POLITECNICO DI TORINO
Master Degree Course in Computer Engineering
Master Degree Thesis

Automotive use cases: from the real time data ingestion to the data analysis of connected vehicles

Supervised by:
Prof. Barbara Caputo

Candidate
Rocco Italiano

Company tutor:
Engr. Marco Gatta

March-April 2020

Abstract

The growing computerization and the increasing diffusion of data collection and storage technology leads to an increasing availability of data. New modern cars can give their owners many new benefits, thanks to Big Data Analytics. Many supply chains start implementing sensors on cars that, at first, were not planned as connected vehicles.
This thesis illustrates the existing infrastructures, architectures and tools in the Big Data field and how these data are analyzed and used in various automotive use cases.
The main objective of the thesis is the implementation of a scalable solution, which is not other than a pipeline that includes a series of steps, ranging from the ingestion of real time data to the analysis of the data itself with appropriate machine learning algorithms to provide useful information in terms of real-time vehicle monitoring.

Acknowledgements

This part is intended for all the people who supported and helped me on this intense but satisfying universitary studies. I write this part in Italian, as it is easier for relatives and friends to understand.

Vorrei ringraziare Marco, il mio tutor aziendale, per la disponibilità dimostrata e per avermi dato l'opportunità di entrare a far parte di un gruppo speciale. Grazie a tutti i colleghi e colleghe di Data Reply che fin da subito mi hanno fatto sentire a mio agio e parte integrante del gruppo.

Una menzione particolare va al mio relatore universitario, la Prof.ssa Barbara Caputo, che nel corso di "Machine Learning and Artificial Intelligence" mi ha fatto appassionare al mondo del Machine Learning e poi, con gentilezza e professionalità, mi ha supportato in questo lavoro.

Ringrazio la mia famiglia, che non mi ha mai fatto mancare nulla e mi ha sostenuto in ogni singolo passo di questo percorso accademico. Mio padre, che mi ha trasmesso la forza per non mollare mai, mia madre esempio di tenacia e fonte di ispirazione. Grazie alle mie sorelle e mio fratello, per avermi saputo comprendere e incoraggiare nei momenti difficili. Ai miei cognati sempre presenti a dispensare consigli. Ai miei nipoti perchè senza la loro spenseriatezza e vivacità questo percorso per me avrebbe avuto un altro sapore .

Ringrazio gli amici con i quali, nonostante la distanza, è rimasto un forte legame e affetto. Gli amici universitari, che sono ormai come dei fratelli, con i quali ho condiviso intense giornate di studio ma anche tante serate di divertimento.

Un ringraziamento va a me stesso per la determinazione e per aver costruito su ogni fallimento le fondamenta per ripartire più forte di prima.

Ultima in posizione ma prima in importanza, Catia, presenza fondamentale che con amore illumina le mie giornate.

Contents

1 Introduction **11**
1.1 Thesis structure 13
1.2 Personal effort 13

2 Big Data background **14**
2.1 Introduction to Big Data 14
2.2 Big Data infrastructure analysis 17
2.2.1 On-Premise 17
2.2.2 On-cloud 18
2.2.3 Hybrid 19
2.2.4 Conclusive remarks 19
2.3 Cloud services typologies 20

3 Machine Learning background **22**
3.1 What is Machine Learning? 22
3.2 Supervised learning vs unsupervised learning 24
3.2.1 Supervised Learning 24
3.2.2 Unsupervised Learning 24
3.3 The workflow to build Machine Learning system 26
3.4 Clustering 27
3.4.1 k-Means 27
3.4.2 DBSCAN 29
3.5 Evaluation methods 32
3.5.1 Silhouette 32

4 Analysis and tool used **34**
4.1 Why choose an On-Cloud solution? 34
4.2 On-cloud tools for Real-Time data processing 35
4.2.1 Microsoft Azure Event Hub 35
4.2.2 Microsoft Azure Service Bus 36
4.2.3 Microsoft Azure Data Lake 36
4.2.4 Microsoft Azure SQL 37
4.2.5 Microsoft Azure Databricks 37
4.2.6 Microsoft Azure Cosmos DB 38
4.2.7 Microsoft Azure Data Factory 38
4.2.8 Microsoft PowerBI 38
4.2.9 Docker 39
4.2.10 Kubernetes 40
4.2.11 Docker-Kubernetes interaction 41

5 Design and development **43**
5.1 Architecture . . . 43
5.2 Deferred path . . . 43
5.3 Real-time path . . . 44
5.3.1 Ingestion Layer . . . 45
5.3.2 Storage Layer . . . 48
5.3.3 Analitycs Layer . . . 48
5.3.4 Visualization Layer . . . 49

6 Use case **51**
6.1 Focus on the Analitycs Layer . . . 51
6.1.1 Dataset . . . 51
6.2 Use case . . . 51
6.2.1 Use case description . . . 51
6.2.2 Pre-processing . . . 53
6.2.3 Vehicles clustering . . . 54

7 Conclusions and Future Developments **59**
7.1 Future Developments . . . 59

A Appendix **62**
A.1 Citations . . . 62
A.2 Images . . . 62
A.3 Bibliography . . . 63

List of Figures

2.1 Big Data growth trend 15
2.2 Big Data 6V model 16
2.3 On-Premise vs On-Cloud typologies 20

3.1 Machine Learning as a subset of Artificial Intelligence 23
3.2 Traditional approach vs Machine learning approach 23
3.3 A branched representation of Machine Learning categories 25
3.4 The workflow to build a Machine Learning pipeline 26
3.5 k-Means clustering 28
3.6 Clustering example of a set of object using k-Means method 28
3.7 k-Means pseudocode 29
3.8 Border, Core and Noise point in DBSCAN 30
3.9 The concepts of directly density reachability, density reachability and density connected 31
3.10 DBSCAN pseudocode 31

4.1 Azure Event Hub architecture 35
4.2 Azure Service Bus flow 36
4.3 Azure Data Lake schema 36
4.4 Azure Databricks environment 37
4.5 Virtual Machine vs Docker architecture 39
4.6 Kubernetes architecture 41
4.7 Schematical representation of Docker-Kubernetes interaction 41

5.1 High level representation of the used architecture to ingest, store and analyze the data of the connected vehicles 44
5.2 Detailed real-time path architecture 45
5.3 Dependency Injection pattern example 47
5.4 Dashboard template 49

6.1 Representation of the joint distribution of engine oil pressure and engine speed for one vehicle for each month 52
6.2 Engine oil pressure box-plot and probability density distribution . . 53
6.3 Engine speed box-plot and probability density distribution 53
6.4 Graphical representation of DBSCAN clustering 55
6.5 Graphical representation of the behaviours of each cluster 56
6.6 Example of the behavior of a vehicle, for a period of six months . . . 56

Chapter 1

Introduction

This thesis illustrates the work done during a six-month Internship period, at Data Reply, a company part of the Reply group, which focuses on Big Data,Data Analytics and Quantum Computing.

Big data, in itself, is a constantly evolving concept. In this thesis the concept of Big Data is used to refer to the process of collecting large quantities of data and analyzing such data. With the growing spread of connected vehicles, it is becoming clear that Big Data Analytics will also play a very important role in the automotive field. Big data in the automotive industry allows manufacturers to gather information of great interest for vehicles. With the ability to access information from multiple sources and through the analysis of relevant data, connected vehicle systems and third party service providers can provide vehicle owners with a wide range of features and services, to significantly improve performance and the use of existing services.

The work documented in the thesis concerns one of the main customers of the company, which operates in the automotive field.
The request was to process, in a real-time context, data coming from the connected vehicles, store them in appropriate structures and according to certain formats, analyze them to provide estimates and monitor the vehicles activities and show the deserialized data and the results in suitable dashboards.
Given the requests, a solution was studied to adapt to the specifications provided, starting from the infrastructure to be used (On-Premise vs On-Cloud), then passing through the architecture and tools suitable for this context, until the choice of the algorithms to analyze data.
During the study phase, the various software and tools available on the Big Data and Data Analytics market were analyzed and considered , to try to implement a scalable and flexible solution closer to the customer's needs.
Since it has proved more advantageous in our context, it was choosen an On-Cloud infrastructure, using the tools offered by the Microsoft Azure platform, which allowed high interoperability between them and easily interaction between all blocks of the developed pipeline.
The applications to manage the ingestion and data storage phases have been developed in C # on the ASP.NET Core environment, using Docker and Kubernetes technologies.

Databricks notebooks were used for data analysis, the data were analyzed using Python as programming language. In particular, the analysis phase focuses on a particular use case. A vehicle can show different *engine oil pressure* and *engine speed* behaviors during its life. The analysis aims was to understand if there was a relationship between these parameters of the vehicle and a particular event, situation or vehicle mission.
Dashboards have been developed for the data visualization phase, using PowerBI , also them a tool provided by the Microsoft Azure platform.

1.1 Thesis structure

This thesis is structured in seven chapters, including the introductory one.
The second chapter introduces the concept of Big-Data and analyzes, highlighting the main features, the possible infrastructures available to manage these large amounts of data.
The third chapter aims to introduce the concept of Machine Learning, making an overview of the various steps necessary to build a good model for data analysis, explaining the clustering algorithms used in this thesis work and the method used to evaluate the models implemented.
This is followed by a focus on the reasons for which an On-cloud solution was chosen and on the tools used to implement the proposed solution.
Chapters five and six represent the core of the whole thesis work.
The fifth chapter explains the architecture designed for ingestion, storage, analysis and visualization of data referring to the tools used illustrated in the fourth chapter.
The sixth chapter focuses on the data analysis phase and in particular on the use case analyzed to predict the activity of a vehicle based on the values of some parameters of the vehicle itself.
After all, the final chapter summarizes the whole work and introduces possible future development scenarios.

1.2 Personal effort

The solution presented in the thesis was developed by dividing the work into several stages:

- analysis of the infrastructure and architecture to be used to adapt to customer needs.

- study and implementation of the ingestion layer to ingest real-time data, using Microsoft Azure technologies.

- study and implementation on how to structure data for saving on databases and other on-cloud platforms, such as Data Lake, to make them suitable for the analysis phase.

- evaluation of suitable algorithms to be applied in the data analysis phase using the Azure Databricks platform.

The personal effort covers all phases of analysis, design and development of the above described pipeline.

Chapter 2

Big Data background

2.1 Introduction to Big Data

The Big Data concept is a constantly evolving concept. Two quotes help us to better frame this phenomenon:

> ”Big data exceeds the reach of commonly used hardware environments and software tools to capture, manage, and process it within a tolerable elapsed time for its user population” [0].
>
> ”Big data refers to data sets whose size is beyond the ability of typical database software tools to capture, store, manage and analyse” [1].

The previous definitions make us understand that what qualifies as Big Data will evolve over time as technology advances. What was Big Data in the past or what is Big Data today will not be Big Data tomorrow.
The volume of data is constantly increasing, doubling year after year.
In the last decades, the rapid development of information technology, in particular the widespread use of social network services, Internet of Things and cloud computing services, has led to the emergence of semi-structured and unstructured data, such as text, audio, video and social media; the rapid growth of volumes and categories of data is the natural consequence of this technological phenomenon.

This different type of Big Data sources produces various types of data, which should be classified into three main categories:

structured data: the data is represented in a predefined format, organized according to rigid schemes and tables. Structured data is easily processed by traditional data processing tools (e.g. relational databases, files, etc.).

semi-structured data: semi-structured data contains some characteristics of structured data and others of unstructured data. In particular, they differ from structured data because they contain markers or tag, which impose record hierarchies and fields within the data (e.g. email, HTML documents, XML, etc.).

unstructured data: as the term itself says, this data is stored without any structure or scheme (e.g. text, video, audio etc.); this is the most complicated type of data to manage.

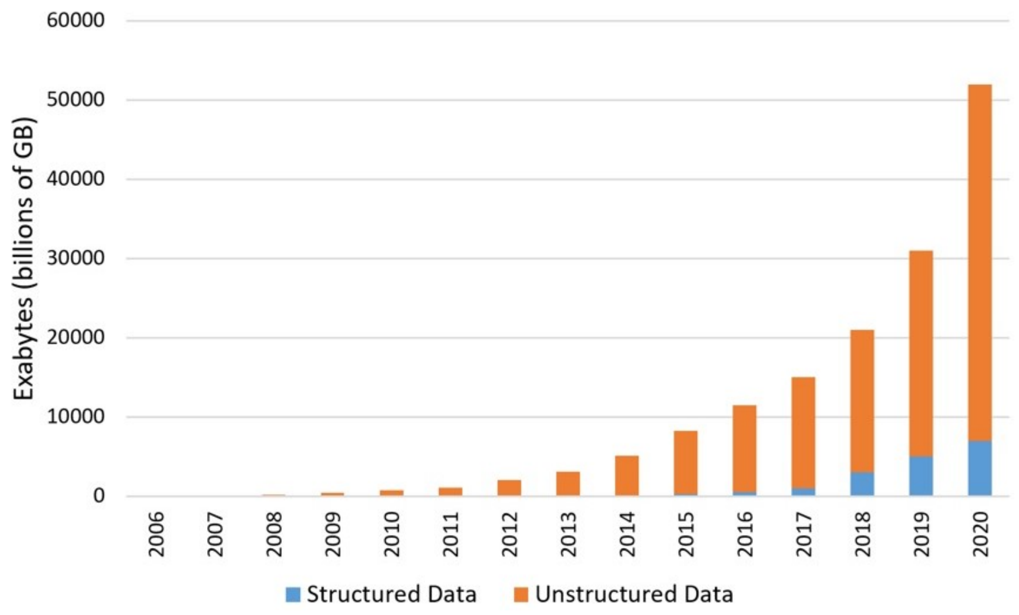


Figure 2.1: Big Data growth trend. [2]

Figure 2.1 shows, making a distinction between structured data and unstructured data, the exponential growth of the last few years.

The concept of Big Data hides within it not a reference to a single technology, but to a combination of old and new technologies that allow companies to obtain usable information.
The current definition of Big Data mainly derives from two different perspectives: the relative characteristic indicates those data sets that cannot be acquired, managed or processed on common devices within an acceptable time, while the absolute characteristic defines Big Data through a model called *Big Data 6V's* [1], represented in Figure 2.2 .

Below we will see what are the characteristics that determine this nomenclature:

Volume: this parameter is intended to indicate the current exponential growth of data produced, of the order of Terabyte, Pentabyte and sometimes also of Brontobyte. As the volume of data increases, the applications and architecture implemented to support the data must be constantly re-evaluated. Sometimes the same data is re-evaluated by looking at it from different perspectives even if the original data is the same.
The large volume essentially represents Big Data. In the past, data was created manually, now that data is automatically generated by machines, networks and interactions with social media, the volume of data to be analyzed is truly enormous.

Variety: variety refers to both structured and unstructured sources and types of data. In recent years the data are presented in the form of e-mails, photos, videos, monitoring devices, PDFs, audio and many other varieties of formats. The presence of this variety of unstructured data creates some problems for archiving, mining and data analysis. The growth of data and the explosion of social media have changed the way we look data.

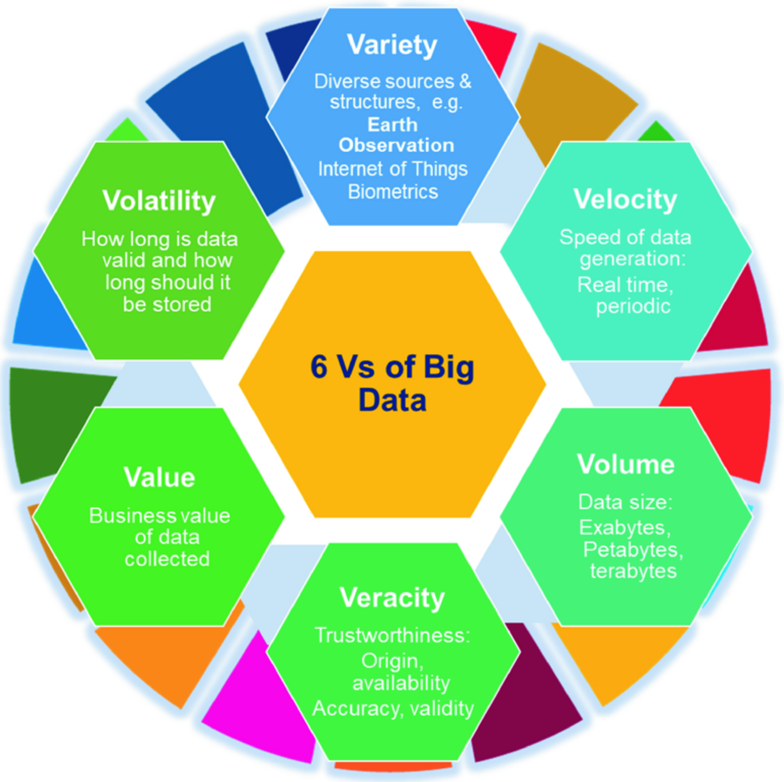

Variety
Diverse sources & structures, e.g. **Earth Observation** Internet of Things Biometrics
Volatility
How long is data valid and how long should it be stored
Velocity
Speed of data generation: Real time, periodic
6 Vs of Big Data
Value
Business value of data collected
Volume
Data size: Exabytes, Petabytes, terabytes
Veracity
Trustworthiness: Origin, availability Accuracy, validity

Figure 2.2: Big Data 6V model. [2]

Veracity: the veracity of Big Data refers to the outliers, noise and anomalies present in the data. Data must be stored and extracted consistently for the problem analyzed. When defining a Big Data strategy, you have to clean the data and processes to avoid the accumulation of "dirty data" in your systems.

Velocity: the speed of Big Data takes into account the pace data such as business processes, machines, networks and human interaction on social networks, mobile devices, and so on. The data flow is massive, continuous and very rapid.
This real time data can help researchers and businesses make valuable decisions that offer strategic competitiveness advantages if you are able to manage the speed. With the real-time movement of data, the update window is reduced to fractions of a second, it is therefore important to keep basic response times to react promptly to incoming inputs.

Value: when we talk about value, we refer to the value of the extracted data. Having infinite amounts of data is one thing, but if you can't extract important information, the value is useless. Although there is a clear link between data, this

does not mean that there is always value in Big Data.
The most important part of the big data initiative is understanding the costs and benefits of data collection and analysis to ensure that the collected data can ultimately be monetized.

Volatily: this property of Big Data refers to the period of time in which data must be considered valid and how long it should be kept. In real-time data world it is necessary to perform analyzes to understand when the data are no longer relevant to be analyzed.

After this overview, it should be clear that "big" in Big Data isn't just about volume. Big data certainly implies a huge amount of data, but it's not just that. Yes, you are receiving a lot of data, but it also comes to you quickly, in a complex format and from a wide variety of sources.
It is a rapidly expanding sector whose potential has not yet been fully explored, involving many professionals with often very heterogeneous skills.

2.2 Big Data infrastructure analysis

What is the right infrastructure for Big Data Analytics?

Is a dedicated internal infrastructure more suitable or should it be completely based on the cloud?

Could one also consider the hybrid infrastructure to store sensitive data at home and the rest on the cloud?

The purpose of this section is to answer this questions by presenting advantages and disadvantages of the different Big Data infrastructures (On-Premise, On-cloud, Hybrid).

2.2.1 On-Premise

The use of an On-Premise infrastructure means that hardware and software are managed internally by the company that uses them; therefore those who choose this option must have their own servers hosted in their corporate offices, or rely on data centers where it is possible to rent servers for their own purposes.

An analysis of this solution, with advantages and disadvantages, is shown below:

cost: initial investment tends to be high, as not just development but associated IT and hardware costs must be borne. This is a critical point for small companies or for companies that have limited budgets, which do not have the possibility of adopting a solution of this type, due to the excessive initial costs.

maintenance and reliability: the IT team will be responsible for ensuring the availability of services at all times and disaster recovery operations; furthermore it can constantly monitor situations and predict when and why services might be down. It is therefore necessary for companies to have a dedicated IT team that constantly ensures the correct functioning of the systems.

scalability: long-term planning and a commitment of resources are required to expand the scope of On-Premise infrastructure operations.
In an On-Premise environment, it is not trivial to guarantee scalability; it is very common to be faced with *under-provisioning* situations, that is cases where you would need to install more performant infrastructures to manage large amounts of data, and *over-provisioning* situations, that is the cases in which there are no large amounts of data to manage and consequently the infrastructure is underutilized.

control and security: control and security represent important advantages of the On-Premise solution, the organization can directly estimate risks and take all logical steps to reduce the same.

2.2.2 On-cloud

Differently from On-Premise, On-cloud infrastructure runs and is hosted on remote servers. Cloud services may include storage, computation and networking capabilities; the hardware and software components can be rented by an organization from a public cloud server, so the user does not have to worry about installing something. The service provider charges the user of the services a cost that is proportional to the use of the services themselves.

Let's now analyze the characteristics of this solution:

cost: initial investment is low, no extra hardware is needed and costs tend to become predictable over time.
Who choose the On-cloud solution do it also because they don't have high initial investments but only pay for the requested resources. This concept is radically different from what happens with On-premise infrastructure, in which there is an initially cost in the creation of the cluster and subsequent costs are necessary for its maintenance and or update, although not fully exploited.

maintenance and reliability : the vendor is responsible for security and a solid backup plan for recovery of data. On-cloud services are accessible from anywhere in the world at any time, but the downside is that valuable data is provided to the service provider.

scalability: this is one of the most advantageous aspects of On-cloud solution, little effort and time are needed to scale up or scale down as per requirements and wastage of resources is also minimized.It is the environment that adapts to the task, and no longer the task that adapts to the environment as it happened in the On-Premise solution.

speed and productivity: another important aspect to consider is how long it takes to create a new Big Data infrastructure.

On-cloud new environment is released in a few minutes, while On-Premise you must first configure the machines and install the services.This is a very important aspect for companies, because the time saved often turns into an increase in productivity.

2.2.3 Hybrid

Hybrid infrastructure is a mix of On-Premise and On-Cloud. This solution allows you to move workloads in the event of changes between needs and processing costs, optimizing data distribution and ensuring greater flexibility in business operations.

The main features are summarized as follows:
cost: the use of cloud-based services when needed, means that the user of the service pays more only when additional resources are needed. This can be an advantage for those who use an On-Premise solution and do not want to incur high costs to expand their infrastructure.

maintenance: a lot depends on which operation the organization conducts in-house and which it computes on the cloud; the organization reserves the possibility of safeguarding its processes and data as it deems most appropriate.

reliability: the hybrid infrastructure offers the greatest reliability among all the solutions seen so far. If any problems occur On-Premise, operations can be conducted On-Cloud, and if the cloud is inaccessible for some reason, it is possible to conduct the operation On-Premise.

scalability: this characteristic depends on which particular operations the organization conducts On-Premise and which are conducted on the cloud, so scalability varies according to the choices made.

control and security: these are other good characteristic of the hybrid architecture; the user of the service has the freedom to create security solutions for sensitive data and critical operations and rely on the vendor for the rest.

2.2.4 Conclusive remarks

After the overview of the pros and cons of the infrastructures described above, a company can choose the solution that best suits its needs based on various factors such as current business capabilities and future business objectives, growth prospects and strategic roadmap .
In the short term, hybrid infrastructure is the best solution. This allows an organization to easily switch to 100% cloud-based architecture if that is its goal. Local innovation and cloud-based analysis can be aligned to achieve maximum results and evolve interconnected.
In the long term, the best solution is certainly the adoption of cloud-based solutions, which allow to reduce long-term costs as well as guarantee scalability, productivity and competitiveness.

2.3 Cloud services typologies

Cloud computing is an IT technology that is growing very fast and embracing different types of online services.
It is important to analyze the main cloud models available (**IaaS**, **PaaS** and **SaaS**) and understand the difference.

An **IaaS** (Infrastructure as a Service) provider takes care of managing the entire infrastructure, but users have total control over it. Users are solely responsible for the installation and maintenance of apps and operating systems, as well as for security, runtime, middleware and data.

PaaS (Platform as a service) is a platform with integrated software components and tools, which allows developers to manage applications.
In this case, the vendor's responsibilities are those of managing the server, operating system updates, security patches and backups. Customers focus on app development and data without worrying about infrastructure, middleware and operating system maintenance.

Saas (Software as a service) with this plan, users have access to the vendor's cloud software. The users of the service do not have to download or install SaaS applications on local devices, they may need only a few plugins.It is possible to access the software via the Web or API, since it resides on a remote cloud network.
This is the most used category of cloud computing. Customers are exempt from any liability in this model, they only use programs to complete their tasks. In this case, the experience of the client software completely depends on the provider.

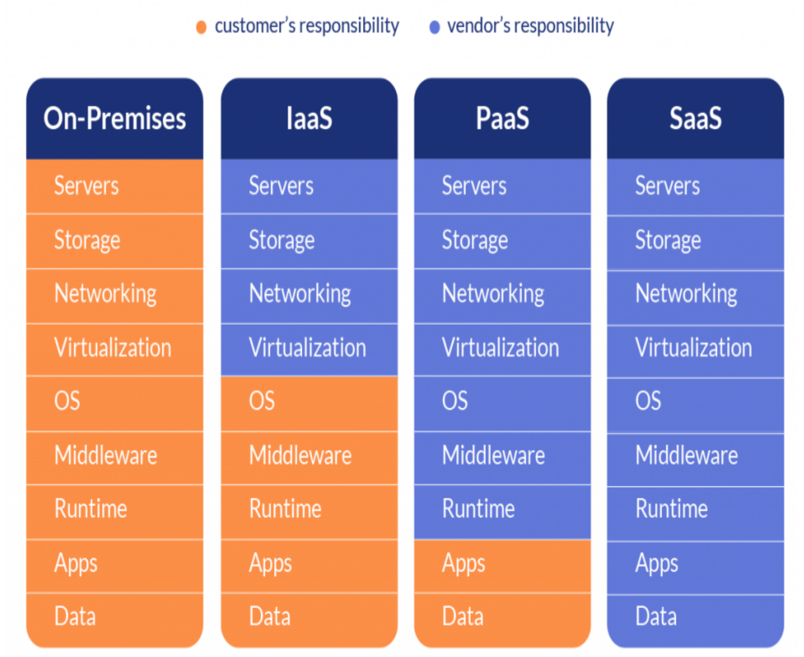


Figure 2.3: On-Premise vs On-cloud typologies.

Chapter 3

Machine Learning background

This chapter is intended to shed some light on the concept of machine learning and what is meant when you use this term.
As you can imagine, a complete discussion of the subject would require pages and pages of books and is beyond the scope of this work. The following pages aim to illustrate some useful concepts to understand the work done in this thesis and the functioning of the algorithms used.

3.1 What is Machine Learning?

" Machine Learning is the field of study that gives computers the ability to learn without being explicitly programmed " [8]

" Machine Learning is the science of programming computers so they can learn from data" [9]

The two previous quotes provide us with a slightly clearer concept of what is meant when you hear about machine learning. It seems to be something magical, but behind this concept there are no more than concepts of applied mathematics.

To explain what is Machine Learning , we need to take a step back and define what is meant by the term Artificial Intelligence.
In Computer Science, Artificial intelligence (AI) is a discipline that studies the theoretical foundations, the methodologies and techniques that allow the design of hardware and software systems capable of providing the electronic processor with performance that, to a common observer, would seem to be of exclusive relevance of human intelligence.
Machine learning (ML) is a branch of Artificial Intelligence. It is a variant of the traditional approach to computer programming tehat includes a series of statistical and mathematical techniques. This allows systems to automatically learn and improve themselves from experience without being explicitly programmed or without human intervention. The main goal of this branch is to automatically learn computers from experience.
Figure 3.1 is a hierarchical and schematic representation of what has been said so far.

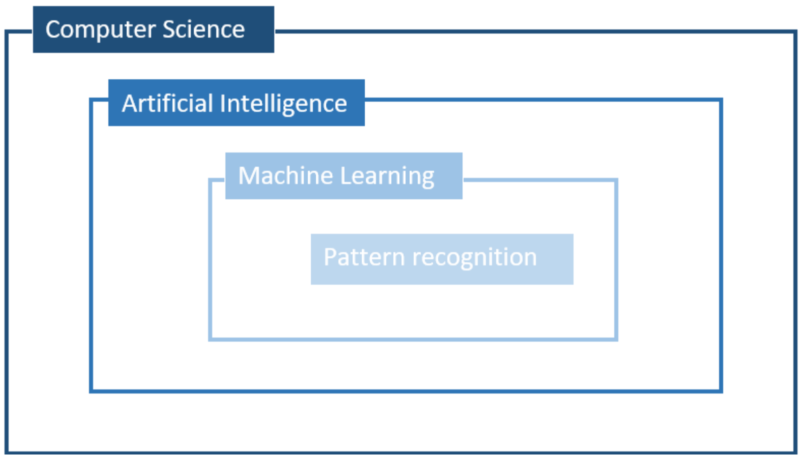


Figure 3.1: Machine Learning is Machine Learning as a subset of Artificial Intelligence.

Both in the traditional approach to computer programming and in Machine Learning the basic ingredients (data, program, computer and results) remain the same, while their order is changed: the program/model that manages to bind data with the result becomes the output from the computer, no longer an input. The machine learning algorithm learns how to pass from the data to the result and returns this process outgoing.

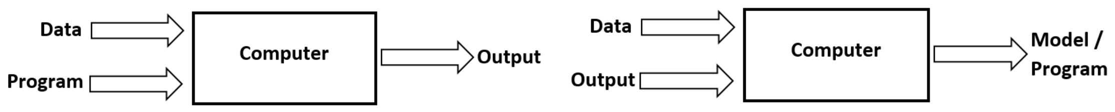


Figure 3.2: Traditional approach VS Machine learning approach.

This new approach has enormous potential and offers us mainly two advantages compared to traditional programming:

• since the program/model is generated by computer, nothing prevents us from reusing it as we please, so we could test it on new data and get new results. In this way we exploit what the model has learned in order to reuse it on future cases, thus amplifying the value of our program.

• a model can be able to identify extremely complex connections between data and results, since it is able to evaluate many more data together than a human being can do. Sometimes these connections or patterns are unusual, unexpected or, simply never noticed before. Consequently, in some cases, it is the human being who learns from the data using Machine Learning, which therefore offers us the advantage of being able to create knowledge.

Now that we are aware of the potential contained in machine learning, we can go into detail on the various categories of machine learning available.

3.2 Supervised learning vs unsupervised learning

There are some variations regarding the classification of Machine Learning algorithms, but they can generally be divided into categories based on their purpose. We mainly have three categories: Supervised Learning, Unsupervised Learning and Reinforcement Learning.
In this section we will analyze the first two categories because the third is far from the goals covered in this work.

3.2.1 Supervised Learning

In the case of supervised learning, the machine learns by observing past examples, i.e. data equipped with results. If data and results are in a single table, the column containing the results is also called the target column.
The supervised learning algorithms aim to find and reproduce, with a certain degree of approximation, the rule that binds the data in input to the target.

Just as a schoolboy is supervised by his teacher who explains how things have gone in the past and what consequences (positive or negative) they have produced, hoping that the child will be able to recognize what will happen in the future and discern what is right from what's wrong, so in supervised learning the machine learns from past examples and the results produced by them.
To use this approach, the availability of results is necessary: the teacher must necessarily "label" what is right and what it is wrong for every single example, so as to allow the schoolboy to learn.
In supervised learning algorithms, when the target assumes a numerical value, this is called *regression*. When, on the other hand, the target is of a categorical nature, we speak of *classification*.

The most commons supervised learning algorithm are:

- Decision Trees
- Linear Regression
- Support Vector Machines
- Neural Networks
- K-Nearest neighbors

3.2.2 Unsupervised Learning

When we talk about unsupervised learning, the goal of the machine is to recognize some aspects of the structure of input data, without having a past result, or a target column.

In practice in this case a teacher or supervisor is not needed: the schoolboy explores past history and identifies common traits, similarities and differences, without hav-

ing a clear right labeling available or wrong.The most common case of unsupervised learning is that of *clustering* (i.e. K-Mean, DB-SCAN ecc.).
Imagine that we want to train a machine that is able to create groups or clusters of apartments similar to each other for allow people interested in an apartment to alternatively evaluate all those who resemble it. Here we have no past results to feed the machine,in fact we want the machine to do everything by itself, without our supervision.
Figure 3.3 is a branched representation of the machine learning categories just described and the most common used algorithms for each category.

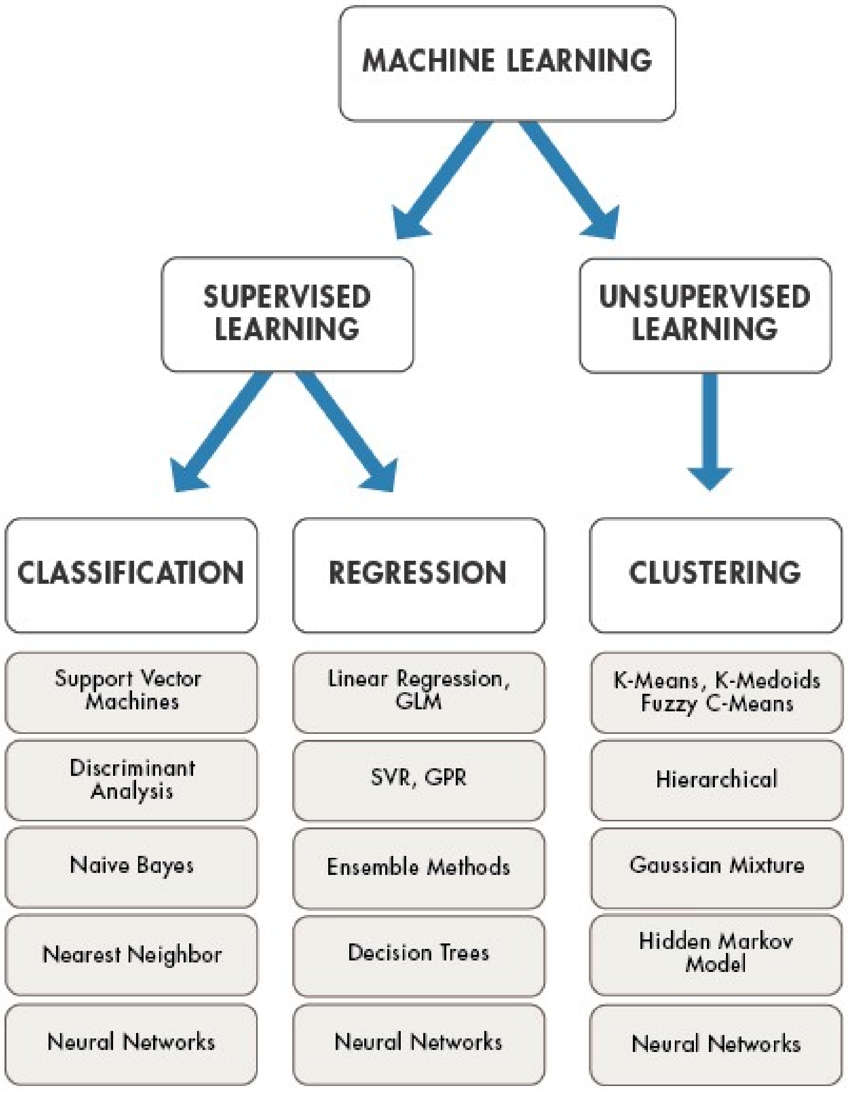


Figure 3.3: A branched representation of Machine Learning categories. [10]

3.3 The workflow to build Machine Learning system

To create an efficient Machine Learning model, you need to follow a series of steps that affect the model's performance. These steps are not unidirectional, but cyclical flows, and allow iteration to improve the scores of the machine learning algorithms and make the model scalable.
It is possible to observe the Figure 3.4, that shows a typical Machine Learning pipeline, to better frame this workflow.

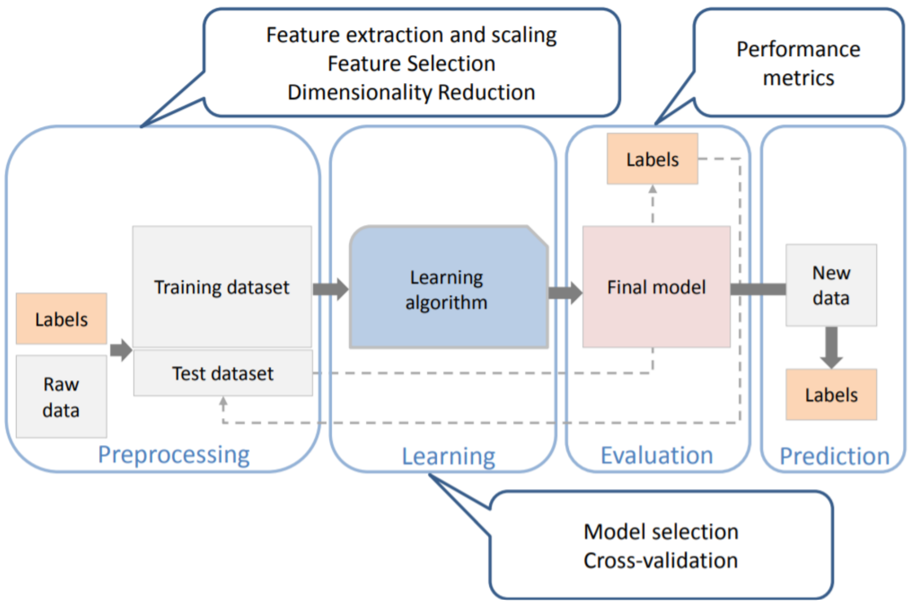


Figure 3.4: The workflow to build a Machine Learning pipeline. [11]

We will examine below the various phases of data pre-processing, algorithm learning, model evaluation and prediction and we will show the impact it has on the performance of the Machine Learning model:

Pre-processing: it is the primary task in a Machine Learning pipeline if we wants to make meaning out of data. In pre-processing phase we have a lot of different steps. First of all we have the *feature extraction* and *scaling* of data, scaling because we have a lot of different types of data with different scales, ranges and so on; so we need to aggregate this data in a correct mathematical point of view.
But all the extracted features are relevant?
This is time to use *feature selection* and *dimensionality reduction* to select the significative features.
Sometimes we have too much, noise or partial data that are not informative and we want to delete these problems, because impact in a negative way on the performance of the classifier (redundancy impact, noise impact and not informative impact).

Learning : in the second step of the described pipeline, we need to select the proper classifier, that is the mathematical best model that describe better our tasks. Depending on the data I have, the distribution of data I have, probably, I need one classifier than another one.

Evaluation: at the end of the training step we have a final classifier and we can evaluate the performance of the final model using other data, in particular using never seen before data in training phase. Of course the data using in the evaluation phase must be data that are labeled in the same way of the training data.
Compared the predicted label and real label , we can get an evaluation of the classifier goodness in the classification tasks. This is the phase that tells us how much the model is robust and how much it work well.

• **Prediction**: In the last phase the model is able to predict the task, it is trained for and new data never seen before. This is the phase in which the model use what it has learnt to provide prediction .

3.4 Clustering

Clustering is a method by which large data sets are partitioned into smaller groups, called clusters.
Clustering a dataset, containing objects described by a set of attributes, means identifying groups of objects such that:

• elements belonging to a cluster are more similar to each other (high intra-class similarity);

• elements belonging to different clusters are less similar to each other (low inter-class similarity).

Clustering is an unsupervised classification, for which there are no predefined classes and in which the discrimination of objects takes place by evaluating their attributes based on a predetermined measure of similarity.
In general, cluster detection is used when it is suspected that in the data to be analyzed there are natural groupings representative of classes or to reduce the complexity of the analysis and thus increase the probability of success of a technical analysis, in the case that the presence of different patterns in the dataset creates confusion without bringing useful information.
The application sectors in which clustering methods are used are numerous and very different from each other. The variability of the fields of application and the different characteristics of the specific problems to be faced involve, for the choice of an appropriate clustering method, the use of one or more discriminatory criteria.

The following sections illustrate k-Means and DBSCAN which are two clustering algorithms that measure data similarity using different metrics.

3.4.1 k-Means

k-Means [12] method, for its simplicity and speed is a widely used clustering technique. In this algorithm we need to define an initial target number k, that is the

number of centroids you need in the dataset. A centroid is the location that represents the center of one cluster.
Given an integer k and a set of n data points , the k-Means goal is to build k clusters, based on the data attributes and characteristics, so as to minimize the sum of the squared distances between each point and its closest cluster center.
In this way, each point is assigned to the nearest cluster and the center of cluster is recomputed as the center of mass of all points assigned to it. The assignment and center calculation steps are repeated until the process stabilizes.

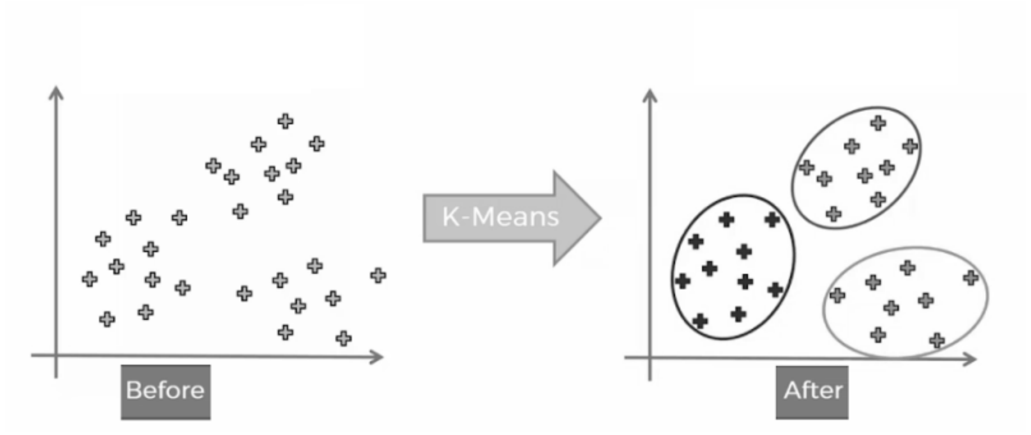


Figure 3.5: k-Means clustering. [12]

To process the data, k-Means algorithm starts with a first group of randomly selected centroids, that are used as the beginning points for every cluster. Starting from the initially chosen value of k (centroids) then it performs iterative calculations to optimize the positions of the centroids.
The iteration stops when the following two conditions are met:

• there are no change in the centroids values because they has stabilized.

• the defined number of iterations has been achieved, since there are at most k^n possible clusterings.

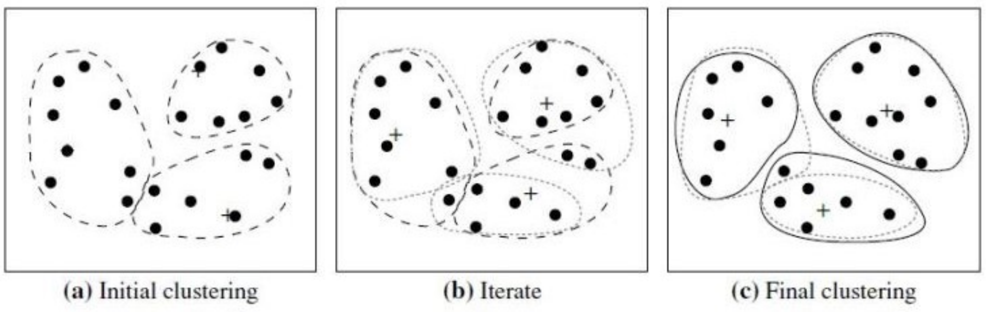


Figure 3.6: Clustering of a set of object using k-Means method. The centroid of each cluster is marked by + symbol. [12]

k-Means characteristics

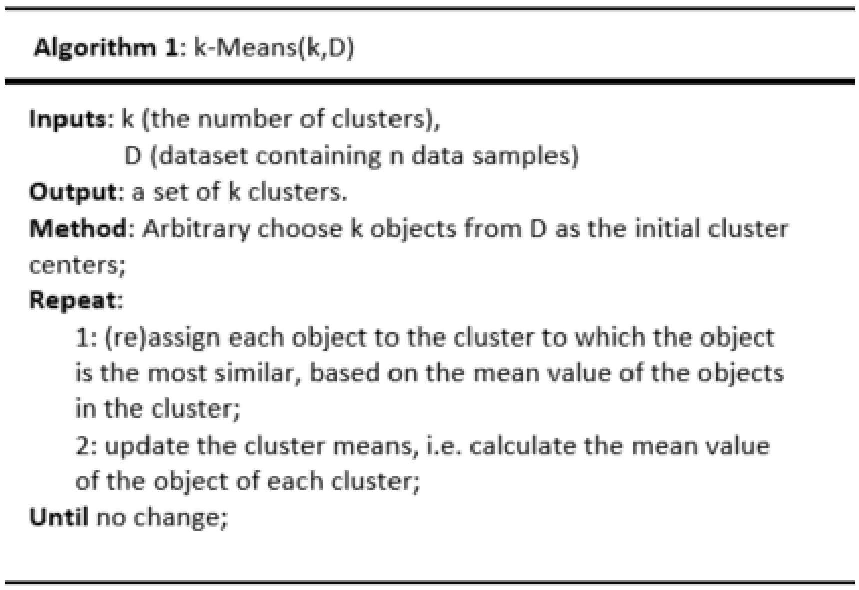
• it does not require a labeled dataset,being an unsupervised clustering algorithm.

• the algorithm is simple, easy to understand and converges quickly.

• excellent results are obtained only when the clusters within the data set to be analyzed are well separated and spherical in shape.

• the number of clusters must be fixed a priori.

• extremely sensitive to the presence of anomalies and noise in the data set.



```
Algorithm 1: k-Means(k,D)

Inputs: k (the number of clusters),
        D (dataset containing n data samples)
Output: a set of k clusters.
Method: Arbitrary choose k objects from D as the initial cluster
centers;
Repeat:
    1: (re)assign each object to the cluster to which the object
    is the most similar, based on the mean value of the objects
    in the cluster;
    2: update the cluster means, i.e. calculate the mean value
    of the object of each cluster;
Until no change;
```

Figure 3.7: k-Means pseudocode.

3.4.2 DBSCAN

DBSCAN (Density-Based Spatial Clustering of Applications with Noise) is an algorithm belonging to the clustering family.
It is based on the density of the points, identifying clusters as high density data sets separated by low density areas of points.
The main idea inside DBSCAN is that, every point belonging to a cluster must be surrounded in an area of radius *eps* by at least a minimum number of *MinPts* points. These two parameters, required at the initialization step of the algorithm, respectively define the radius of proximity around a point x and the minimum number of points that must coexist within the area of *eps* radius in order to form a cluster. The density concept adopted by DBSCAN is defined mainly by these two parameters.
Before going to see how the algorithm identifies the cluster, it is necessary to make a preliminary overview of some concepts present in the DBSCAN.

First of all, the algorithm defines three types of points (illustrated in Figure 3.8):

• each x point of the dataset it contains, within its radius region *eps*, a number of data greater than or equal to *MinPts* is called the **core point**;

• a given x is considered **border point**, if the number of its neighbors inside of its region of radius *eps* is less than *MinPts* but the data itself is placed within an area of any *core point*;

• the points which are neither *core points* nor *border points* are considered **outlier**.

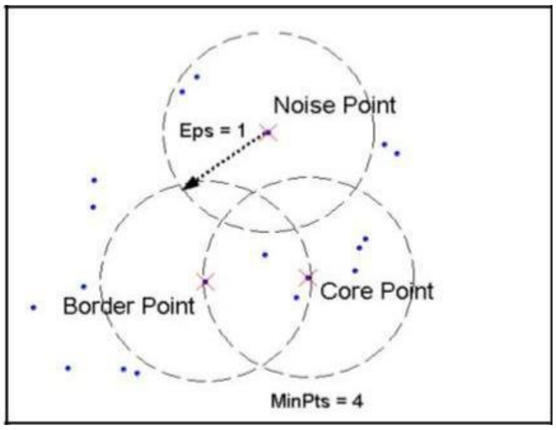


Figure 3.8: Border, Core and Noise point in DBSCAN. [13]

Furthermore, the following concepts, that are illustrated in Figure 3.9, must also be introduced:

• **Direct Density Reachable**: a point *u* is *direct density reachable* from a point *v*, if *u* is within the *eps* radius area of *v* and *v* is a *core point*;

• **Density Reachable**: a point *u* is *density reachable* from *v*, if there is a set of *core points* ranging from *v* to *u*;

• **Density Connected**: points *u* and *v* are *density connected* if there is a *core pointm* such that both *u* and textitv are density reachable from *m*.
But in practice how does the algorithm work?
The main steps of the algorithm are illustrated below:

• for each point x, it is calculated the distance between x and the other points of the dataset. All the points within them radius area *eps* contain a data number greater than or equal to *MinPts*, they are marked as *core points*;

• for each core point not yet assigned to a cluster, create a new cluster. At this point, recursively, search for all *density connected* points and assigns them to the same cluster of the *core point*;

• the algorithm ends as soon as all points have been marked as visited; all points that do not belong to any cluster are considered *outliers*;

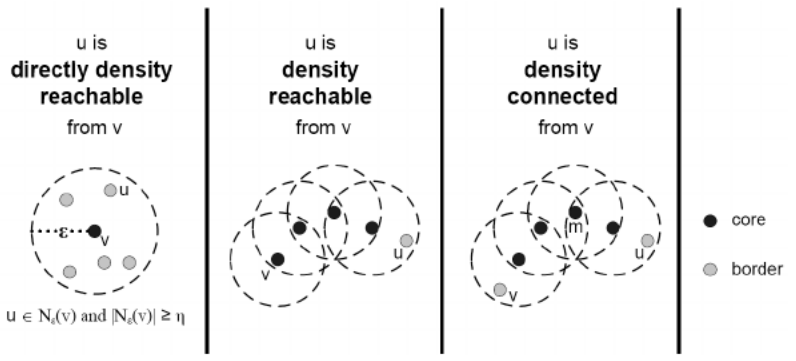

u is directly density reachable from v
u is density reachable from v
u is density connected from v
u
v
ε
m
$u \in N_\varepsilon(v)$ and $|N_\varepsilon(v)| \geq \eta$
● core
● border

Figure 3.9: The concepts of directly density reachability, density reachability and density connected. [14]

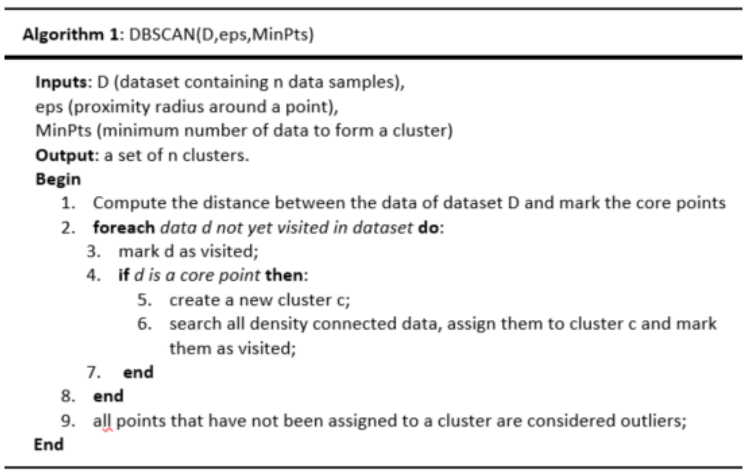
DBSCAN characteristics

• differently than k-Means, DBSCAN does not require you to specify in advance the number of clusters k to be generated;

• it automatically identifies outliers and arbitrary-shaped clusters can be easily identified;

• the parameters *eps* and *MinPts*, must be fixed a priori even if they are often difficult to determine;

• the algorithm goes into difficulty when it has to identify and separate clusters of similar density;



```
Algorithm 1: DBSCAN(D,eps,MinPts)

Inputs: D (dataset containing n data samples),
eps (proximity radius around a point),
MinPts (minimum number of data to form a cluster)
Output: a set of n clusters.
Begin
  1. Compute the distance between the data of dataset D and mark the core points
  2. foreach data d not yet visited in dataset do:
     3. mark d as visited;
     4. if d is a core point then:
           5. create a new cluster c;
           6. search all density connected data, assign them to cluster c and mark
              them as visited;
     7. end
  8. end
  9. all points that have not been assigned to a cluster are considered outliers;
End
```

Figure 3.10: DBSCAN pseudocode.

3.5 Evaluation methods

Evaluating the performance of an unsupervised classification algorithm is not as simple as calculating the accuracy of a supervised classification model. When the labels of a dataset are not known a priori, the evaluation must be performed using the model itself.
One of the best known methods in this field is the "Silhouette coefficient".

3.5.1 Silhouette

The **Silhouette coefficient** measures how similar a data is to the belonging cluster compared to other clusters.

The value of Silhouette s for a single data sample is calculated as:

$$s = \frac{b - a}{max(a, b)}$$

where:

a is the average distance between a data sample and the other data belonging to the same cluster.

b is the average distance between a data sample and the other points of the nearest cluster.

The value of the Silhouette coefficient is limited in the range between -1 and +1.
Values around zero indicate overlapping clusters.
Values close to -1 indicate that data sample has been assigned to the wrong cluster and therefore another cluster is more similar.
Values close to +1 indicate that clustering techniques have correctly identified the most similar data sets.

Chapter 4

Analysis and tool used

This chapter, after a brief introduction that will explain the reasons why an On-Cloud solution has been chosen, is intended to introduce the tools used, mostly belonging to the Microsoft Azure platform, to develop the thesis work described in the next chapter.

4.1 Why choose an On-Cloud solution?

The reasons that led us to adopt an On-Cloud solution and in particular Microsoft Azure Cloud were manifold. This solution is well suited to the evolution of the entire customer platform that include the management of new advanced telematics devices and the transmission of an ever increasing amount of data.
Adopting an On-Premise solution would not have been optimal for a number of reasons, such as:

scalability: an On-Premise solution made the scenario difficult to manage and not very scalable. Scalability is a very important element to consider because of the amount and flow of data, mainly of the real-time type. An On-Cloud infrastructure has one of its major strengths in scalability.

performance: the performance aspect is much easier to manage with an On-Cloud solution. With a cloud-based solution, you can better manage under-provisioned and over-provisioned situations, adapting resources to incoming data flows in a flexible way.

software and hardware Maintenance: installing new components requires excessive efforts, skills techniques and a lot of attention. Furthermore, in the case of additions of new machines, it would have been necessary to install dated software to have compatibility with the other pre-existing nodes.
Requesting the replacement or repair of some nodes that make up the cluster, would have meant waiting for at least a few weeks, with consequent disservices. Also in this case a cloud-based solution frees us from a series of problems, facilitating, among other things, work and increasing productivity.

costs: the use of an On-Cloud infrastructure is much more advantageous also from the point of view of cost reduction. On-Premise the costs would have been much higher due to the maintenance of the architecture, monitoring of the systems and uninterrupted consumption of power.

4.2 On-cloud tools for Real-Time data processing

Before proceeding to the explanation of the various implementation phases of the solution requested by the customer, the main tools used in the various phases will be introduced, with a particular focus on the Microsoft Azure components used.

4.2.1 Microsoft Azure Event Hub

The **Azure Event Hub** is a big data streaming platform and event ingestion service, that is able to receive and process millions of events per second. It is a scalable platform , in fact it is possible to work with data streams of different magnitude, starting from megabytes up to gigabytes or terabytes.
Since data is valuable only when there is an easy way to process it and get timely information, the event Hub provides a low latency distributed stream processing platform.
The event hub is the event ingestor of our architecture and it represents the "front door" for our event pipeline. Since it uses a partitioned consumer model, in which each consumer only reads a specific subset, or partition of the message stream, it enable multiple applications to process the stream concurrently and control the processing speed. A partition is nothing more than an ordered sequence of events held in the event hub and when new events arrive, they are added to the queue of the sequence. It is possible to use a partition key, that is a sender-supplied value passed into an event hub, to map incoming event data into specific partitions for the purpose of data organization.
Basically the event hub sits between an event publishers and an event consumers to separate the production of an event stream from the consumption of those events (Figure 4.1).

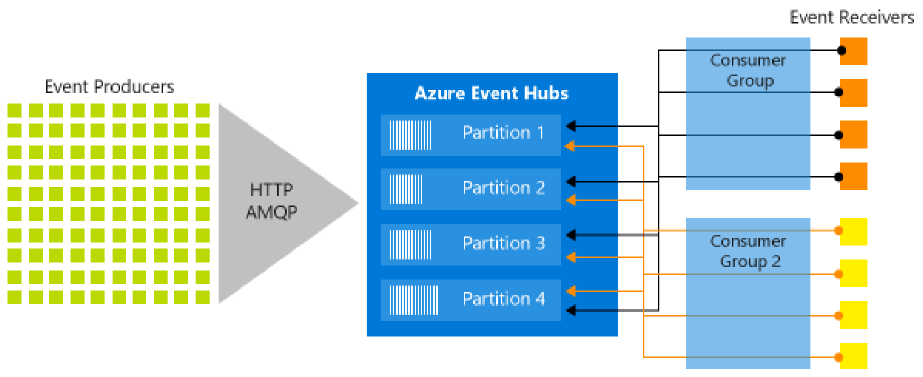


Figure 4.1: Azure Event Hub architecture. The event producer sends data to the event hub, each consumer only reads a specific subset, or partition, of the message stream and finally event receiver reads event data from the event hub. [15]

4.2.2 Microsoft Azure Service Bus

The **Azure Service Bus** is a fully managed message broker that guarantees reliability and security for the asynchronous transfer of data and status.
This component offers the possibility of interaction between applications and services and allows the management of messages through simple queues, or the management of exchanges through publisher/subscriber mechanisms, or the direct connection between applications.We are talking about a service shared between multiple users (multi-tenant service) who have the possibility to create a namespace and define the communication mechanisms necessary within it. A namespace is a container for all messaging components and multiple queues and topics may be in a single namespace.

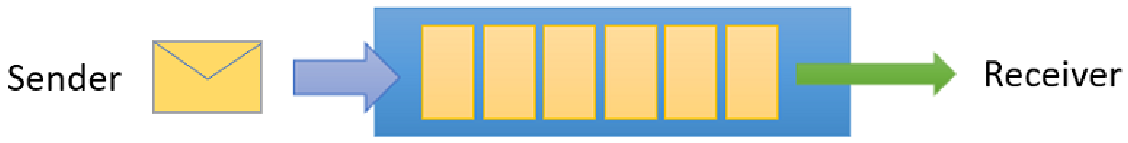


Figure 4.2: Message queue with messages. [15]

In the implementation of the application object of the thesis, the simple queue mechanism was exploited: a sender sends a message and a recipient uses it at a later time. Each message is composed of two parts: a first part that describes the properties of the message, understood as key-value pairs and the second part which is the payload, which can be in textual, binary or even XML format.

4.2.3 Microsoft Azure Data Lake

The **Azure Data Lake** is a scalable and optimized data storage and analysis service, through which it is possible store structured, semi-structured or unstructured data in their native format.
This tool, in addition to keeping the data in their native format, provides tools for analysis, interrogation and processing. Using data lake involves eliminating the restrictions present in a typical data warehouse system by providing unlimited storage space, unlimited file sizes (orders of magnitude greater than one petabyte), and different ways of accessing data(including programming, REST call or query similar to SQL).In addition, the data lake allows you to propagate parts of files on individual storage servers, making the speed recorded when reading files parallel. This makes the instrument particularly suitable for analysis of large amounts of data, as in our case.

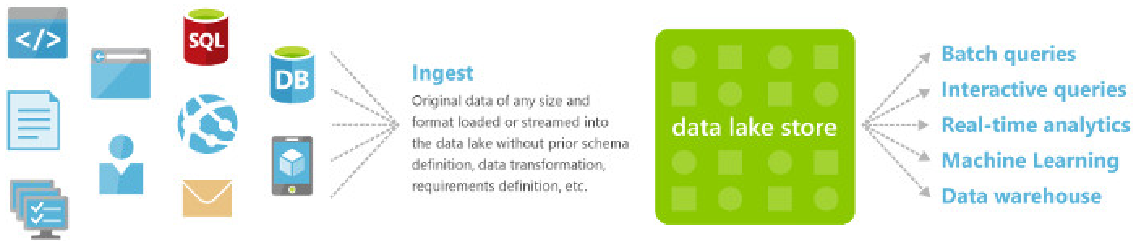

SQL
DB
Ingest
Original data of any size and format loaded or streamed into the data lake without prior schema definition, data transformation, requirements definition, etc.
data lake store
Batch queries
Interactive queries
Real-time analytics
Machine Learning
Data warehouse

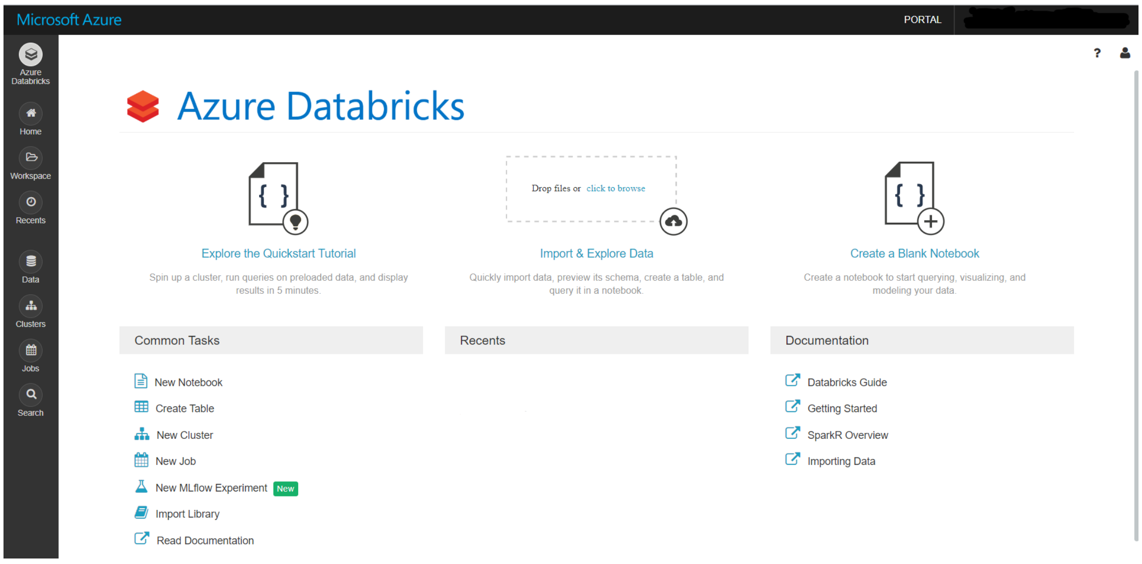
Figure 4.3: Azure Data Lake schema. [15]

4.2.4 Microsoft Azure SQL

The **Azure SQL** database is a cloud-based general-purpose relational database that offers scalability, backup and data availability.
It also includes integrated intelligence to recognize application models and maximize performance, reliability and data protection. It is provided as managed service. The user can handle the database through tools such as sql-cli, VS-Code and Microsoft tools such as Visual Studio, SQL Server Management Studio, PowerShell, API REST or directly from the Azure portal.

4.2.5 Microsoft Azure Databricks

Azure Databricks is an Apache Spark-based analytics platform thought and optimized for the Microsoft Azure cloud services.
Databricks offers simple configurations, optimized workflows and an interactive workspace that facilitates collaboration between data scientists and data engineers. The platform implements the role-based access control functionality and other optimizations that improve usability for users and reduce costs and complexity for administrators.It is possible to use different programming language such as Scala, Python, R and SQL.



Spin up a cluster, run queries on preloaded data, and display results in 5 minutes.

Quickly import data, preview its schema, create a table, and query it in a notebook.

Create a notebook to start querying, visualizing, and modeling your data.

Figure 4.4: Azure Databrick environment screenshot.

This platform is mainly based on the following services/tools:

Clusters: they are a set of Linux virtual machines that host the Spark Driver and worker nodes, on which applications will run.
Through a simple and intuitive interface, it is possible to create and customize clusters, specifying the number of nodes, the autoscaling and self-termination policies, the types of virtual machine instances for the nodes and other Spark-related configurations.

Workspace: each user has his own workspace in which to organize his notebooks and own libraries. The organization is hierarchical and there is the possibility of giving visibility of your workspace to other users.

Notebooks: notebooks are very useful because they allow Spark applications to be run directly on the cluster. Inside the same notebook it is possible to use different programming languages (Python, Scala, SQL), moreover it is possible to configure the accesses, to share or not to share the code with other people.

Libraries: these are containers intended to host the user's Python, R, Scala libraries. Once it has been imported, the source code will be visible within the workspace.

Job: the Spark code is executed by the Databricks clusters in the form of "Jobs". The platform offers graphic tools to create and monitor the Jobs that are running.

4.2.6 Microsoft Azure Cosmos DB

Azure Cosmos DB is globally distributed multi-model database service, launched by Microsoft in 2017.
Given its characteristics, it is particularly suitable for companies that need a scalable and globally distributed database and for applications that must be very responsive and always online. This means that all resources are replicated in different geographic areas in a capillary way.
It can be used for different data models (documents, key-value pairs, relationships), using a single backend. It is considered a NoSQL database because it is not based on anyone scheme. However, since it uses SQL-like query language and can easily support the ACID transactions, it is often cataloged as a type of NewSQL database.
It should be remembered that it is a Multi-API service: the data are automatically indexed and the user can access them using the prefered API (SQL, MongoDB, Cassandra, Gremlin, Javascript, Azure Table Storage.)

4.2.7 Microsoft Azure Data Factory

Azure Data Factory is a service designed to allow developers to integrate data from different sources.
It is a platform that allows to create an activity pipeline, thanks to a simple interface and block elements to drag. In this way it is possible to create ETL (extraction, transformation, loading) processes and data integration processes.
To create your own flow of execution all the user just select the various available components , set the desired configuration and then schedule the execution. Data Factory also offers a dashboard that allows you to monitor your data pipeline.

4.2.8 Microsoft PowerBI

PowerBI is an analysis service provided by Microsoft. It provides interactive visualizations with business intelligence functionality, it is easy to use and very intuitive, so even inexperienced users can create simple dashboards and reports in a short time.

Thanks to different connectors, continuously growing, it is possible to access different data sources, from Azure Data Lake to Azure SQL and also to non-Microsoft data sources, using different data formats, from json to csv to tables.
The user can therefore create graphs, tables and other graphical elements of their interest and it also has the possibility of sharing its products privately with other users or publicly on the Internet.

4.2.9 Docker

Docker [17] is a software component, which is nothing more than a technology that allows you to create and use Linux containers.
The containers have characteristics similar to those of virtual machines, the main difference between these two entities is in the different way in which their architecture was designed. In particular, the containers share the host system kernel (i.e. the operating system) with other containers, while the virtual machines compress the kernel and user space for each new virtual machine (as shown in Figure 4.4).
This feature makes containers light and portable, offering the possibility of creating and distributing containers from one environment to another in a flexible way.

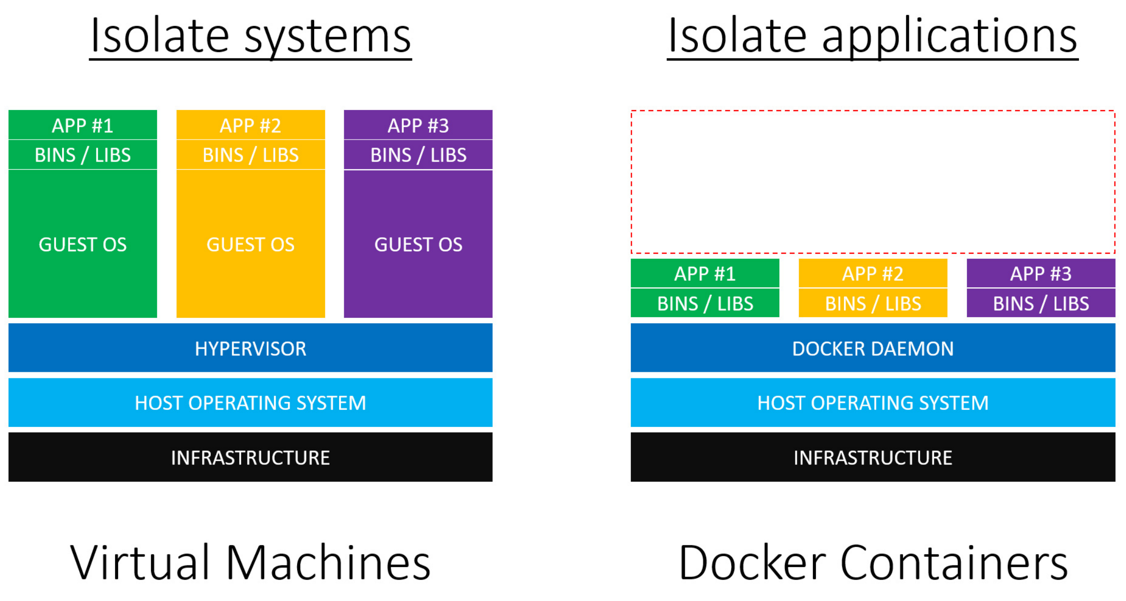


Figure 4.5: Virtual machine vs Docker architecture. [16]

Docker takes advantage of the Linux kernel and its features in order to isolate processes and be able to run them independently.
The reasons why Docker was used in this work is mainly due to the advantages it offers, which are described below:

Modularity and **scalability**: Docker uses an approach, based on microservices, that allows you to work at the level of individual components of an application as well as allowing the sharing of processes between multiple applications.
In addition, the ability to connect containers to create your own application makes it scalable.

Layers and **versioning**: Docker images are made up of multiple layers, each layer corresponds to an image modification. Each modification is shared between the images and this allows to have improvements both in terms of speed and efficiency. As for the versioning of the images, the change registry, each time a change is made, allows you to have control of all the images present in the container and possibly make a rollback to return to the previous desired version.

Quick deployment: this feature saves a lot of time, in fact it is possible to make application releases quickly, since it is not necessary to restart the operating system to manage the containers.

4.2.10 Kubernetes

Kubernetes [17] is an open source platform that allows you to easily manage clusters of hosts on which Linux containers are run. Kubernetes has the main purpose of orchestrating, managing and optimizing the distribution of containers (sometimes it is necessary to distribute them on multiple hosts) to improve the management of workloads. Since the Kubernetes architecture being of the master-slave type is composed of at least one master and multiple computing nodes.

Like any technology, Kubernetes also has specific terms and concepts, which will be explained below:

Master: as the name itself says, it is the origin of all activities, in fact it represents the machine that controls the nodes and the entire cluster.

Nodes: they are machines, physical or virtual, that perform the required tasks and they are managed by the master node.

Pod: A pod is the instance of an application or process, consisting of one or more containers, running on a single node. It is a group of containers that can be started, stopped and replicated from the platform.At runtime, the replicated pods guarantee *reliability*, replacing the containers with possible failures, *scalability*, adding resources in case of need, *load-balancing*, allowing the distribution of traffic on different instances.

Replication Controller: it has the task of controlling the number of identical copies of a pod to be run on the cluster.

Service: it aims to decouple job definitions from pods. Even if pods are repositioned in the cluster, Kubernetes' service proxies automatically redirect requests to the correct pod.

Kubelet: this service is performed on the nodes and has the task of reading the manifest of the containers and ensuring their correct start and execution.

kubectl: is the Kubernetes command line configuration tool.

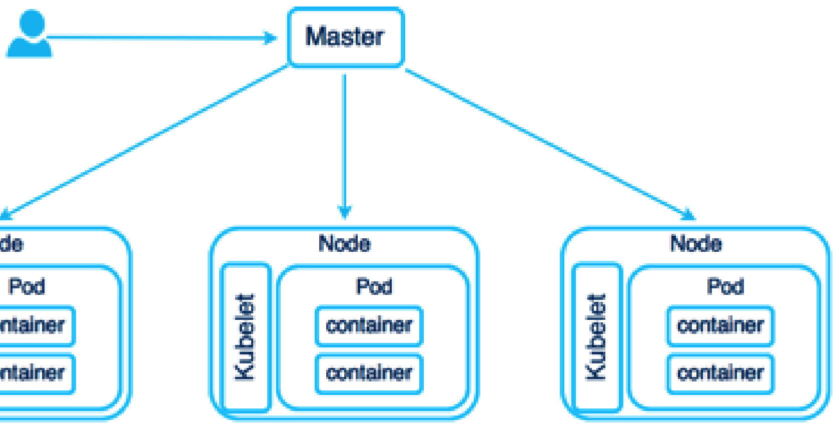


Figure 4.6: Kubernetes architecture. [17]

4.2.11 Docker-Kubernetes interaction

Figure 4.7 illustrates a schematic representation of the interaction between Kubernetes and Docker.

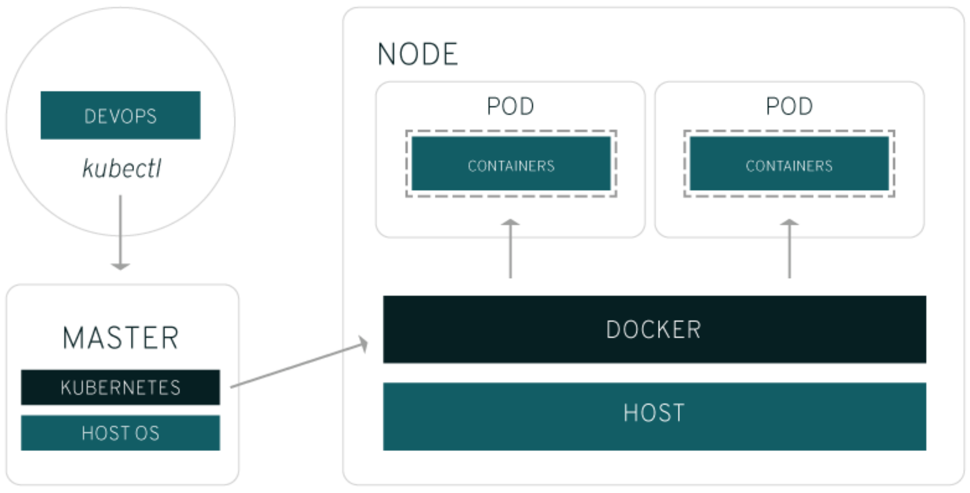


Figure 4.7: Schematical representation of Docker-Kubernetes interaction. [18]

Kubernetes' task is essentially to interact with the pods of the containers running on the nodes; the main node receives commands from an administrator and forwards these instructions to the subservient nodes.
At this point the kubelet on the subservient nodes asks Docker to start the necessary containers, allocating the resources and assigning the pods in that node.Container status information is aggregated on the main node from the kubelet.
Docker then sends the containers to the node and takes care of stopping and starting them.
Then Kubernetes interacts with Docker to orchestrate by asking Docker to perform operations in an automated way.

Chapter 5

Design and development

The purpose of this chapter is to explain the architecture, the various phases and the tools through which the data have been processed to reach the desired result.
The incoming data had to be managed in real-time, which meant that a large amount of data per second from the connected vehicles had to be managed and that data could only be accessed temporarily,through the use of appropriate memory buffers .
Therefore, it was necessary to have a system capable of rapidly scaling to adapt to the peaks of the data volume, maintaining low latencies and high productivity.
It was also very important to maintain a temporal backup of the received data to ensure system availability and to deal with any system failures, so as not to lose large amounts of data.

5.1 Architecture

In Figure 5.1, it is possible to observe the architecture of the solution, in a high-level view, and the tools used for each step. The work presented in this thesis, starts from the moment the data, coming from the connected vehicles, are received from the event hub, the previous part of receiving messages directly from the connected vehicles and the sending of these to the event hub has not been managed directly from our work team but from other Reply colleagues.

The used architecture forks into two different paths: **real-time path** and **deferred path**.
As it is easy to guess from the Figure 5.1, deferred path was designed to save the data as it arrives from the vehicles, in raw format without any manipulation, in order to mantain a backup of the data. Real-time path instead includes a series of blocks to do the ingestion, storage, analysis and visualization of the real-time data.

The following sections explain these two paths in more detail by placing the focus on the various phases of each path.

5.2 Deferred path

As previously mentioned, deferred path is the path used in our architecture to store data in raw format to have a long-term backup of the data.

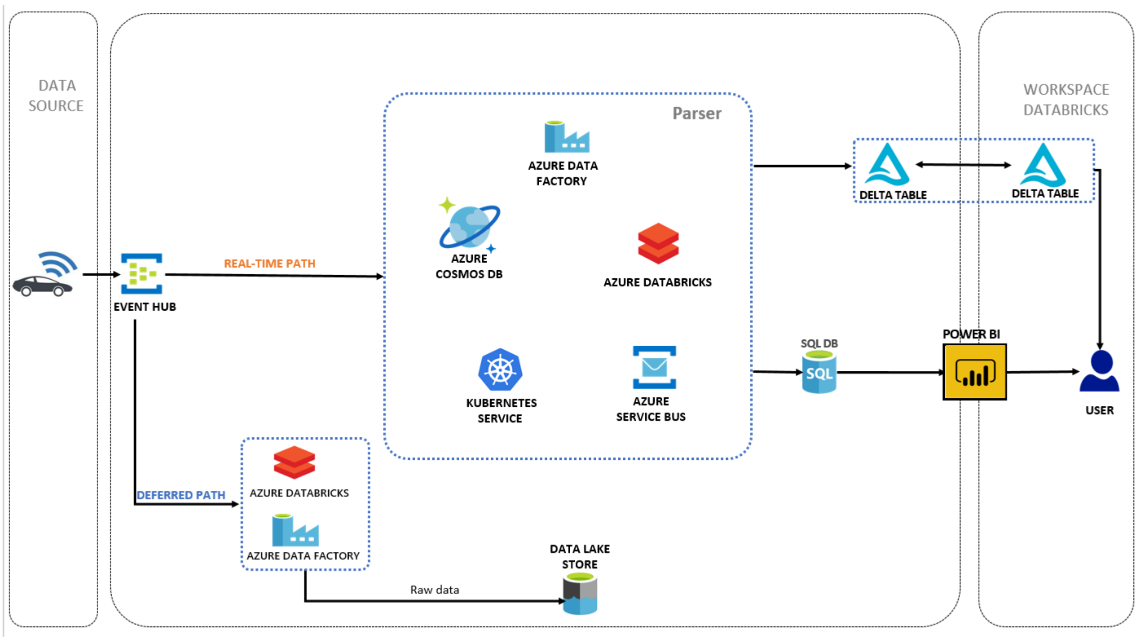


Figure 5.1: High level representation of the used architecture to ingest, store and analyze the data of the connected vehicles.

This path was designed to deal with any cases of malfunction or failure of the real-time path. In this way, the availability of the real-time service is provided even in the case of unforeseen events, avoiding to lose the data received from the connected vehicles. This is possible thanks to the data backup present in the storage, which can be reprocessed later.
The data that end up on the deferred path, which are not designed to give immediate answers to the user but which will be used to have a data backup within the company, are organized in mini-batches and are collected and saved on storage with a specific time interval.
The tools used to perform this latter operation are Databricks, the Data Factory and the Data Lake.
In particular, the data are organized in suitable structures thanks to the use of Databricks notebooks, that runs on a cluster, specifically created for this purpose.
At this point the Data Factory comes into play. Thanks to Data Factory, it is possible to schedule and save this data collection through a pipeline: a trigger is configured, to run at defined time intervals the Databricks notebook, passing as parameters the current timestamp and the duration of the interval to be analyzed.
This is the process that leads to store data, partitioned by year/month/day/hour, on the Data Lake which allows us to store a long-term data backup.

5.3 Real-time path

Real-time path is the path used to process data from connected vehicles in real time. These are data packets sent from the connected vehicles with high frequency, which will be processed to analyze them laterand to allow the user to view the evolution in real time, through graphical interfaces.

Hence the choice to base the architecture of the new applications on Docker and Kubernetes technologies, writing them in C# on the ASP.NET Core environment. Regarding the Databricks notebooks used for data analysis, it was all developed using Python language and its libraries.
The Figure 5.2 illustrates in detail the various tools and blocks that make up the parsing, storage, analytics and visualization phases of this path, which we will analyze in the next sections.

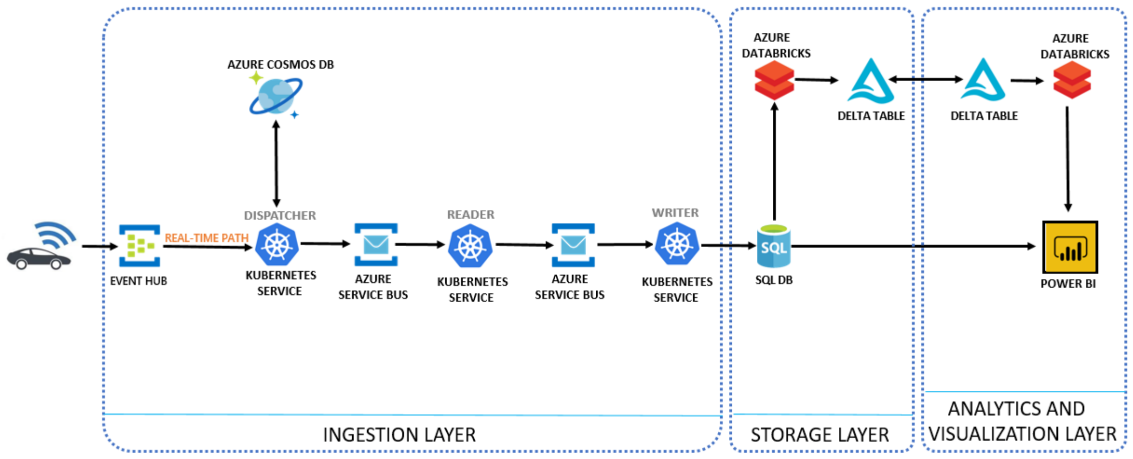


Figure 5.2: Focus on real-time path architecture.

5.3.1 Ingestion Layer

Upstream of the ingestion phase, there is the event hub which uses a publisher/consumer model to receive and transmit messages to the next phase. Once the publisher publishes the data received from the connected vehicles, the event hub stores the data internally in partitions, using a data buffer. At this point, the consumer reads the data from the corresponding partition and sends it to the next step to continue processing.
Received messages contain a series of important metadata , as well as a Header and a Payload, that is a JSON file to be deserialized.

Among the fields of the Header we must mention the following:

EnqueuedTimeUTC: is a timestamp indicating the moment when the message from the vehicle board started.

UserProperties: is a field similar to a dictionary, with key and value, which contains various indications including the data flow type,, some tags that discriminate the application behavior and so on.

SystemProperties: also in this case, it is a field that maps a key with a value and contains a message ID and other technical information related to the connectivity.

Regarding the Payload fields, in addition to the data that identifies the activities and properties of the vehicle, it is necessary to mention the presence of a field that contains a URL that indicates where to retrieve the file, some flags that suggest the behavior to the application, a vehicle ID and the type of message.

The message from the event hub is sent to the *Dispatcher*, which is an application written in C# on the ASP.NET Core environment and released using Docker and Kubernetes. Its task is to verify some fields present in the *UserProperties* of the Header and in particular a flag that indicates whether or not the Cosmos DB should be queried before continuing. If the information obtained from the Cosmos database indicates a negative result, the message must be discarded and looks forward to receiving another message; otherwise the message can continue and is sent on the correct Service Bus.
Service Bus allows the interaction between applications and services, facilitating the management of messages through simple queues.
At this point *Reader* and the *Writer*, which are always applications written in C# on the ASP.NET Core environment and deployed using Docker and Kubernetes, come into play.
Reader reads the message from the Service Bus, whose address is set internally by code using a connection string. It parses the message, transforming the raw message into a tabular format. Once the *Reader* has done the message parsing, it sends this message on the Service Bus to which the Writer is connected to read.
After taking the message in charge, the Writer has instructions in its code to proceed with writing the message in a database table. This writing, obviously, presupposes the creation of the structures and fields of the database table, in order to guarantee compatibility with the message to be written.

Implementation

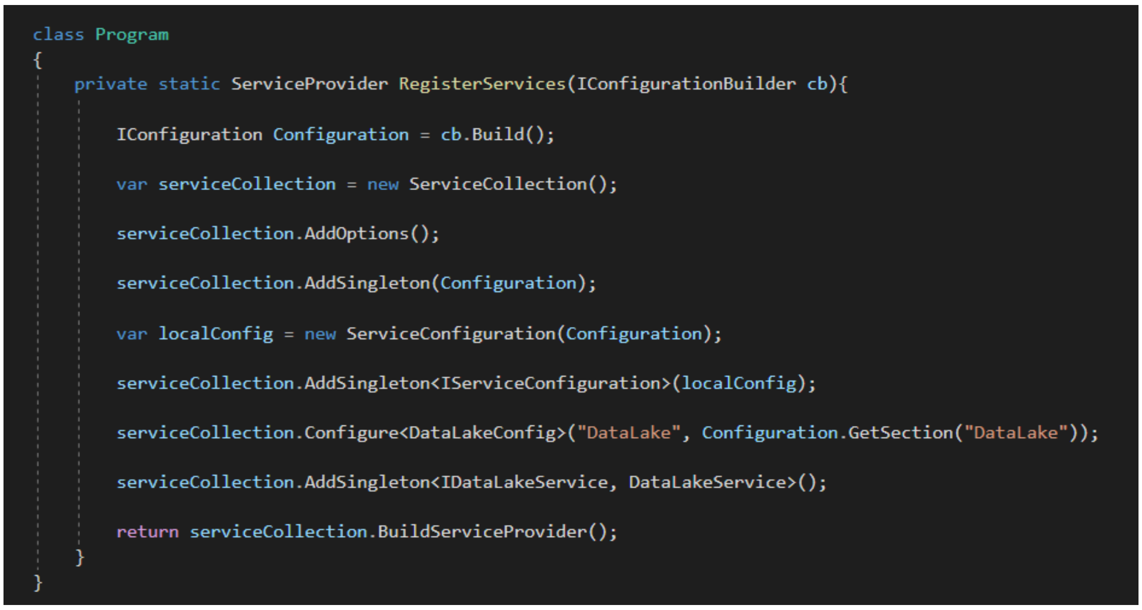
In the development of the *Dispatcher*, *Reader* and *Writer* applications the same programming pattern was followed, called Dependency Injection (DI).
ASP.NET Core supports the dependency injection software design pattern, which is a technique for achieving Inversion of Control (IoC) between classes and their dependencies.
A dependency is any object required by another object, and programmatically realizing them through the new () operator can cause problems.
If a class A uses another class B, then A depends on B.
A cannot do its job without B and A cannot be reused without also reusing B.
This can cause problems because to replace class A with a different implementation, class B should be modified.
This shows how dependencies are a negative aspect, especially in the case of large projects where the configuration would be difficult to keep under control. They reduce reuse which instead has a significant positive impact on the speed of development, the quality and the readability of the code.
The Dependency Injection pattern solves the problems just described by "injecting" the dependencies into the class constructor.

The approach followed by this pattern is as follows:

- an interface is used to abstract the implementation of dependencies.

• the dependency is registered in the *IServiceProvider* service container provided by the platform.

• the framework will create the instance of the dependency to be injected to the manufacturer and eliminate it when it is no longer needed.



```
class Program
{
    private static ServiceProvider RegisterServices(IConfigurationBuilder cb){

        IConfiguration Configuration = cb.Build();

        var serviceCollection = new ServiceCollection();

        serviceCollection.AddOptions();

        serviceCollection.AddSingleton(Configuration);

        var localConfig = new ServiceConfiguration(Configuration);

        serviceCollection.AddSingleton<IServiceConfiguration>(localConfig);

        serviceCollection.Configure<DataLakeConfig>("DataLake", Configuration.GetSection("DataLake"));

        serviceCollection.AddSingleton<IDataLakeService, DataLakeService>();

        return serviceCollection.BuildServiceProvider();
    }
}
```

Figure 5.3: Dependency Injection pattern example.

Figure 5.3 shows a piece of code used in the project that exploits the dependency injection pattern.

In particular, the code in Figure 5.3 shows how the ServiceCollection instance was created and after obtaining the section relating to the Data Lake from the configuration json file, how the corresponding service was hooked.

Going back to the actual implementation, the three applications have classes that perform similar tasks and other classes that perform the specific tasks for which each application has been designed.

The main classes that the three applications have in common are the following:

Program.cs: it is the entry point of the program, it contains the recovery of information and configurations related to access credentials of the individual components involved. It also takes steps to instantiate the corresponding services and the Processor.cs class;

Processor.cs: it is the class that manages the messages arriving on the Service queue Bus. The services that will be used in the various steps are included in its constructor. The execution of its main method consecutively creates and launches the methods of the class FirstClass.cs ;

FirstClass.cs performs different tasks in the three applications:

• *Dispatcher* application: in this case FirstClass.cs has the task of controlling some fields of the message sent by the vehicle, in particular it checks the metadata and properties of the message received. Among these fields there is the flag that indicates whether or not it is necessary to perform an additional query on the Cosmos DB.
If it is not necessary to check the database ,it proceed, otherwise it is necessary to evaluate the answer of the query for understand if the message should be discarded or not.

• *Reader* application: in the case of the Reader, FirstClass.cs has the task of deserializing the payload, going to analyze the various fields.

• *Writer* application : in Writer the FirstClass.cs has the task of writing the deserialized message to the SQL database, compatible with the data structures present in the database.

5.3.2 Storage Layer

The storage layer represents the part of architecture that takes care of storing data so as to guarantee, using queries as specific as possible, the correct display of the results .
This layer involves saving data both on a relational database and on tables that rely on the Data Lake, the so-called delta tables, which are optimized for saving data to be analyzed using Databricks notebooks.
The first step is to create the structures on the SQL database, in particular tables with the fields and relational constraints set in a compatible way with the data that the Writer will write. Already in this first step, in which the data is written to the database, it is possible to query the tables to get an early view of the data that will be exposed to the customer.
Once the structures of the database tables have been set, it is possible to proceed to the population of delta tables that will be useful in the data analysis phase. To populate delta tables, a Databricks notebook written in python is used. This notebook imports the data, present in the database tables, on delta tables and historicises them in order to have the data ready to perform analyzes on them.

5.3.3 Analitycs Layer

This layer is responsible for analyzing the data, previously stored in tabular format in delta tables. In the data analysis phase, Databricks notebooks have been used as a development tool, which integrate very well with delta tables. Python was used as a language to develop the code, resorting to its multiple libraries both as regards the implementation of the algorithms and as regards the visualization of the graphs with the results deriving from the analysis.
Databricks notebooks used for data analysis retrace the analysis in all its steps.
For now let's focus on this general overview, in the next chapter a more detailed

focus on this layer will be made, illustrating in more detail the use case taken into consideration and the results obtained.

5.3.4 Visualization Layer

The visualization layer has the task of illustrating the results obtained from the previous steps, through the use of visual elements.The goal was to provide the customer with two dashboards developed using Microsoft PowerBI. These two dashboards were designed like this:

• a first dashboard that contains tables with previously deserialized data and saved on the database and a series of filters that allow the customer to view the data of his interest (for example by filtering by time period or by vehicle ID or by using others as a filter fields describing the characteristics of the vehicle). This dashboard has the purpose of allowing the customer to have a complete view of the processed data and examine the vehicle data according to their needs in an interactive and simple way. Figure 5.4 shows the dashboard template, the data has been obscured for corporate privacy reasons.

• a second dashboard, on the other hand, is geared towards displaying information from data analysis. In particular, this dashboard contains reports that allow to label vehicles by mission and understand if the vehicle is working or not working.

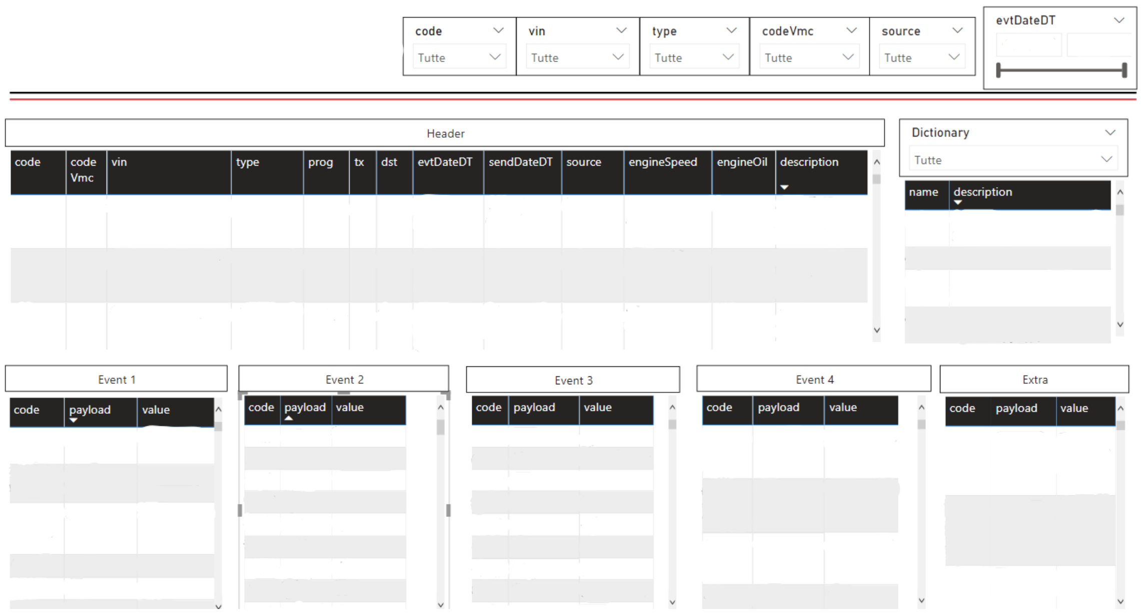


Figure 5.4: Dashboard template.

Chapter 6

Use case

6.1 Focus on the Analitycs Layer

In this chapter, data analysis topic will be explored. In particular, the analytics phases will be illustrated starting from the initial phase of data set building up to the results obtained for the use case considered.

6.1.1 Dataset

Since there was no data set ready for the application of the analysis models, a first phase of analysis and identification of the data useful for the purpose of the research was necessary. A reliable and consistent dataset is one of the essential steps in this type of data analysis experiments.
We collected data from connected vehicles, until the number of vehicles present in our dataset was both consistent in numerical terms and well representative of the categories of vehicles, and these were saved in tabular format on the delta tables.
The delta tables allowed us to easily interface with Databricks notebooks. Given the large number and different types of vehicles under monitoring, it was decided to focus the attention of the following experiment on a limited number of vehicles and then extend it in the future. It was possible to select the vehicles of interest, by querying delta tables through the use of SQL queries, directly from the Databricks notebooks.
Once the dataset was built, we proceeded with a pre-processing phase on the data.

6.2 Use case

Once the data collection phase was completed, the data set was further divided to adapt it to the case study considered. As for this experiment, the dataset was limited to a particular category of vehicle model and was analyzed using two clustering algorithms, k-Means and DBSCAN.

6.2.1 Use case description

The goal of the experiment was to identify the relationship and therefore the correlation that exists between two particular heavy vehicle parameters.The parameters

of the vehicles to which I refer are *engine oil pressure* and *engine speed.*
The analysis of these two parameters was suggested to us by engineering experts, who considered this evaluation to be a priority over others. This priority coming from the fact that there are a significant number of alarms relating to engine oil pressure. To check whether these alarms are really problematic, it is useful to analyze the behavior of the engine oil pressure data compared to the engine speed data, in particular for vehicles that have frequently recorded pressure alarms.
This experiment has the following characteristics:

• the analysis is focused on a particular heavy vehicle category model.

• the dataset is made up of data samples belonging to about 300 heavy vehicles, which are also used for different purposes and activities.

• the *engine oil pressure* parameter reports the last value over 10 minutes.

• the *engine speed* parameter reports the average value over 10 minutes.

• the time series of the two parameters were merged,using the vehicle identifier as key, with a time granularity of seconds.

• every vehicle time-series was splitted in chunks by month.This choice because as can be seen from the distribution of points per month in the Figure 6.1, the vehicles are used to carry out different activities according to the time of year. In addition, the chunks with less than 50 records were filtered.

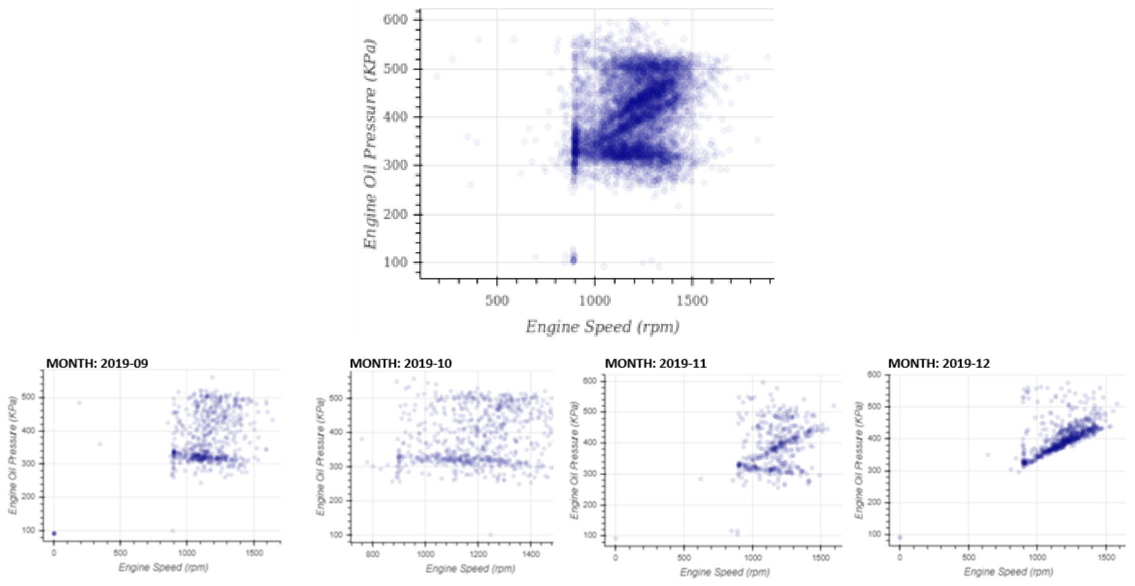


Figure 6.1: The larger image above is the overall representation of all the frames below, which represent the joint distribution of engine oil pressure and engine speed for one vehicle for each month.

• the model evaluation criteria taken into consideration were the Silhouette score and the graphics interpretation of results.

6.2.2 Pre-processing

Generally, before applying any data analysis model, a pre-processing phase of the collected data is necessary in order to make the initial dataset suitable for the application of data analysis techniques. Let's now go in the detail of the pre-processing processes applied.

Data cleaning

The term "data cleaning" refers to those processes aimed at improving the quality of the data present within a dataset to facilitate subsequent analysis.
In the dataset built, there were some fields of our interest that were not valued, some vehicles that have too few records and some outliers. Since these cases of empty fields occurred rarely and were in negligible numbers, it was decided to delete them to clean up the data set and improve its quality, the same for vehicles with fewer than 50 records. Even the outliers have been deleted to try to have clearer relationships between the parameters analyzed.
The following figures (Figure 6.2 and Figure 6.3) illustrate for both *engine oil pressure* and textitengine speed a box-plot by which it is possible to identify outliers, the black dots in the box-plots, and the probability density distribution, after removing this anomalous values.

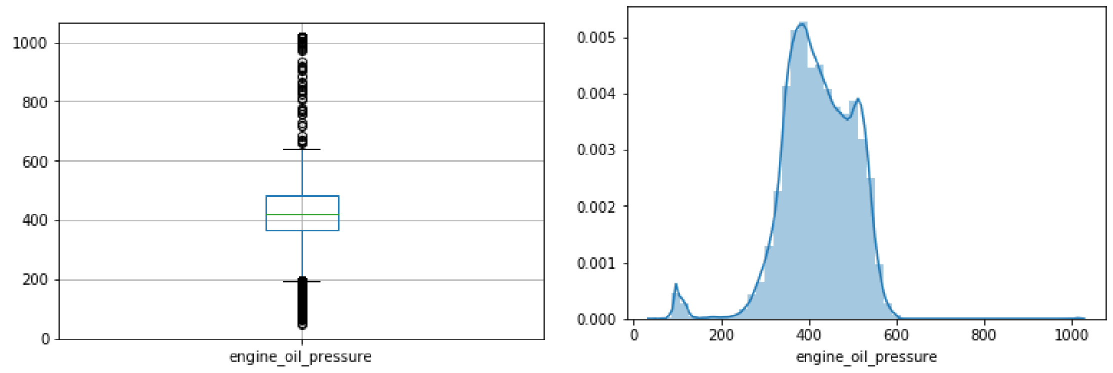


Figure 6.2: On the left the engine oil pressure box-plot, on the right the probability density distribution, considering only the not zero oil pressure values less than 600.

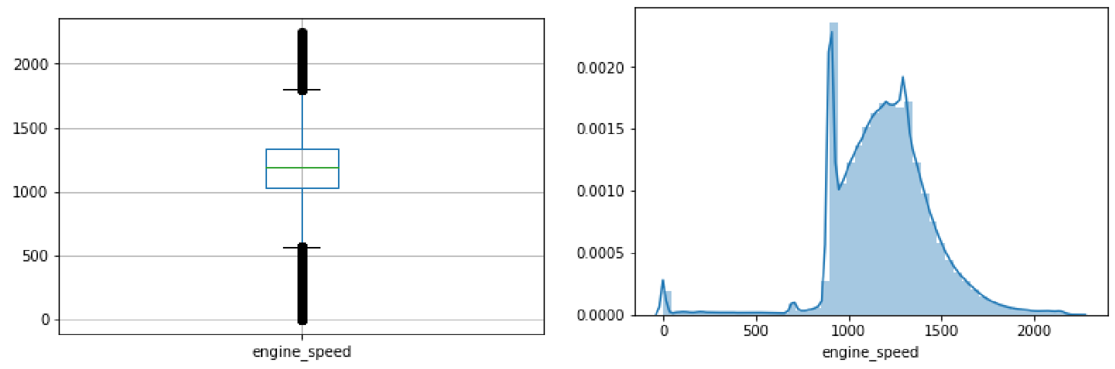


Figure 6.3: On the left the engine speed box-plot, on the right the probability density distribution, considering only the not zero engine speed values less than 2500.

Data Standardization

Another important aspect of pre-processing has been that characterized by the "data transformation" phase, the phase in which an attempt is made to make the dataset values homogeneous with each other for analysis purposes.
In particular, the data standardization was carried out, which has the aim of attributing a variable distributed according to an average and a variance, to a variable with a "standard" distribution, i.e. a variable of zero average and variance equal to 1.
The goal of this standardization was to report the numeric values of the attributes present in the data set on a common scale, since numeric values with large difference intervals were present in the dataset.

Dimensionality reduction

Dimensionality reduction is the process of reducing the number of random variables considered, obtaining a set of principal variables.
Given our dataset, it was necessary to transform the data from the high-dimensional space to a space of fewer dimensions.
In order to have a visual representation of the data a 2D-dimensionality reduction has been applied.
For this purpose, it was decided to use the **t-SNE** technique (t-distributed stochastic neighbor embedding), proposed for the first time by G Hinton and L. Maaten in 2008 [20].It is a non-linear technique for reducing data dimensionality and it is particularly suitable for embedding high-dimensional datasets in a two or three-dimensional space. The algorithm models the points so that close objects in the original space are close in the space with reduced dimensionality, and distant objects are far away, trying to preserve the local structure.

The algorithm is structured mainly in two phases:

• in the first phase, a probability distribution is provided which in each pair of points in the original space (in high dimension) associates a high probability value if the two points are similar, low if they are dissimilar.

• in the second phase, a probability distribution similar to that of the first phase is defined, in the reduced size space, minimizing the Kullback-Leibler divergence of the two distributions and rearranging the points in the reduced size space.

For the implementation the Python library sklearn.manifold.TSNE has been exploited, by setting 2 as the size of the embedded space.

6.2.3 Vehicles clustering

After the preliminary steps just described, to make the dataset suitable for our analysis, we proceeded with the vehicles clustering.
In this step, k-Means and DBSCAN were compared, two clustering techniques which as explained in chapter 3 are based on different principles.

As a first step, the algorithms were tested by choosing to group the data using different values as the number of clusters.
The choice of the number of clusters that best suited our data was calculated using the silhouette coefficient, considering configurations that used from 0 to 15 as possible number of clusters. In particular, we looked for the point where the increase in the number of clusters caused a very small decrease in the score, while the decrease in the number of clusters sharply increased the score. From the results the number of best clusters for our datset was 4.

k-Means algorithm did not give good results as it failed to recognize the different behaviors. In particular, it tended to divide the data, that from a graphical observation seemed to belong to a single cluster, in more than one subset.
In this case the typical problems of this algorithm have arisen. As in the chapter 3 announced, it does not work well on non-globular data structures, itis very sensitive to the presence of noise in the data, it tries to create clusters of the same size and it does not take into account the density of the points present.

Given the k-Means unsatisfactory results, to identify the clusters representing the high density regions, DBSCAN was applied.
As explained in chapter 3, the concept of density of the DBSCAN is based on two fundamental parameters, MinPts and eps. In this experiment MinPts value was set to 10, while the eps value was set equal to 1.The Figure 6.5 shows the resulting group divided by color.

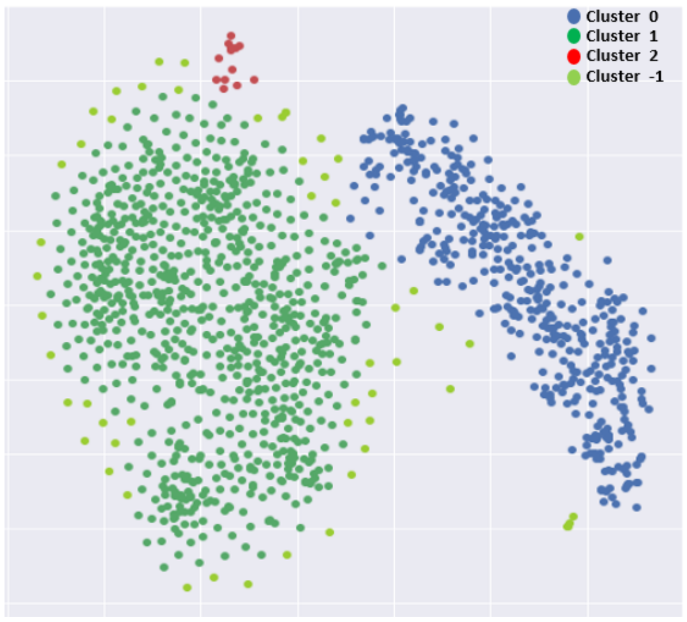


Figure 6.4: Graphical representation of DBSCAN clustering (eps=1, MinPts=10).

To find a mathematical relationship linking the two variables analyzed, it was decided to study the behavior for each cluster. In Figure 6.5 each plot represents the average behavior of the *engine oil pressure* over the *engine speed* in the relative cluster.

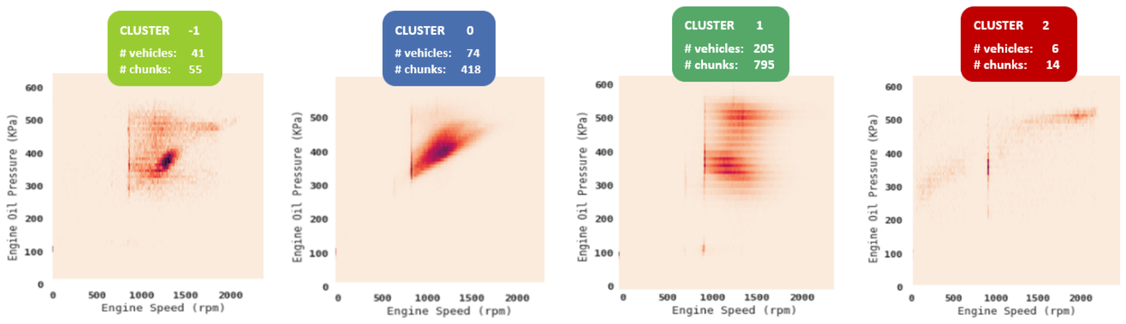


Figure 6.5: Graphical representation of the behaviours of each cluster.

Analyzing all the sub-figures of the Figure 6.5 it is possible to observe that:

• **Cluster -1**, which is the light green one in Figure 6.4 representing noise, has an undefined behavior. In fact it is possible to observe a well-defined set of points in the center and a lot of noise around.

• **Cluster 0** appears to have an almost linear behavior. As the engine oil pressure increases, the engine speed increases.

• **Cluster 1** seems to represent a linear regression in reverse, that is with negative angular coefficient . In reality, if you look closely at the graph, there seem to be two straight lines and not one, separated by noise.

• **Cluster 2** seems to demonstrate that there is an almost logarithmic relationship between engine oil pressure and engine speed.

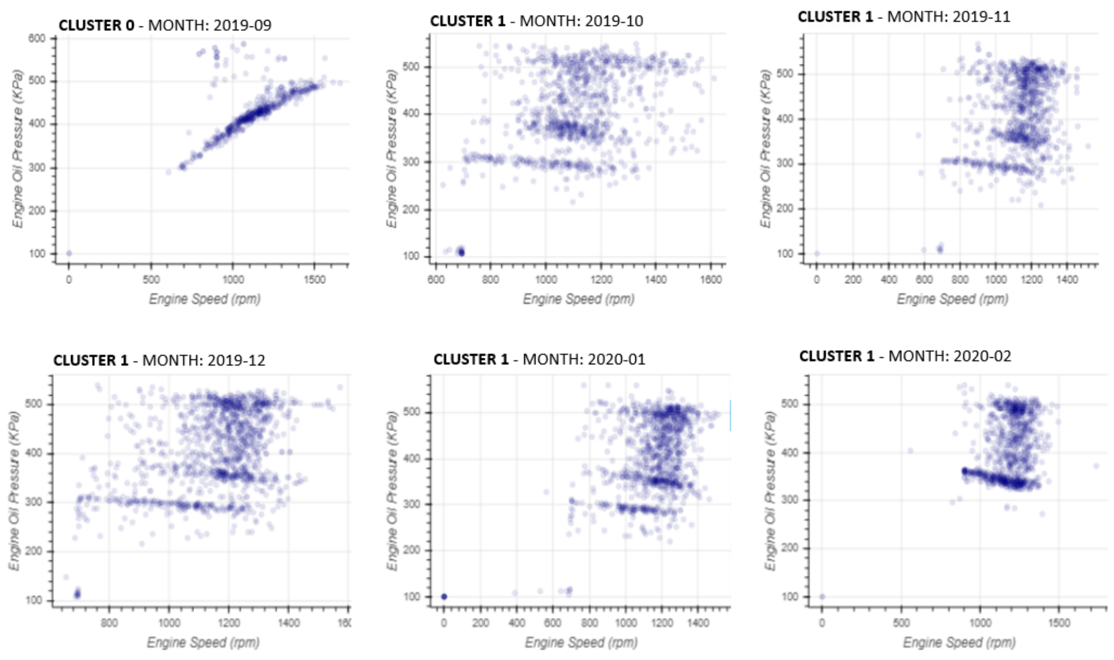


Figure 6.6: Example of the behavior of a vehicle, for a period of six months.

By studying how the relationship between *engine pressure oil* and *engine speed* varies for a single vehicle over a period of 6 months, it was noted that a vehicle that in a month belonged to a certain cluster over time tended to assume a behavior similar to that of another cluster.
The sub-figures in Figure 6.6 show us that a vehicle that in September was associated with cluster 0, over a six-month period tends to increasingly assume the characteristics represented by cluster 1.
This data analysis seems to show how the relationship between the *engine oil pressure* and the *engine speed* behavior can vary, assuming different relationships, based on the activity carried out by the vehicle.
So the relationship between these two variables allows us to understand if a vehicle is carrying out a certain activity rather than another. It is possible to rely on the clusters behavior to understand if an *engine oil pressure* value compared with an *engine speed* value is to be considered abnormal or is lawful for the activity that the vehicle is carrying out.

Chapter 7

Conclusions and Future Developments

The work presented in this thesis was intended to describe the implementation of a pipeline that through various blocks covered various stages, from the ingestion of data flows coming from connected vehicles until the data analysis and the visualization of data and results. By going through the analysis of the different infrastructure solutions, On-Premise, On-cloud, Hybrid, we have come to understand why more and more companies choose to migrate to the cloud-based solution.
The design and development process gave way to explore new technologies and tools and to understand how and when to use them.
In particular, the tools provided by the Microsoft Azure platform have proved to be simple, versatile and suitable for cases of real-time flows. Delta tables represented a fundamental tool for interfacing with the data to be analyzed through Databricks notebooks.
A special mention also goes to Docker and Kubernetes, in fact, thanks to the interaction of these two tools, it is no longer necessary to oversize the machines to be able to manage heavier workflows, but you have the option to scale quickly based on workloads.

The developed solution has allowed a series of advantages for the company both from a performance point of view and from a cost point of view compared to previously developed On-Premise solutions.
The goal of processing data in real time, keeping latency times low, has been achieved, in fact the latency between sending a message from the connected vehicle board to saving the data already deserialized on storage was very low, of the order of the seconds.
The phases of data analysis, in particular the use case described, allowed to find a correlation between events and to give an explanation to certain phenomena that occurred with high frequencies.

7.1 Future Developments

All the implemented solutions provide for future improvements and insights of a different nature.
In the future, it is expected to manage new real-time data flows, using the architec-

ture and all the applications implemented that cause the data to land on the same instance of Data Lake, under different paths based on the type of information.
Thanks to the high interaction capacity between the services offered by Microsoft Azure, it will be possible, using the Data Lake as its source, to carry out new analyzes, reworkings and rewrites of data, in order to adapt it to the customer's needs.
In addition, being the applications modular and flexible,it will be possible to add new functions within them without affecting the existing sections of code.

Besides this, the customer, satisfied with the results obtained, seems to be very inclined to invest more on data analysis in the future.
In addition to delve in to analyzes already made using other algorithms and new points of view, we will also explore other fields of interest to be able to do predictive maintenance and anomaly detection on vehicles.

Predictive maintenance will allow the customer to predict future vehicle failures. It will be possible to monitor the vehicles in real time, sending continuously data to the machine learning model, which will be able to promptly detect and report how to intervene to avoid or minimize the percentage of future failures.
As far as anomaly detection is concerned, the objective is to understand whento consider the values of one parameter or several parameters of the vehicles as if they were anomalies and when not.

The data visualization part will also be improved, implementing other dashboards and enriching existing dashboards with new features, to improve the contents and customer's visual experience .

Appendix A

Appendix

A.1 Citations

[0] A. Merv, Teradata, 11(1) (2011) 1.

[1] Big Data: The Next Frontier for Innovation, Competition, and Productivity,McKinsey Global Institute, New York, 2011.

[8] Aurélie Géron, "Hands-On Machine Learning with Scikit-Learn and TensorFlow Concepts, Tools, and Techniques to Build Intelligent Systems", 2017.

[9] J. McCarthy, E. Feigenbaum "In Memoriam Arthur Samuel: Pioneer in Machine Learning", AI Magazine, vol. 11, no. 3, 1990.

A.2 Images

[2] https://medium.com

[10] https://machinelearningcongress.blogspot.com

[11] https://medium.com

[13] https://www.slideshare.net

[14] Schlitter, Nico Falkowski, Tanja Laessig, Joerg. (2011). DenGraph-HO: Density-based Hierarchical Community Detection for Explorative Visual Network Analysis. 10.1007/978-1-4471-2318-7_22.

[15] https://docs.microsoft.com

[16] https://diveintodocker.com

[17] https://dzone.com

[18] https://www.redhat.com

A.3 Bibliography

[3] Metternicht G., Mueller N., Lucas R. (2020) Digital Earth for Sustainable Development Goals. In: Guo H., Goodchild M., Annoni A. (eds) Manual of Digital Earth. Springer, Singapore

[4] Hammer CL, Kostroch DC, Quiros G (2017) Big data: potential, challenges, and statistical implications. IMF staff discussion note, SDN/17/06
https://www.imf.org/en/Publications/Staff-Discussion-Notes/Issues/2017/09/13/Big-Data-Potential-Challenges-and-Statistical-Implications-45106. Accessed 19 Jun 2019

[7] F.T.Liu, K.M.Ting, Z-H.Zhou, "Isolation Forest", 2008 Eighth IEEE International Conference on Data Mining, Pisa (Italia), Dec 15-19, 2008, pp. 413-422, DOI 10.1109/ICDM.2008.17

[12] P.K.Agarwal, N.H.Mustafa, "k-means projective clustering", PODS '04 Proceedings of the twenty-third ACM SIGMOD-SIGACT-SIGART symposium on Principles of database systems, Parigi (Francia), Jun 14-16, 2004, pp. 155-165, DOI 10.1145/1055558.1055581

[19] A.Sari, "A Review of Anomaly Detection Systems in Cloud Networks and Survey of Cloud Security Measures in Cloud Storage Applications", Journal of Information Security, Vol. 6, No. 2, April 2015, pp. 142-154, DOI 10.4236/jis.2015.62015

[20] Maaten, Laurens van der e Hinton, Geoffrey (2008). Visualizing data using t-SNE. In: Journal of Machine Learning Research 9.Nov, 2579–2605.