not be able to download the file again after it's created.

Figure 16: Prompt to generate a key-pair in AWS EC2

After generating this key and downloading it onto the local machine, we can ssh into the instance. Once we are in we'll need to download all the necessary packages to run our Docker application.

After setting up the registry and the instance, we'll need to connect with ssh into the instance and we'll need to, once again, log in with our AWS credentials on the AWS CLI (inside the EC2 instance).



```
$ ssh -i "DOCKERTRANSLATOREU.pem" ubuntu@ec2-15-188-62-27.eu-west-3.compute.amazonaws.com
Welcome to Ubuntu 22.04.1 LTS (GNU/Linux 5.15.0-1026-aws x86_64)

 * Documentation:  https://help.ubuntu.com
 * Management:     https://landscape.canonical.com
 * Support:        https://ubuntu.com/advantage

  System information as of Sat Jan 14 16:42:16 UTC 2023

  System load:  0.080078125       Processes:                104
  Usage of /:   76.6% of 7.57GB   Users logged in:          0
  Memory usage: 67%               IPv4 address for docker0: 172.17.0.1
  Swap usage:   0%                IPv4 address for eth0:    172.31.44.49

 * Ubuntu Pro delivers the most comprehensive open source security and
   compliance features.

   https://ubuntu.com/aws/pro

36 updates can be applied immediately.
12 of these updates are standard security updates.
To see these additional updates run: apt list --upgradable


*** System restart required ***
Last login: Fri Jan  6 11:42:52 2023 from 90.75.195.9
ubuntu@ip-172-31-44-49:~$ 
```

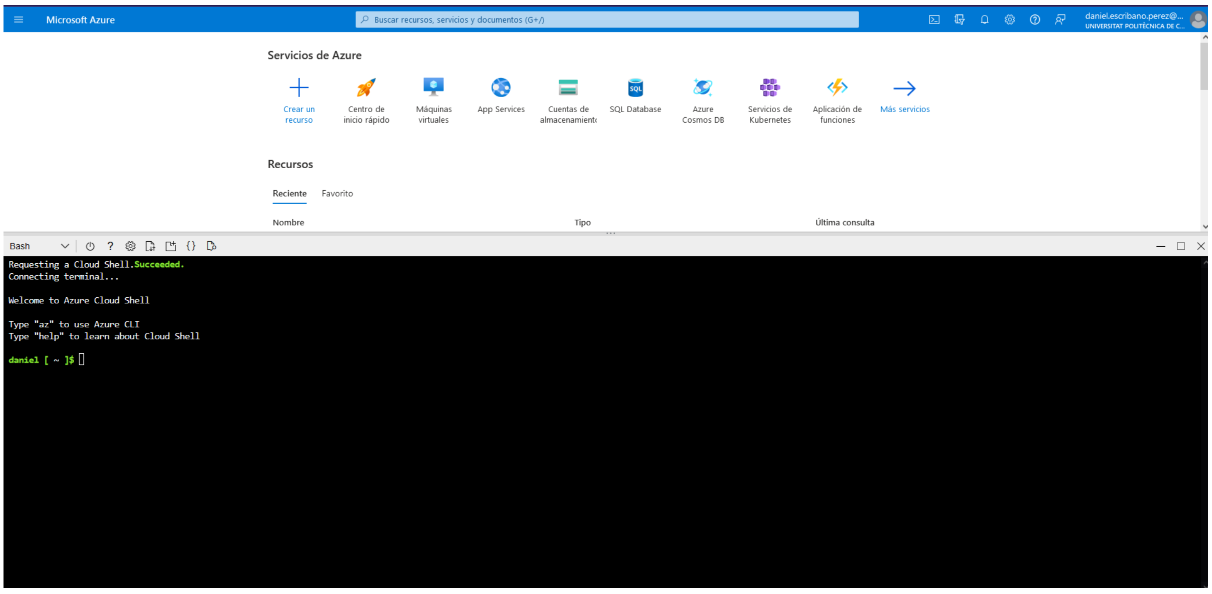
Figure 17: Change in Linux prompt signalling a correct ssh connection with EC2 instance

After doing this we can pull the docker image from the registry onto the virtual server to run the application.

To finish with the deployment process we’ll need to access the EC2 dashboard to configure our instance, allowing all traffic to communicate with the instance. This is a fairly insecure rule, but given that our application will be used only for measuring purposes we can take this risk.

Azure cloud

Similarly to AWS, the cloud platform of Microsoft follows a methodology of deployment using registries and virtual server instances, all of this via its own CLI tools. In contrast to AWS, we don’t need to download anything on our machine, as the Azure portal also provides a shell.



```
Requesting a Cloud Shell.Succeeded.
Connecting terminal...

Welcome to Azure Cloud Shell

Type "az" to use Azure CLI
Type "help" to learn about Cloud Shell

daniel [ ~ ]$ 
```

Figure 18: Integrated shell in the Azure web services website

So, to begin with, the process we need to create our registry. We can do this either via its CLI or with the graphical user interface provided by the Azure portal. As the graphical interface provides detailed descriptions of the available options that can be changed, it is recommended to use over the cli. As to which configuration to use, continuing on the same basis as in AWS when instantiating the EC2 is recommended, by accepting the recommended options when possible.

After creating the registry instance we need to build the image to push it there. To do this we need to download our application source code in the Azure shell. Cloning a GitHub repository is an easy way to do this, although other options are possible to achieve this. After downloading the source code, we need to build the docker image by using the Azure CLI, building the image on the registry automatically. Doing this we will have the image of our application in the registry.

Before creating a virtual server instance with Azure App Service, we need to allow admin users access to the registry, by enabling the necessary permissions on the registry. After doing this we can create the Azure Web App resource on the App Service, applying the recommended options, with the exception that in this case, we need to specify that the application to be published will be a Docker container. After creating the Web App instance, we need to link the image and registry to the newly created virtual server instance.

After following these previous steps, our Docker container will be deployed on the Web App instance and, after a cold start delay, it will be accessible online.

Heroku

As the first platform where the application was deployed, many changes were made to fit the constraints imposed by the provider. These proved to be of use, as they molded the system into being efficient and with low resource consumption.

Firstly, to fit the slug size specified by Heroku, (meaning the size of the whole application including the dependencies) an important change was performed to the requirements specification file. This was to use the CPU version of PyTorch, as it was much smaller than its GPU counterpart. By doing this, the slug size of the project is decreased enough to pass the constraints established by Heroku as to a project belonging to the most basic dyno available.

After performing this improvement, we must handle the resource usage of our program. Heroku machines previously available to free users(now belonging to the Eco & Basic pricing plans) had a maximum RAM size of 512 MB, heavily limiting the resource usage accessible to programs. To handle this, we must hone the computation performed by our program to only perform the strictly necessary. Besides this, we can also use the available option huggingface provides to reduce the memory usage of its pre-trained models [45].

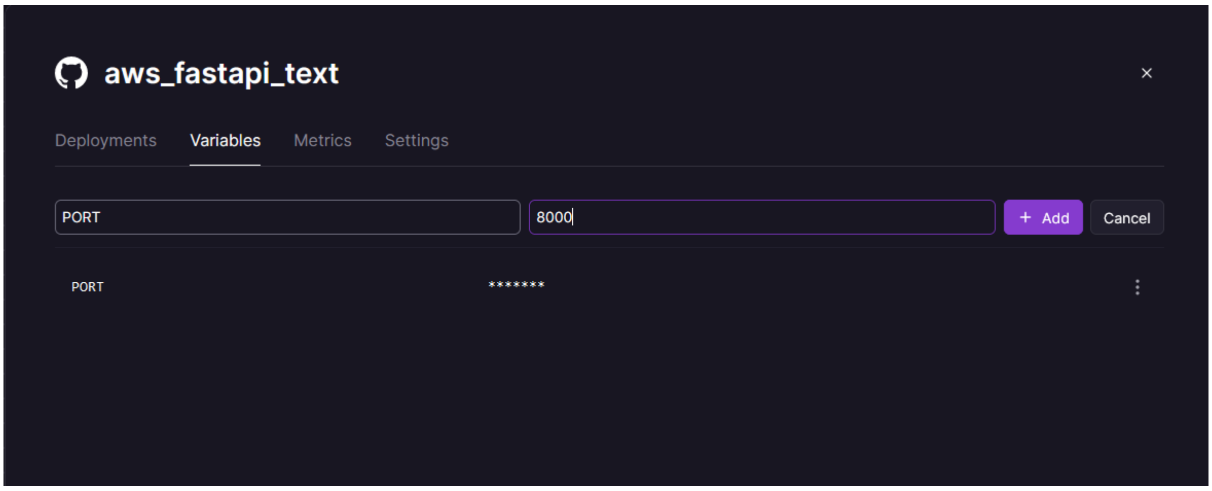
As for the deployment process itself, it is similar to AWS and Azure following the same structure of:

- Logging in to the Heroku registry of docker images
- Push an image into the registry
- Releasing the image into an application

Railway

Railway offers GitHub integration to facilitate the deployment of our application. Not only that, as Railway scans the selected project repository and, if a Dockerfile is found, it will automatically build the Docker container from the configuration file, generating and storing our image without needing further effort from the user part.

The only additional input from our part Railway needs is if our Dockerfile contains an ENTRYPOINT / CMD command which runs on a specific port, we need to tell Railway about this. By specifying the PORT environment variable on its Dashboard, Railway will know which port the application is exposing.



PORT 8000

PORT *******

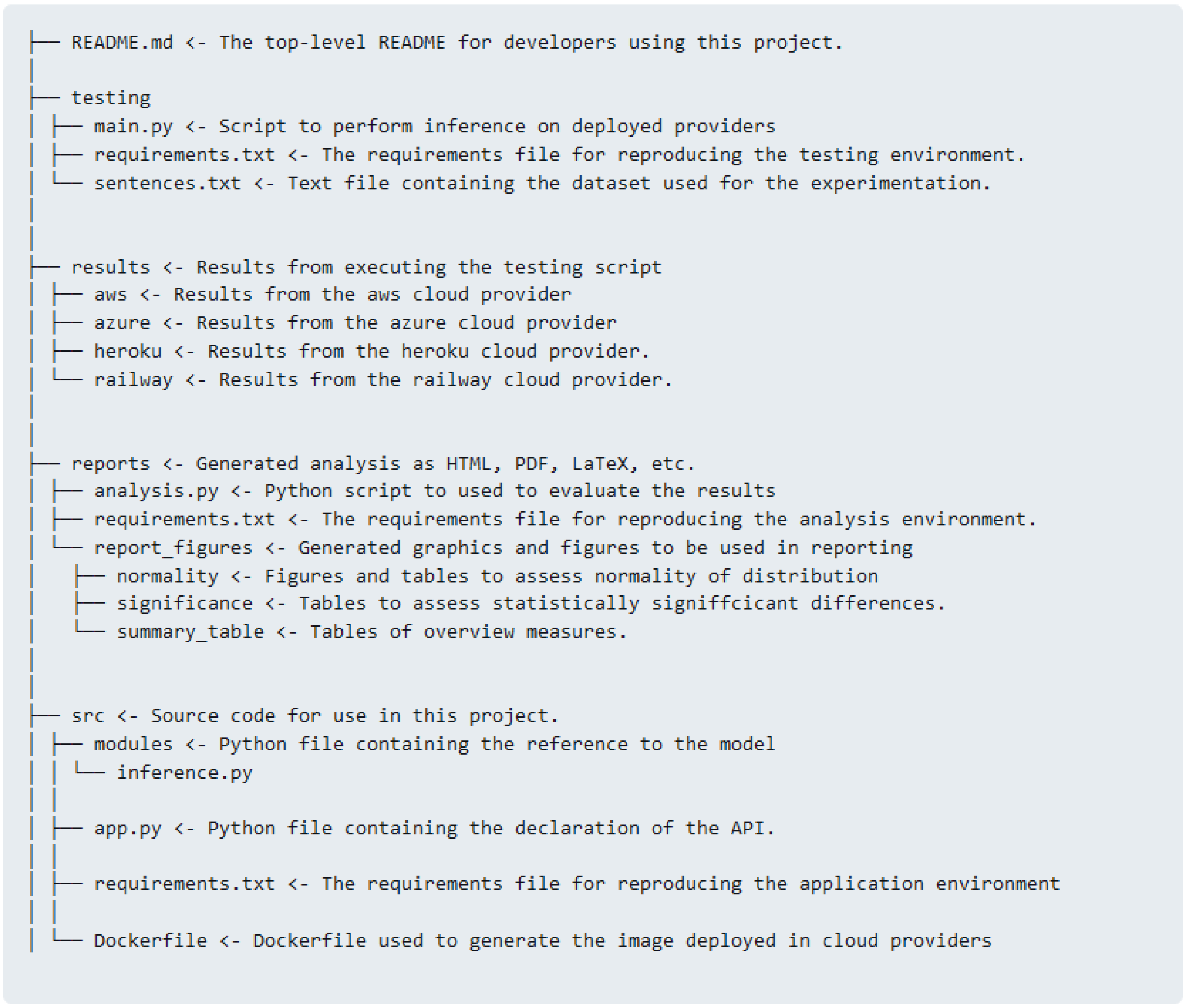
Figure 19: Adding an environment variable to Railway using its dashboard

Annex 2: Replication package

This annex explains how the scripts used for the development of this project have been structured.

When developing any kind of software development project, it is important to keep the codebase of the highest quality possible, properly organizing the code to make it easier for future developers, who may continue the work in the future.

The diagram below summarizes the structure of the code and datasets.



```
├── README.md <- The top-level README for developers using this project.
│
├── testing
│ ├── main.py <- Script to perform inference on deployed providers
│ ├── requirements.txt <- The requirements file for reproducing the testing environment.
│ └── sentences.txt <- Text file containing the dataset used for the experimentation.
│
│
├── results <- Results from executing the testing script
│ ├── aws <- Results from the aws cloud provider
│ ├── azure <- Results from the azure cloud provider
│ ├── heroku <- Results from the heroku cloud provider.
│ └── railway <- Results from the railway cloud provider.
│
│
├── reports <- Generated analysis as HTML, PDF, LaTeX, etc.
│ ├── analysis.py <- Python script to used to evaluate the results
│ ├── requirements.txt <- The requirements file for reproducing the analysis environment.
│ └── report_figures <- Generated graphics and figures to be used in reporting
│   ├── normality <- Figures and tables to assess normality of distribution
│   ├── significance <- Tables to assess statistically signiffcicant differences.
│   └── summary_table <- Tables of overview measures.
│
│
├── src <- Source code for use in this project.
│ ├── modules <- Python file containing the reference to the model
│ │ └── inference.py
│ │
│ ├── app.py <- Python file containing the declaration of the API.
│ │
│ ├── requirements.txt <- The requirements file for reproducing the application environment
│ │
│ └── Dockerfile <- Dockerfile used to generate the image deployed in cloud providers
```

Figure 20: Diagram of the replication package

Each python script listed can be executed by following the next steps:

- Generate a virtual environment using the venv library. For Ubuntu: python -m venv .env

- Activate the virtual environment. In Ubuntu: source .env/bin/activate
- Install the python requirements: pip install -r requirements.txt
- Run the script: python3 <script_name>.py

The code can be found in the following GitHub repository.

References

[1]R. Schwartz, J. Dodge, N. A. Smith, and O. Etzioni, "Green AI," Communications of the ACM, vol. 63, no. 12, pp. 54–63, 2020, doi: 10.1145/3381831.

[2]I. C. Education, "What is Machine Learning?," IBM, Jul. 15, 2020. https://www.ibm.com/cloud/learn/machine-learning

[3] Cruz, Luís. "All You Need to Know about Energy Metrics in Software Engineering." Luís Cruz, September 1, 2021. https://luiscruz.github.io/2021/09/01/energy-units.html

[4] Principles.Green. "Carbon Intensity • Principles of Green Software Engineering.". https://principles.green/principles/carbon-intensity/

[5] Amazon Web Services, Inc. "What Is Latency? - Latency Explained - AWS." . https://aws.amazon.com/what-is/latency/?nc1=h_ls

[6]R. C. Castanyer, S. Martínez-Fernández, and X. Franch, "Integration of Convolutional Neural Networks in Mobile Applications," 2021.

[7]E. García-Martín, C. F. Rodrigues, G. Riley, and H. Grahn, "Estimation of energy consumption in machine learning," Journal of Parallel and Distributed Computing, vol. 134, pp. 75–88, 2019, doi: 10.1016/j.jpdc.2019.07.007.

[8]S. Georgiou, M. Kechagia, T. Sharma, F. Sarro, and Y. Zou, "Green AI: do deep learning frameworks have different costs?," in IEEE International Conference on Program Comprehension, 2022, vol. 2022-March, pp. 36–47.

[9]R. M. Hampau, M. Kaptein, R. Van Emden, T. Rost, and I. Malavolta, "An empirical study on the Performance and Energy Consumption of AI Containerization Strategies for Computer-Vision Tasks on the Edge," in ACM International Conference Proceeding Series, 2022, pp. 50–59.

[10]E. Strubell, A. Ganesh, and A. Mccallum, "Energy and Policy Considerations for Deep Learning in NLP." [Online]. Available: https://bit.ly/2JTbGnI

[11] Volere Requirements. "Volere Requirements.". https://www.volere.org/

[12]"Manifesto for agile software development." https://agilemanifesto.org/

"Manage your team's projects from anywhere," Trello. https://trello.com/

[13]"Early-Career Researcher, Scientific Salary in Spain | PayScale."

https://www.payscale.com/research/ES/Job=Researcher%2C_Scientific/Salary/116e5c1a/Early-Career

[14]"Project Manager, Information Technology (IT) Salary in Spain | PayScale."

https://www.payscale.com/research/ES/Job=Project_Manager%2C_Information_Technology_(IT)/Salary

[15]"Early-Career Development Operations (DevOps) Engineer Salary in Spain | PayScale."

https://www.payscale.com/research/ES/Job=Development_Operations_(DevOps)_Engineer/Salary/f4356f9b/Early-Career

[16]"Early-Career Software Tester Salary in Spain | PayScale."

https://www.payscale.com/research/ES/Job=Software_Tester/Salary/92d4e6b9/Early-Career

[17]"Data Analyst Salary in Spain | PayScale."

https://www.payscale.com/research/ES/Job=Data_Analyst/Salary

[18]"Así es la jornada laboral en España: horas a la semana, duración máxima y días de descanso." https://www.larazon.es/economia/20201204/r7b4are3snfofjk2ur6ksjenfi.html

[19]"¿Cuánto cuesta el kWh de Endesa 2022? | Precio del kWh de luz y gas."

https://tarifasgasluz.com/comercializadoras/endesa/precio-kwh

[20] "Distilgpt2 · Hugging Face." https://huggingface.co/distilgpt2

[21] "Machine Learning CO2 Impact Calculator.". https://mlco2.github.io/impact/#compute

[22] Heroku Dev Center. "Heroku Python Support."

https://devcenter.heroku.com/articles/python-support

[23] *CloudBees. "The Shortlist of Docker Hosting.".*

https://www.cloudbees.com/blog/the-shortlist-of-docker-hosting.

[24]Cohen, Jason. "4 Companies Control 67% of the World's Cloud Infrastructure." PCMag, December 29, 2021. https://www.pcmag.com/news/four-companies-control-67-of-the-worlds-cloud-infrastructure.

[25] Website Builder Insider. "What Is the Difference between T2 and T3 AWS? - WebsiteBuilderInsider.Com," August 16, 2022.

https://www.websitebuilderinsider.com/what-is-the-difference-between-t2-and-t3-aws/.

[26]Heroku. "Introducing Our New Low-Cost Plans."

https://blog.heroku.com/new-low-cost-plans

[27]Heroku Dev Center. "Slug Compiler.".

https://devcenter.heroku.com/articles/slug-compiler.

[28] Cruz, Luís. "Green Software Engineering Done Right: A Scientific Guide to Set Up Energy Efficiency Experiments." Luís Cruz, October 10, 2021.

https://luiscruz.github.io/2021/10/10/scientific-guide.html

[29] "The Stanford Natural Language Processing Group."

https://nlp.stanford.edu/projects/nmt/

[30] "CodeCarbon.Io." https://codecarbon.io/#howitwork

[31] "Output — CodeCarbon 2.0.0 Documentation."

https://mlco2.github.io/codecarbon/output.html

[32] "Developer Interface — Requests 2.28.1 Documentation.".

https://requests.readthedocs.io/en/latest/api/?highlight=elapsed#requests.Response.elapsed

[33] MDN. “An Overview of HTTP - HTTP.”.

https://developer.mozilla.org/en-US/docs/Web/HTTP/Overview#http_flow

[34] Contributors to Wikimedia projects. “Shapiro–Wilk Test.” Wikipedia, December 12, 2022. https://en.wikipedia.org/wiki/Shapiro%E2%80%93Wilk_test

[35] Yeager, Kristin. “LibGuides: SPSS Tutorials: One-Way ANOVA.” LibGuides at Kent State University. https://libguides.library.kent.edu/spss/onewayanova

[36] Bevans, Rebecca. “One-Way ANOVA.” Scribbr, March 6, 2020.

https://www.scribbr.com/statistics/one-way-anova/

[37] Bewick, Viv, Liz Cheek, and Jonathan Ball. “Statistics Review 10: Further Nonparametric Methods.” Critical Care 8, no. 3 (January 1, 2004).

https://doi.org/10.1186/cc2857

[38] “R Handbook: Kruskal–Wallis Test.” https://rcompanion.org/handbook/F_08.html

[39] Unexpected Regularity. “Effect Sizes and Its Interpretation.”.

https://tien-nguyen.github.io/effect-size-and-its-interpretation/

[40] Wan, Zhiyuan. “Interpretation of Cliff’s Delta Value.” ResearchGate..

https://www.researchgate.net/figure/Interpretation-of-Cliffs-delta-value_tbl2_335425733

[41]mlco2. “Codecarbon/Impact.Csv at Master · Mlco2/Codecarbon.” GitHub.

https://github.com/mlco2/codecarbon/blob/master/codecarbon/data/cloud/impact.csv

[42] mlco2. “Codecarbon/Eu-Carbon-Intensity-Electricity.Csv at Master · Mlco2/Codecarbon.” GitHub.

https://github.com/mlco2/codecarbon/blob/master/codecarbon/data/private_infra/eu-carbon-intensity-electricity.csv

[43] “Output — CodeCarbon 2.0.0 Documentation.”.

https://mlco2.github.io/codecarbon/output.html

[44] Google Cloud. "Carbon Free Energy for Google Cloud Regions.".

https://cloud.google.com/sustainability/region-carbon?hl=fr

[45] "Models.".https://huggingface.co/docs/transformers/v4.25.1/en/main_classes/model#large-model-loading