POLYTECHNIC OF TURIN

Master's Degree Dissertation

in

Automotive Engineering

DEVELOPMENT OF MACHINE LEARNING TECHNIQUES FOR THE CREATION OF NEW FAULT PREVENTION ALGORITHMS

Academic Supervisor:

Prof. Stefano D'Ambrosio

Company Tutor:

Nicola Rao

Candidate:

Rocco Leopardi

ACADEMIC YEAR 2022 - 2023

POLYTECHNIC OF TURIN

Master's Degree Dissertation

in

Automotive Engineering

DEVELOPMENT OF MACHINE LEARNING TECHNIQUES FOR THE CREATION OF NEW FAULT PREVENTION ALGORITHMS

Academic Supervisor:
Prof. Stefano D'Ambrosio

Company Tutor:
Nicola Rao

Candidate:
Rocco Leopardi

ACADEMIC YEAR 2022 - 2023

Table of Contents

List of Figures

List of Tables

Abstract

The aim of this dissertation is to bring together the work carried out on the development of machine learning techniques for the creation of new fault prevention algorithms within Iveco heavy range vehicles. In particular, the Additional Water Heater is considered: the goal is to foresee its potential lock twelve hours in advance.

The dissertation is structured in seven chapters and relative paragraphs, which order is in according to the actual path followed during the work development.

In “Chapter 1”, an overview of Iveco is offered both from a historical and organizational point of view. In “Chapter 2”, all the features related to the new generation vehicles are described, stressing the role of telematics and connectivity made possible by a new electrical system. In “Chapter 3” the Additional Water Heater is described: an overview about its components is given and the working logics are analysed.

In “Chapter 4” the basics of machine learning theory are introduced: several issues underlying the field are addressed, along with the techniques involved. In “Chapter 5” machine learning techniques are applied to an actual case study: the Additional Water Heater. Exploratory data analysis is carried out on data coming from telematics in order to get insights, and then a model has been built in order to forecast the component lock twelve hours in advance. In “Chapter 6” the achieved results are presented along with some possible improvements.

Finally, in “Chapter 7”, some Python code lines are reported: only the most important customized functions or ad-hoc methodologies are considered in order to show how the obtained results are actually achieved.

The presented work is entirely attributable to the author of the dissertation: the discussed analysis and the entire algorithm are the result of a six-months internship during which the author played a crucial role in the development of the project. The author was constantly assisted by the Iveco Team to integrate the obtained results into the ecosystem and to undertake technical analysis directly on the vehicles.

1 Iveco

1.1 Historical Background

IVECO S.p.A. (Industrial Vehicles Corporation) is an Italian transport vehicle manufacturing company based in Turin and owned by CNH Industrial.

Figure 1 IVECO logo

The core business consists in designing and producing light, medium and heavy duty commercial vehicles, activity that is managed since 1975. On 1 January 1975 IVECO was incorporated with the merger of five different brands: FIAT Veicoli Industriali, OM, Lancia Veicoli Speciali, Unic and Magirus-Deutz. Therefore, since the beginnings a global vision with European roots was the basis of the company: a global visibility was achieved during the first years, allowing a complete restructuration of its organization and the launch of new innovative products such as the TurboStar. The TurboStar was produced from 1984 to 1993: with more than 50,000 units produced, it was one of the best-selling Iveco trucks in Italy. The success was mainly due to the engine performance.

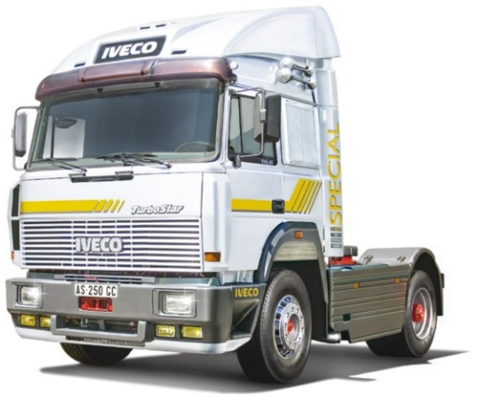


Figure 2 IVECO TurboStar

Initially, two configurations were available: 6-cylinder in-line 13,8 L supercharged with intercooler and 330 hp for the former, 8-cylinder V-turbo (without intercooler, 17,2 L) and 420 hp for the latter. Both versions could be fitted with either mechanical or ZF ECOSPLIT 16-speed synchronised gearboxes or Fuller 13-speed quickshift gearboxes. In 1987 the smaller series was replaced by the engine boosted to 360 hp and further uprated in 1990 to 377 hp, followed in 1989 by the 480 hp turbo intercooler version with ZF gearbox. This was followed in 1989 by the 480hp turbo intercooled version with ZF gearbox. The whole series was then replaced by the Iveco Eurostar presented in 1993, produced by IVECO from 1993 until 2003. Both 6-cylinder and 8-cylinder configurations were available with engine powers between 375 hp (9.5 L) to 514 hp (13.8 L).

In the late 1990s, a milestone can be set: a new way of thinking was introduced, and the costumer becomes the centre of the strategy defining a new approach to the company's activities: these innovations are concatenated to the businesses expansion around the world and with the introduction of new important products related both to engines field and commercial one. Moreover, from a technical point of view it is worth noting the introduction of Cursor engines (between 1998 and 1999) that met Euro 2 and later Euro 3 standards opened the door to a new era of vehicles, of which the Stralis would be the future spokesman in 2002.

The 21st arrived, as many new developments that revolutionize the ranges: the new Daily, the Eurocargo and the Stralis constitute the IVECO's backbones on which the brand's success of the first decade of 2000 is based on. The IVECO Stralis is a heavy-duty truck produced 2002: it covers the range above the Eurocargo (between 19 and 44 tonnes) and replaced the EuroStar [1].

Figure 3 IVECO Stralis

With the support of CNH Industrial, IVECO continues to grow. Important achievements are obtained: the new Stralis Hi-Way has been named the "Truck of The Year 2013" and the New Daily the "International Van of the Year 2015".

CNH INDUSTRIAL

Figure 4 CNH Industrial logo

Moreover, in 2019 the new S-Way was introduced, the 100% connected, driver-centric long-haul truck: it is the first vehicle in the new Iveco Way heavy range developed to deliver a package of features and services focused on the driver, on sustainability and on an advanced level of connectivity. Therefore, the new S-Way marks a strong shift with respect to the previous generation trucks: the services around the product become more important than the product itself.

Figure 5 *IVECO S-Way*

Nowadays, Iveco can be considered a leading manufacturer of commercial vehicles and vans and an international leader in the development, manufacture, marketing, and servicing of a vast range of light, medium and heavy commercial vehicles. Iveco can be considered a major player in the global transport world: the vehicles adopt the latest engineering technologies, applied to a comprehensive range of engines running on diesel and alternative fuels. These include natural gas (CNG), biofuels, hybrid technologies and electric engines.

1.2 Aftermarket Solutions and Control Room

Among the several IVECO's missions, there is the one consisting in protecting the value, performance and productivity of the vehicles over time: with MYIVECO WAY SOLUTION, an integrated transport solution, including also services that can be tailored on the specific customer and mission is offered. These services are fully integrated into the ecosystem in order to exploit digitalisation, automation, and vehicle electrification to anticipate and meet the demands [2].

Within this service it is fundamental to ensure the best maintenance, and skilled experts ready to help the customer whenever he need and high-performance parts: Iveco guarantees that products and suppliers comply with quality standards about raw materials, production processes and supply chain. Iveco knows how important it is to minimise vehicle downtime, and it continues to invest in a growing network of spare parts warehouses and relevant logistic services, including a team of more than 150 people all across Europe to solved cases of urgent parts for a Vehicle Off Road (VOR).

Iveco connectivity helps in maximising vehicle's uptime and productivity through a proactive approach with the Remote Assistance Service tool for remote diagnostic, teleservices, and over-the-air software updates. This service is possible thanks to a remote monitoring done by the Control Room: it analyses real-time data and enables Iveco specialists to monitor vehicles round-the-clock to maximise uptime and productivity through a proactive approach [3].

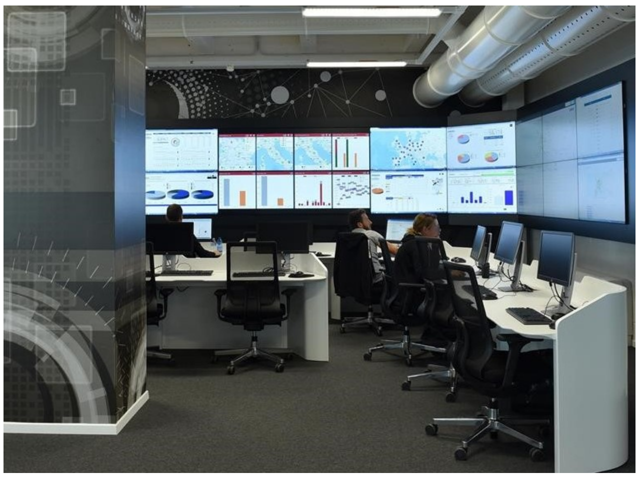


Figure 6 *Iveco Control Room*

In particular, fault code alerts are received if some warning about a component arises, they are analysed and then a planned stop at workshop is organized in according to the specific service intervention. Moreover, data coming from the connected vehicles are use also for the "Remote Assistance" and "Assistance non-stop" services: it is possible to request assistance during the stop in case of vehicle issues with the diagnosis that is performed remotely.

It is worth noting that Control Room's work aims at identifying patterns able to recognize potential breakdowns with a precision higher than 95%: both algorithms development and specialization in workshop field operations are needed in order to recognize the issue and develop repairing procedures. Therefore, a constant collaboration with other departments is necessary: among the departments involved in the ecosystem it is possible to identify the Telematics Quality, C&CSV Digital, P&CS Markets, etc.

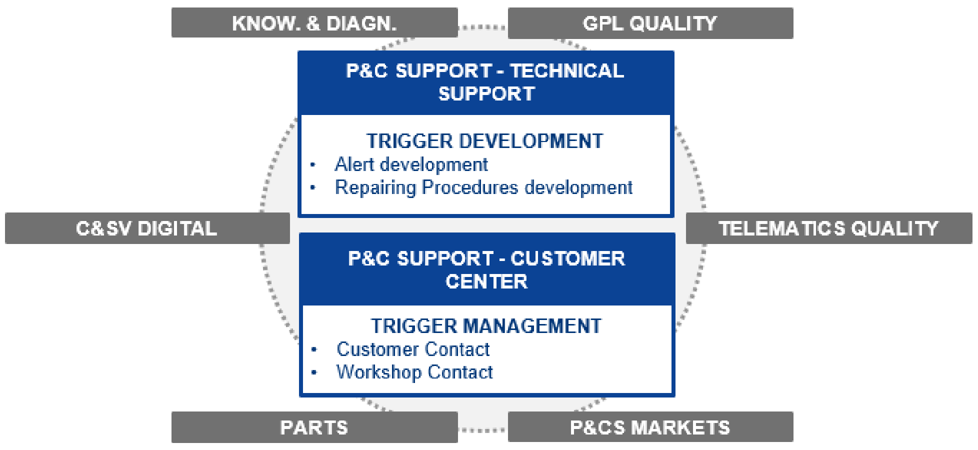
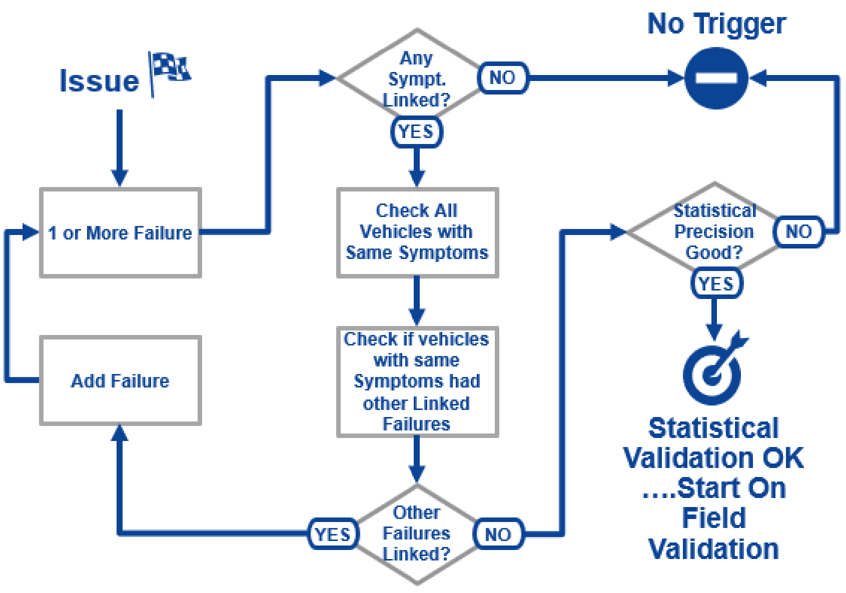


Figure 7 *Iveco Departments involved in the ecosystem*

Among the several activities carried out by the Control Room it is possible to identify the most relevant:

- Alert Development: standard or complex; the final output is called Trigger, and it is the results of vehicles symptoms study, previous breakdowns analysis and statistical analysis. It is worth noting that an empirical process could be adopted exploiting the experience from field (inverse engineering): this would be a reactive approach that is very accurate but also time consuming (a high time to market is needed);



Issue
1 or More Failure
Add Failure
Any Sympt. Linked?
NO
YES
No Trigger
Check All Vehicles with Same Symptoms
Check if vehicles with same Symptoms had other Linked Failures
Other Failures Linked?
YES
NO
Statistical Precision Good?
NO
YES
Statistical Validation OK ....Start On Field Validation

Figure 8 Standard trigger development

- Repairing Procedures Development, providing a streamlined repairs service to network;
- Reporting: all the relevant information acquires by the Control Room are spread with all IVECO departments;
- Customer Centre 2nd Level Support.

The work of this dissertation is performed within the complex trigger development: more advanced tools are exploited (mainly related to machine learning techniques and Python code) in order to properly configure complex problems.

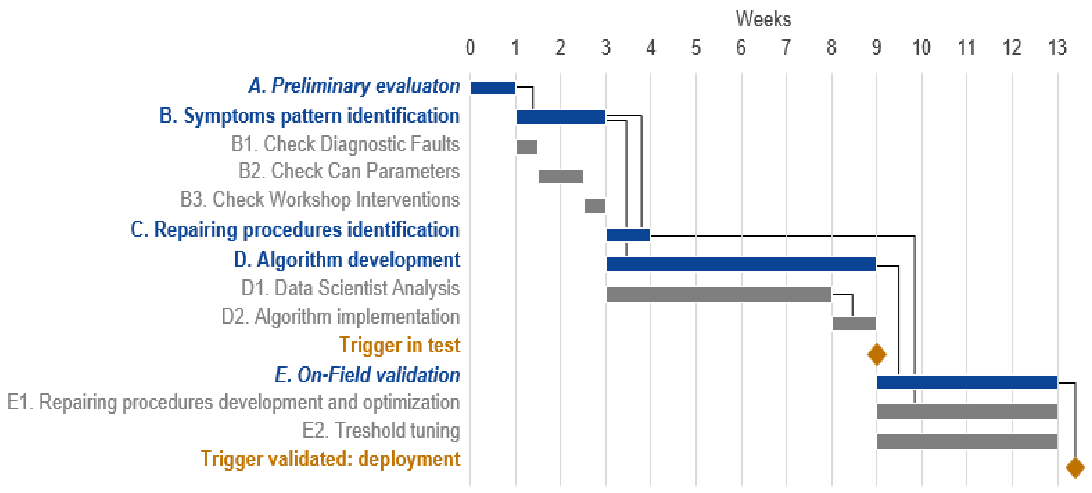


Figure 9 Gantt: chart: complex trigger development

2 New Generation Vehicles

The Iveco S-Way is 100% connected: a superior driving experience is provided, with advanced driver assistance and maintenance services. A new range of services is enabled: driver, fleet owner, dealer and Iveco are constantly connected ensuring a shift from a reactive to a predictive approach to offer predictive maintenance and service planning [3].

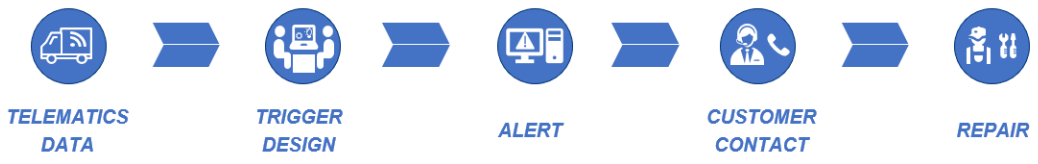


Figure 10 Control Room Process

The S-Way is the first vehicle in the new Iveco Way heavy range developed to deliver a package of features and services focused on the driver, on sustainability and on an advanced level of connectivity. Therefore, the new S-Way marks a strong shift with respect to the previous generation trucks: the services around the product become more important than the product itself.

This is possible by putting in communication the vehicle and Iveco Control Room providing a real-time data analysis and a proactive approach by means of alerts triggered by Diagnostic Trouble Codes (DTCs): a new connectivity and a new telematics system are necessary to achieve this goal, and the starting point is the Electrical System design.

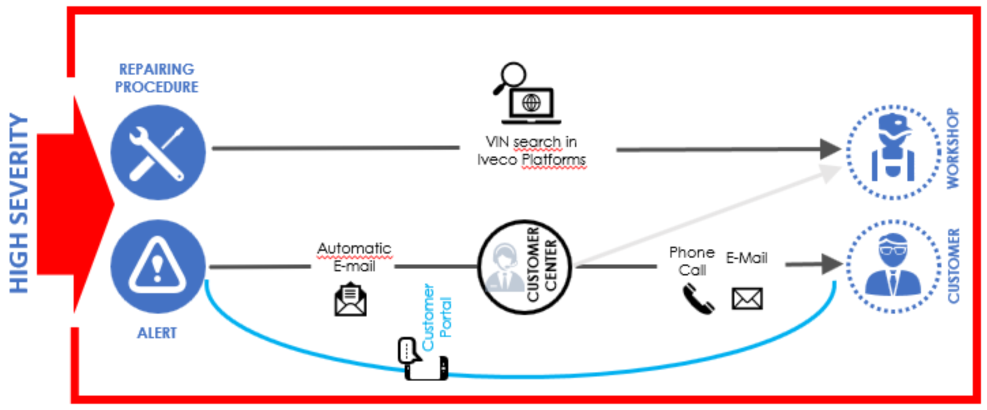


Figure 11 Alert Management

2.1 Electrical System: HI-MUX System Architecture

The Multiplex System (MUX) is based on the creation of a communication network between the various control units that allows a fast and reliable data transfer: each switch transmits its 'state' to a CAN (Databus) communication line and subsequently the necessary voltage is provided to the user. In particular, the Control Area Network (CAN) line is a network protocol for the exchange of data (BUS) between control units (ECU): from the physical point of view, the CAN line consists of two intertwined wires [5].

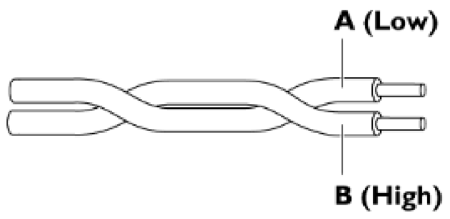


Figure 12 Twisted pair cables

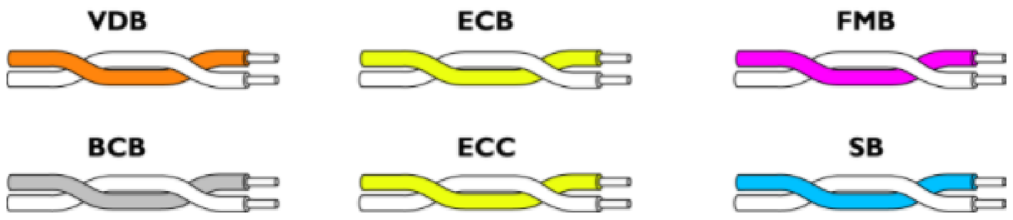


Figure 13 Twisted pair cables for different communication lines

The aim is to greatly simplify wiring compared to a traditional system in which each user must be controlled directly by its switch, with an increase in the total length of the cables and therefore the risk of malfunctions due to wear, defective assembly, corrosion and oxidation [24].

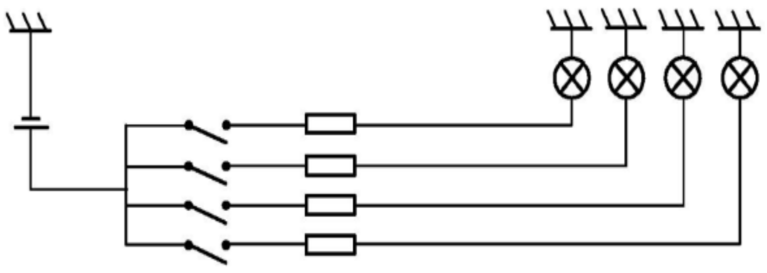


Figure 14 Standard control

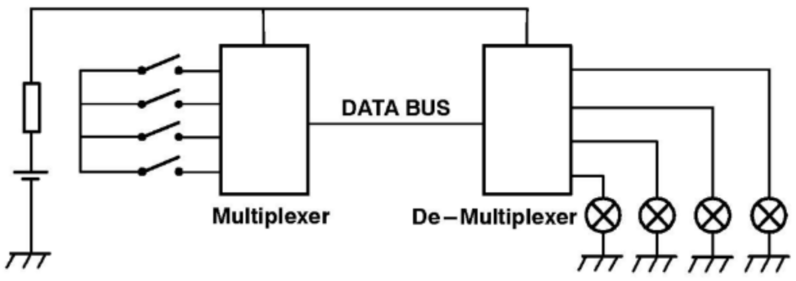


Figure 15 Multiplex control

Therefore, vehicles equipped with Hi-Mux use body computer module (BDM) and frame computer module (FCM) control units and the following CAN lines are used:

1. ***BCB (Body Control Bus)***

   It allows the communication between the hi-mux system power plants and the following control units of on-board services:

   - ***SWI***, Steering Wheel Interface ECU;
   - ***MIRROR,*** Mirror Control ECU;
   - ***AC (CLIMATE),*** Air conditioner ECU;
   - ***EAC (PC)***, Electric compressor and Parking Cooler system ECU;
   - ***AHT-W***, Additional Water Heater ECU;

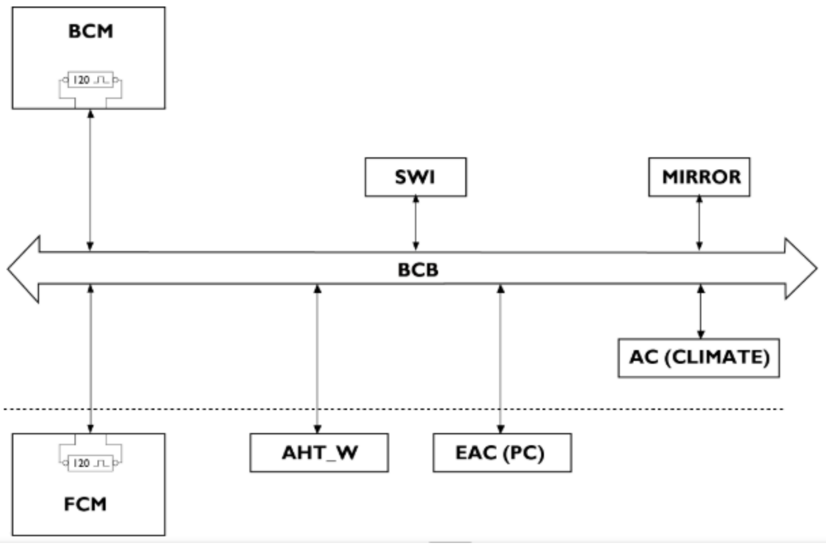


Figure 16 Body Control Bus (BCB) communication line

2. *VDB (Vehicle Data Bus)*

It allows the transfer of information between the various electronic vehicle systems. Among the most important:

- ***BCM,*** *Body Control Module;*
- ***P&CM***, Processing & Connectivity Module;
- ***VCM***, Vehicle Control Module;
- ***DTCO***, Digital tachograph;
- ***IC***, Cluster;
- ***EBS,*** *Electronic Brake System ECU;*
- ***INTARDER;***
- ***ECAS***, Air suspension control unit;
- ***TRAXON***, Automated gearbox control unit Hi-Tronix;
- ***NIS***, Infotainment system control unit.

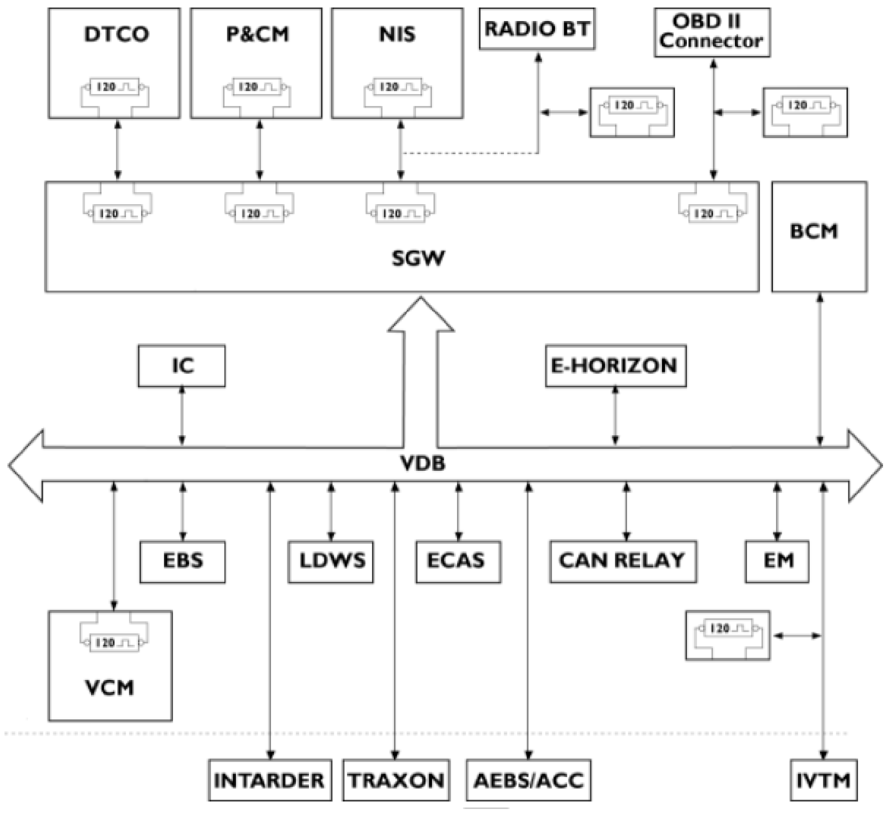


Figure 17 *Vehicle Data Bus (VDB) communication line*

3. ***ECB (Engine Control Bus)***

It allows the communication between:

- ***VCM,*** *Vehicle Control Module* (torque generation request depending on engine revolutions and accelerator pedal position, Cruise Control, engine brake activation, etc.);
- ***ECM***, Engine Control Module;
- ***SGW***, Secure GateWay ECU;
- ***OBD*** connector for On Board Diagnosis**.**

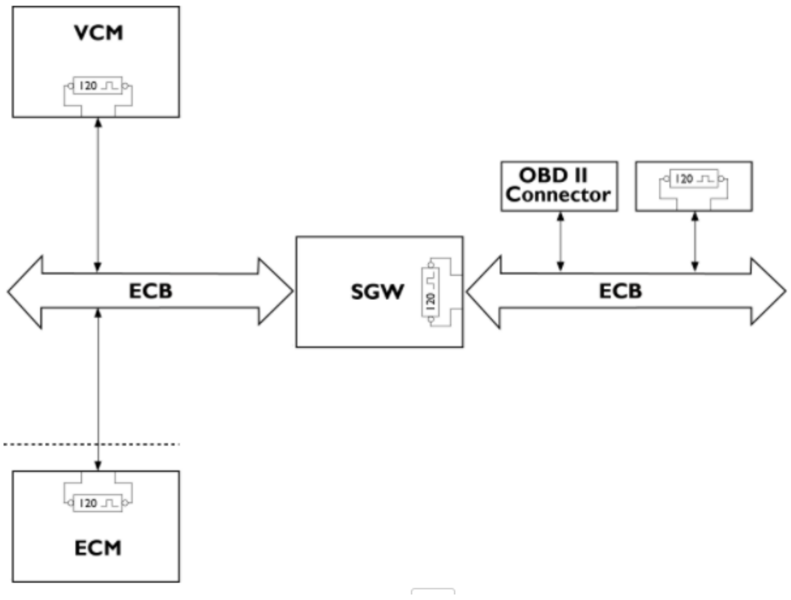


Figure 18 *Engine Control Bus (ECB) communication line*

4. ***ECC (Engine Component Control Bus)***

   It allows the communication between:

   - ***ECM,*** Engine Management Module***;***
   - ***NOx*** sensors***;***
   - ***PM*** sensors;
   - ***UQS,*** urea quality sensors;
   - ***EVGT***, turbine actuator.
   - ***ExFIp,*** throttle engine brake actuator.

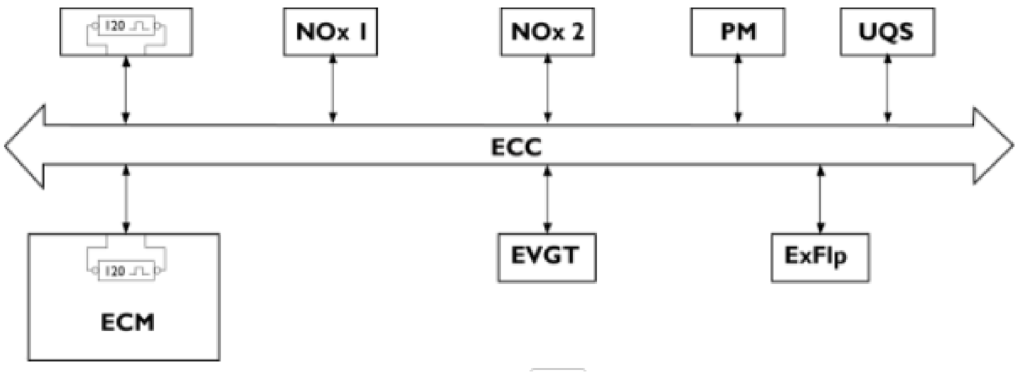


Figure 19 *Engine Component Control Bus (ECC) communication line*

2.2 Telematics: Processing & Connectivity Module

Telematics is the technology used to monitor a wide range of information related to an individual vehicle or a fleet: telematics systems gather several data, process them, and allows the transmission over lengthy distances. Telematics technology developed in the last decade due to the greater availability and practicability of telecommunications technology, and it is adopted because of several benefits. The focus in this dissertation is about fleet maintenance: telematics allows to warn managers of issues with vehicles and equipment. As already stated, this approach allows them to address these problems sooner and reduce the danger of breakdowns. Within this landscape, safety has a crucial role.

All this is possible by means of the P&CM [24]. The P&CM is the control unit through which the acquisition and transfer of data related to the dynamics and behaviour of the vehicle takes place thanks to a deep integration with the electronic system. The P&CM application extracts and stores diagnostic data from the CAN bus of the vehicle and stored in a local database.

Figure 20 *Process and & Connectivity Module*

It is worth noting that normally a vehicle manufacturer is not responsible of the development of all mechanical and electronic technology in-house but relies on external suppliers. For many of these components, the original manufacturer has developed its own diagnosis, but often the vehicle manufacturer already has its own diagnosis logic, which normally does not coincide with that of the supplier.

To overcome this problem, the communication between the electronic control units was standardized: the American Society of Automotive Engineers (SAE) has developed various protocol specifications for self-diagnosis. The most important for the present work is the standard SAE J1939, that is a high-level protocol based on the physical CAN network that recommends practices used for communication among commercial vehicle components. The J1939 data packets contain the actual data and a header that contains an index called “Parameter Group Number”: it identifies a message’s function and associated data, and standard PGNs are defined by J1939. The actual data is described by the SPNs, a number assigned by SAE to a specific parameter within a Parameter Group: an important Parameter Group is DM1, usable by the Instrument Cluster to report the status of the system to the driver.

These concepts are key because there are two important applications installed on the P&CM that work on these data [22]:

1. Diagnostic Logger (gestione DTC DM1 e LAMP);
2. DataCollector (management of CAN parameters defined as "environmental", such as temperatures, pressures, engine revolutions, voltages, etc.).

As for the frequency of sending the collected data, it is fixed in 15 minutes.

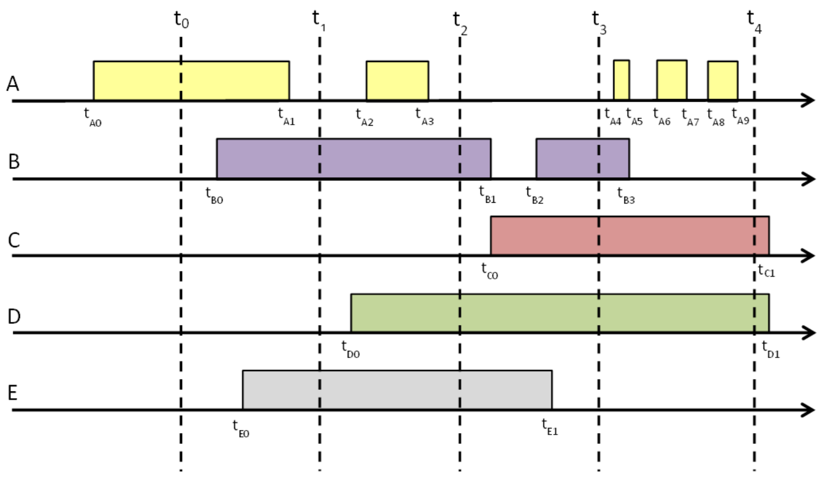


Figure 21 Diagnostic Logger, fault and lamp sequence example

Within this dissertation, DM1 are considered as events consisting in the aggregation of different parameters:

- ***VIN***, an univoque Vehicle Identification Number;
- ***ECU ID,*** an univoque Control Unit Identification Number;
- ***SPN***, Suspect Parameter Number
- ***FMI***, Failure Mode Identifier
- ***OC***, Occurrence Count
- ***Activation and Deactivation epoch;***
- ***Environmental Parameters*** (Engine Speed and Torque, Engine Coolant Temperature, Engine Oil Temperature, etc.).

2.2.1 Autodiagnosis

The complex system behind a vehicle needs to identify and communicate faults to different subsystems: diagnostic messages (DM) are used when a vehicle is repaired as well as during vehicle operation, while the diagnostic trouble codes (DTCs) are used to report potential fault conditions in the system within the SAE JI939-73 standard [5]. Each DTC is characterized by several independent fields that give information about the fault:

- ***SPN*** (Suspect Parameter Number): identifies the component or system on which the anomaly occurred;
- ***FMI*** (Failure Mode Identifier): indicates how the failure occurs;
- ***OC*** (Occurrence Count): is a fault counter; every time fault goes from inactive to active, the OC is incremented by 1 until 126 is reached.

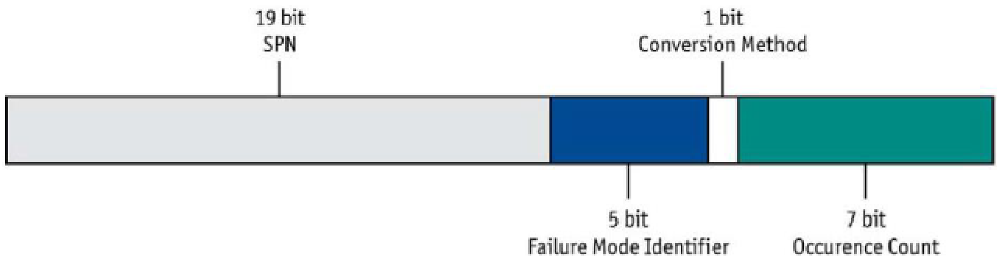


Figure 22 Diagnostic Trouble Code Structure

FMI	Description
0	High – most severe (3)
1	Low – most severe (3)
2	Erratic, Intermittent, or Incorrect
3	Voltage Above Normal or shorted to high fault
4	Voltage Below Normal
5	Current Below Normal or open circuit fault
6	Current Above Normal or Shorted to ground fault
7	System Not Responding Properly
8	Abnormal Frequency, Pulse Width, or Period

Table 1 Failure Mode Identifier partial list

3 Additional Water Heater

3.1 System Overview

The Additional Water Heater performs the function of preheating the passenger compartment and the engine working independently of the latter: its operation is mainly expected with the engine off during stops to ensure, respectively, better driver comfort and/or better engine efficiency during ignition. As the matter of fact, this heating solution provides an efficient working environment even at extreme outdoor temperatures, while helping to reduce engine idling times: cold engines are not efficient, increase wear phenomena and reduce the useful life of engine components, leading to higher fuel consumption and higher costs.

In order to fulfill the requirements, the Additional Water Heater is integrated into the engine coolant circuit: it is able to bring the engine up to operating temperature even before engine startup. Therefore, the Additional Water Heater ensures startup readiness even with low temperatures: fuel-intensive idling while stationary and during breaks are no become less invasive.

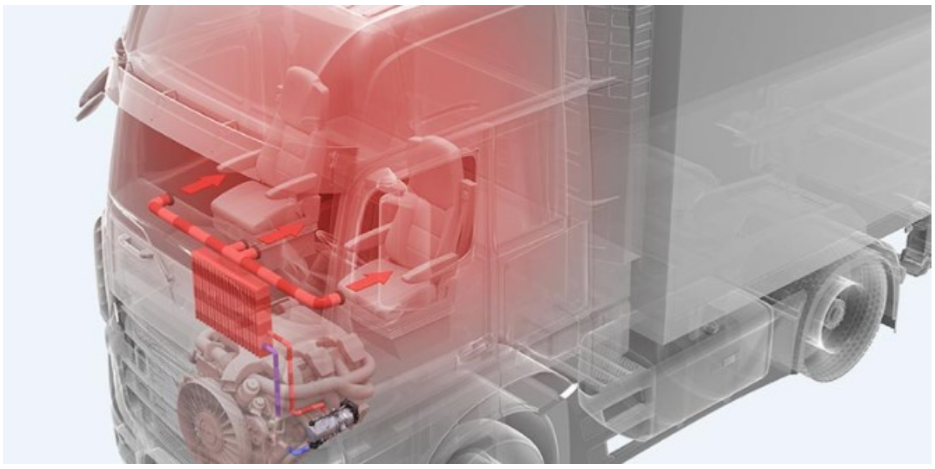


Figure 23 *Additional Water Heater and Engine integration*

The Additional Water Heater comprises the following component [23]:

1. *Coolant Pump*, directly assembled on the lower part of the burner. It is used to circulate the engine cooling water in the circuit. The power supply voltage is 24 V and the internal resistance is 20 +/- 1 kΩ;

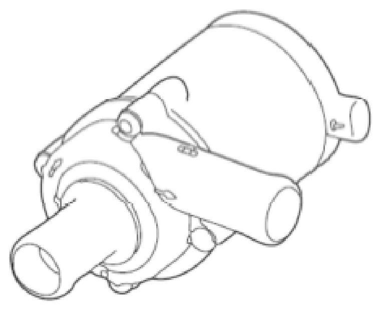


Figure 24 Coolant Pump

2. *Fuel metering pump*, it is installed on the chassis near the fuel tank with an inclination of 15° to facilitate the bleeding of air. Used to take and inject diesel into the burner system. The control unit supplies the pump with a pulse signal. For correct operation, the internal diameter of the delivery pipe must be 2 mm and must not exceed the length of 5 m. Incorporates a fuel filter and non-return check valve. The fuel capacity is: ≈ 0,374 +/- 10% L/min. The pump has a current draw of 0,64 A·h with a power supply of 24 V;

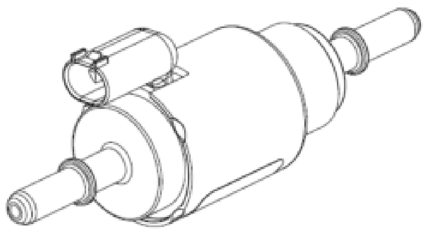


Figure 25 Fuel Metering Pump

3. *Injectors,* with which fuel is injected into the combustion chamber;
4. *Combustion chamber,* where the air-fuel is mixed, and its combustion takes place. The fuel is supplied to the vaporizer through the fuel pipe and is vaporized with the help of the spark plug;
5. *Glow plug*, a resistor located inside the combustion chamber (cold resistance at 22 +/- 5 °C equal to 0,780 +/- 0,110 Ω). The control unit supplies the glow plug with pulses of power by means of an internal electronic regulator;
6. *Fan,* with which the necessary air is supplied for combustion;
7. *Temperature* sensors:

- Exhaust gas: detects the flame and recognizes non-permitted exhaust gas temperatures, it has a resistance between 2050 – 2220 Ω;
- Refrigerant: detects coolant temperatures in the heat exchanger;
- Overheating: The temperature sensor detects the coolant temperature in the heat exchanger of the heater as the electrical resistance (value between 5047 and 2296 Ω). This signal is sent to the control unit where it is processed. The temperature sensor (W5) and the overheating sensor (W6) together with the duct and the connector make up one unit. The overheating sensor (resistance value between 250 and 30 Ω) protects the heater from excessive and not permitted operating temperatures. At temperatures exceeding 125 +/- 8 °C, it activates the safety shutdown of the heater. The temperature sensor is a semi-conducting element with a negative temperature coefficient (NTC). This means that the resistance of the component reduces as the temperature rises. The overheating sensor is a semi-conducting element with a positive temperature coefficient (NTC). This means that the resistance of the component increases as the temperature increases. Unlike the temperature sensor, the characteristic curve of the overheating sensor does not show a linear behaviour. AT approximately 125 °C, a sudden increase in the resistance occurs;

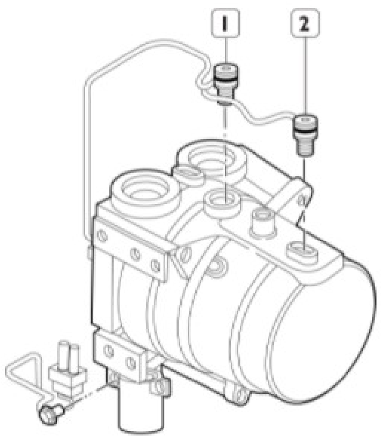


Figure 26 *(1) Coolant temperature sensor, (2) Overheating sensor*

8. *Heat* exchanger, with which the thermal energy is transferred, and hot air is conveyed into the cockpit through the existing ducting;
9. *3/2 way solenoid valve,* with which it is possible to activate the mode "Cabin heating" and /or "Engine preheating" by bypassing the refrigerant implemented remotely thanks to a button on the dashboard. In the "Cab Heating Mode" some of the engine coolant flows into the radiator (1) located in the cab inside the air conditioning and heating unit. The ventilation system lets the air pass through the radiator (1) thereby heating it: then, it can be re-introduced into the interior compartment.

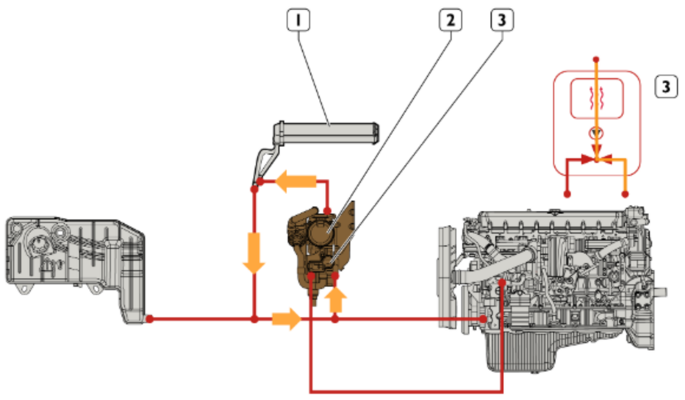


Figure 27 *Cab Heating Mode*

In the "Cab/Engine Heating Mode" (that can be activated by the dedicated button on the dashboard), the previous process is repeated with one difference: after flowing inside the radiator (1) the coolant is sent to the engine.

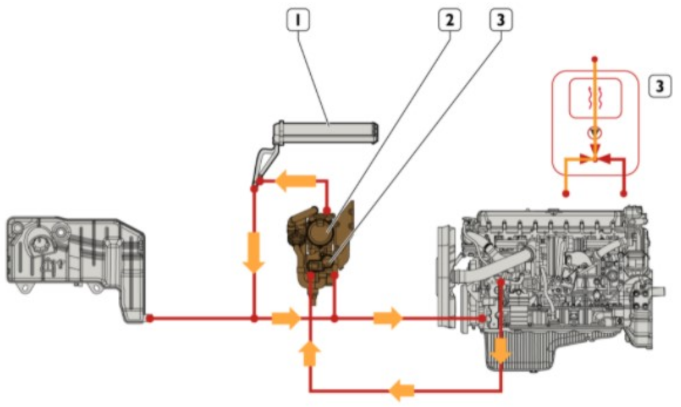


Figure 28 *Cab/Engine Heating Mode*

About the management of the thermal fluid temperature, the following graph can be considered [25].

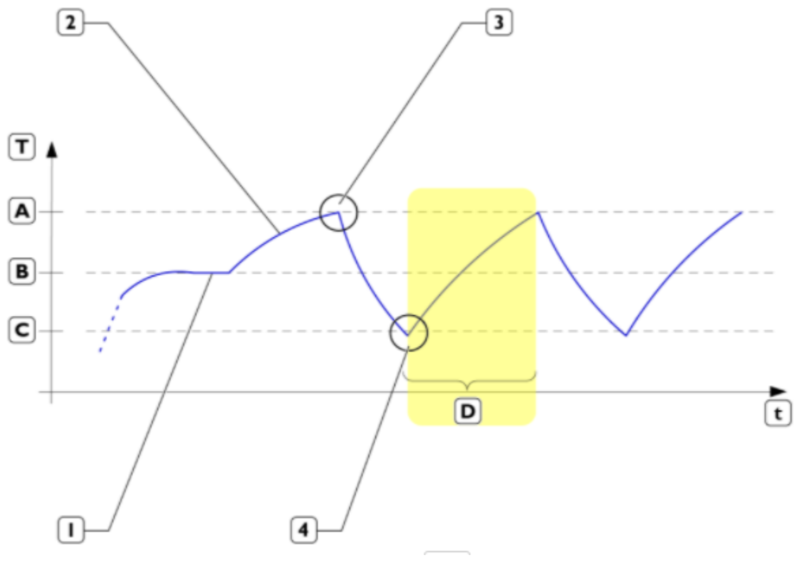


Figure 29 Fluid temperature management

Once the target temperature (B) has been reached (horizontal line (1)) and is stable, the heater is able to heat the thermal fluid with the minimum possible heat output (between 1 and 2 kW): in this condition, if the thermal fluid is not cooled, its temperature exceeds the target temperature (B) (ascending curve (2)). The temperature of the thermal fluid continues to rise until it reaches temperature (A) (point (3)). In this condition, the heater goes into standby mode, i.e. it stops burning the fuel and no longer heats the thermal fluid (standby mode is maintained even when heater activation is requested and the thermal fluid temperature is higher than the temperature (C)). As a result, the thermal fluid begins to cool and its temperature begins to fall (point (3)). The heater remains in standby mode until the thermal fluid temperature drops below temperature (C) (point (4)). When point (4) is reached, the heater reactivates the fuel burner and the thermal fluid temperature control switches to on/off ((ON/OFF) between temperature (A) and (C). The heater is programmed so that the time interval between a shutdown (point (3)) and an activation (point (4)) is a minimum of 15 min (interval (D)).

It is worth noting that the start-up phase of the Additional Water Heater is divided into two phases: flame ignition and flame stabilisation. The first has a variable duration, the second one has a fixed duration and starts immediately after the ignition phase.

It is also interesting analyse the Additional Water Heater failures management: generated errors are classified into error groups, and each group has a maximum limit for each counter. If the limit is reached or exceeded, a permanent lockout occurs: details are given in the following section.

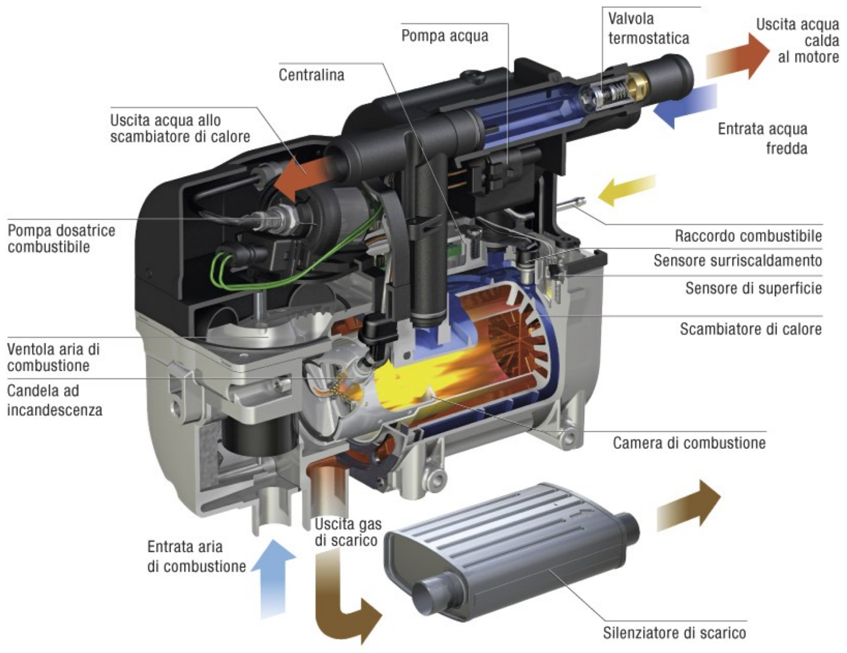


Figure 30 *Additional Water Heater Assembly*

3.2 Diagnostic Trouble Codes Management

Each error causes an Additional Water Heater temporary interlock only: if a group error counter reaches a specified limit, the Additional Water Heater will go into the "Interlock Mode". When this mode is activated, it is not possible to restart the Additional Water Heater: a reset by means of the Diagnostic Tool is needed. Some DTCs clusters can be identified, each one associated to a specific issue: each group includes a list of DTCs with all the parameters needed to characterize the problem. Among the most relevant clusters it is possible to identify the following ones [25]:

1. Coolant Pump;
2. Exhaust Gas Temperature;
3. Fuel metering Pump;
4. No Start / No Flame;
5. Glow Plug.

Each cluster has a counter, and for each group a threshold is defined. It's worth noting that the counters that are relevant for the Additional Water Heater Interlock are reset to zero when the heater reach a stable combustion and runs in combustion mode more than ten minutes or when an active Interlock is deleted by means of the Diagnostic Tool [25].

4 Machine Learning: Theory

4.1 Definition

“Machine Learning is the science of getting computers to learn and act like humans do, and improve their learning over time in autonomous fashion, by feeding them data and information in the form of observations and real-world interactions.” The starting point consists in data: algorithm work on data, and they are incredibly useful for difficult tasks when there is incomplete information of information that is too complex to be coded by hand.

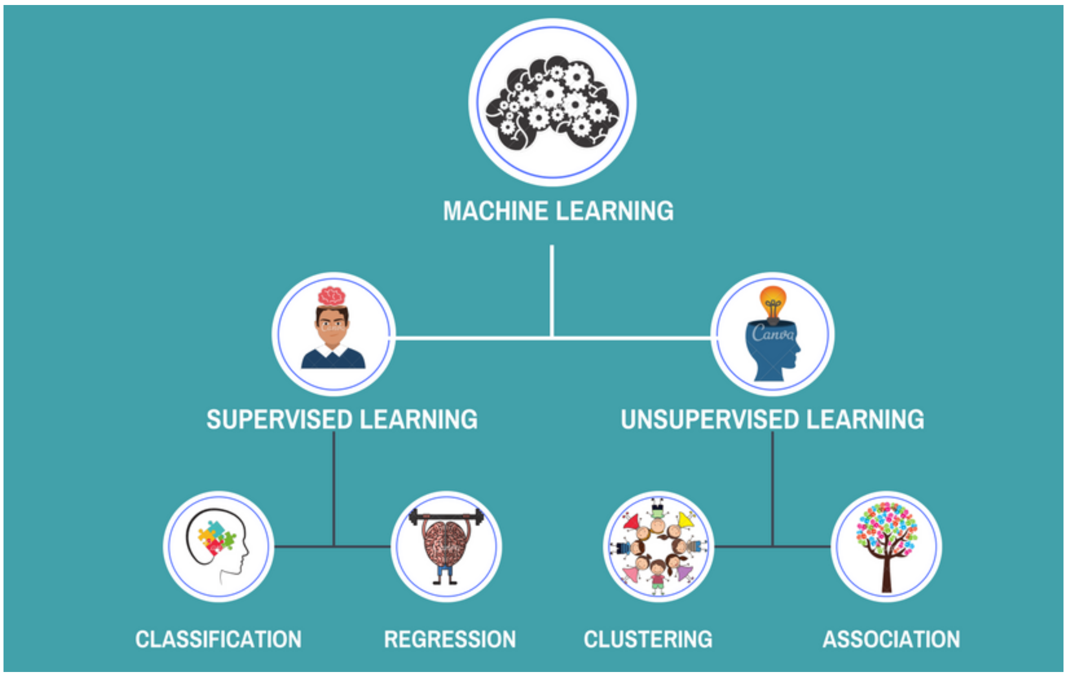


Figure 31 *Machine Learning algorithms classification*

4.2 Supervised and Unsupervised Machine Learning

Within machine learning, there are basically two approaches: supervised and unsupervised learning. While the first uses labelled data to help in predict outcomes, the second does not.

Supervised learning related algorithms are designed to learn by training data that consist of inputs paired with the correct outputs. During the training, patterns are searched within the data in order to correlate input (or features) and output: after the training, a supervised learning algorithm is able to predict the output of unseen inputs in according to the found patterns. Supervised techniques often require large datasets including great numbers of historic examples: this ensures a reliable learning process capable of handle also edge cases. On the other side, this approach can be very time-consuming and challenging from a computational point of view.

In contrast, unsupervised learning can handle large volumes of data in real time: unsupervised learning algorithms are capable of analyse and cluster unlabelled data sets discovering hidden patterns in data without the human intervention: the goal is to get insights from large volumes of new data without know what it is interesting and what not. The cons are a lack of transparency about how data is clustered and a higher risk of inaccurate results.

Therefore, the key difference between the two approaches can be referred to the use of labelled data: in supervised learning, the algorithm learns from the training dataset by iteratively making predictions and adjusting taking into account the correct answer; unsupervised learning models work in their own to discover underlying structure without use labelled data [6].

4.3 Classification and Regression

Within supervised machine learning field, an important distinction between classification and regression problems exists. The key difference between them is that the output variable in regression is continuous while that for classification is discrete.

However, it is also possible to predict a continuous value within the classification problem: in the present work, for example, the probability of each event belonging to a given output class is considered. Then, the predicted probability is turned into a class by selecting a "probability threshold": in other words, the probability is interpreted as the confidence of a given event belonging to that class.

4.4 Linear Classifier

A very important class of techniques for solving classification problems are based on linear models [7]. The goal is to divide the feature space into labelled regions with linear decision boundaries:

- Lines in the bidimensional space;
- Planes in the three-dimensional space;
- Hyperplanes with more features.

4.4.1 Logistic Regression

Consider n observations: for each one, the goal is to find the logistic regression function $p(x)$ such that the predicted responses $p(x_i)$ are as close as possible to the actual response *y_i* (that in a binary classification problem can be only 0 or 1) for each observation *i=1,...,n*. The consequence is that each $p(x_i)$ should be close to either 0 or 1, and that's why it is convenient to exploit the Sigmoid function, an S-shaped curve that have values very close to either 0 or 1 [8].

Once the logistic regression function *p(x)* is available, it is possible to use it to predict outputs for new and unseen inputs assuming that the mathematical dependence is unchanged.

Logistic regression is a classification technique that belongs to the group of the linear classifiers: it means that a linear function is involved (also called logit):

$$f(x) = b_0 + b_1\, x_1 + \cdots + b_r\, x_r$$

where the variables b_i are the estimators (also called predicted weights or coefficients) and they are determined by the logistic regression such that the function *p(x)* is as close as possible to the actual responses (model training phase). Once that the best weights are defined, the function *p(x)* is defined because it is the sigmoid function of *f(x)*:

$$p(x) = \frac{1}{1 + \ exp^{-f(x)}} > 0.5$$

such that it is possible to get the predicted output $p(x_i)$ of any given input x_i considering a threshold (generally equal to 0.5):

- if $p(x_i) > 0.5$, the output is 1;
- if $p(x_i) < 0.5$, the output is 0.

Considering a problem in which there is only one independent variable (single-variate logistic regression), the following figure can be considered in which it is easy represents the aforementioned concepts:

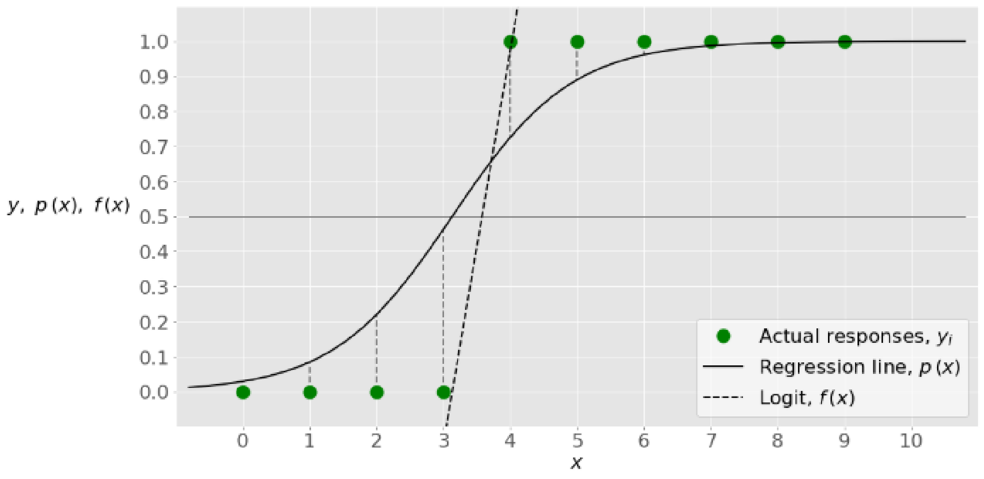


Figure 32 *Single-variate logistic regression*

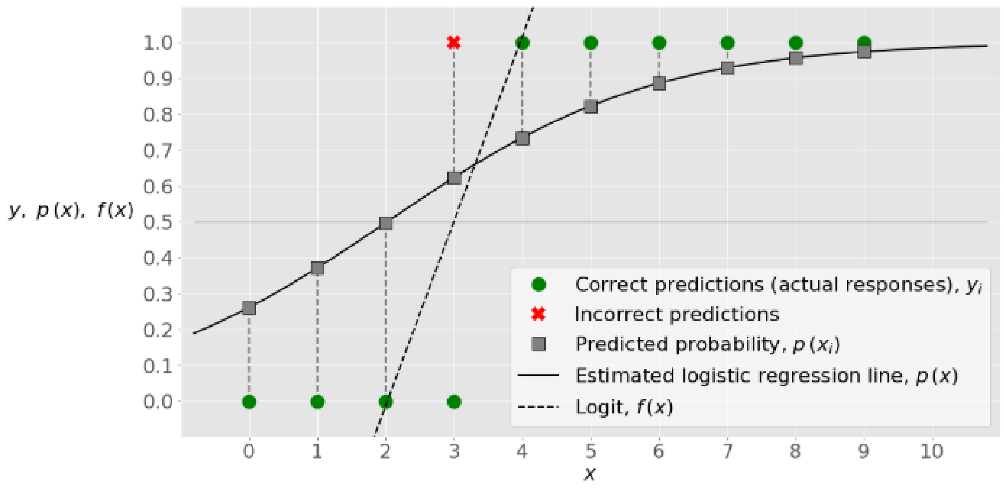


Figure 33 *Single-variate logistic regression*

A Logistic Regression model can be created in Python with the help of Scikit-Learn. When a model is created, several optional parameters can be set in order to define the behaviour of the model:

- ***penalty***: is a string that decides which approach to use for the regularization. It's worth noting that the regularization tries to reduce the complexity of the model: in other words, large coefficients b_i are penalized (it can be helpful with unseen data). Several norms can be used in the penalization:
  - *L1*, that considers the sum of the absolute values of the weights;
  - *L2*, that considers the sum of the squares of the weights;
  - *Elastic-net*, that is a linear combination of *L1* and *L2*.
- ***C***: it is a positive floating-point number. It is the defined as $1/\lambda$, where λ controls the trade-off between a complex model and a simple one. The consequence is that:
  - if λ is very low, the model is allowed to become more complex (overfitting) such that for each coefficient b_i an high value is assigned;
  - if λ is very high, the model will become too simple (underfitting).

  Therefore, the *C* parameter works in an inverse way with respect to the λ parameter: it defines the strength of regularization and small values indicate strong regularization.

- ***solver***: is a string and decides what solver to use for fitting the model, and therefore it is related to the algorithm to use in the optimization problem:
    - *Liblinear*: good choice with small datasets, but is not able to handle multiclass problems;
    - *Newton-cg*;
    - *lbfgs*;
    - *Sag*: fast with large datasets;
    - *Saga*: fast with large datasets and it is the only one to support the *elasticnet* penalty;
- ***max_iter***: is an integer that defines the maximum number of iterations done by the solver during model fitting

4.5 Nonlinear Classifier

Within nonlinear classifier, nonlinear functions can be used to separate instances that are not linearly separable

4.5.1 KNN

The K-NN algorithm considers input data as vectors in a multidimensional feature space: feature vectors are stored during the training phase assigning the reference class. An important parameter is the 'k value', a user-defined constant in according to which an unlabelled vector is assigned to a class considering the most frequent one among k training samples nearest the point.

Since the choice of k is up to the user, a critical point is to select the best k value: large values make boundaries smoother and less distinct (cons) t but reduce the noise in the classification (pros).

It's clear that when using K-NN algorithm a skewed distribution in the classes is a problem in a "majority voting" classification method: the most frequent class will be the most common among k nearest neighbours because they are in a larger number [9]. Some solutions can be adopted in order to overcome this problem:

- Balancing techniques (random oversampling and undersampling, SMOTE, etc.)
- Weight the classification: on Scikit-Learn several options are available
    - ‘uniform’, the default one that ensure an equal weight to all points;
    - ‘distance’, in according to which closer neighbours of a query point have a greater influence than further ones (the invers of the distance is used as a weight);
    - ‘custom’
- Self-organizing map (SOM)

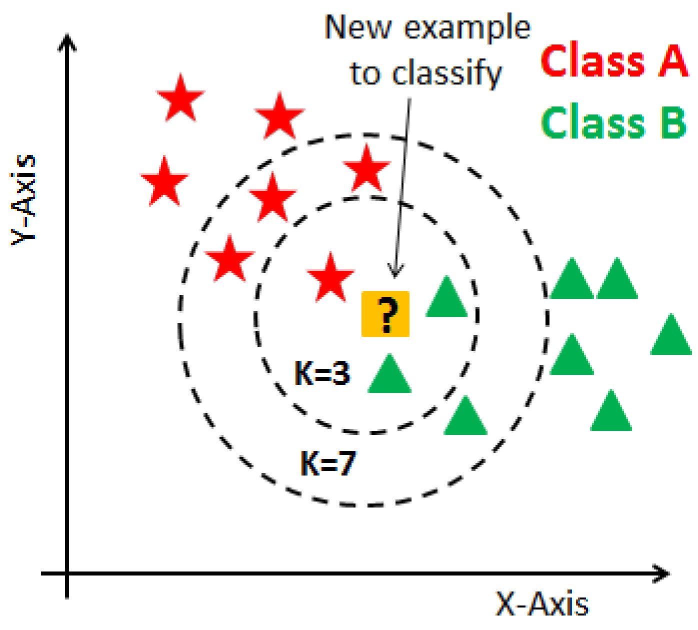


Figure 34 KNN logic

4.5.2 Decision Tree

In a decision tree there are internal nodes (test on a feature), leaf nodes (class label chosen after the considerations about all the features) and branches. A classification rule is the path from root to leaf [10].

The tree is built iteratively thanks to the training dataset: the goal should be a split of the training set into areas where only the events belonging to one label are present. A decision boundary is set by testing all the possible decision boundaries splitting the dataset and choosing the one that ensures Gini impurity minimization, i.e. the probability from a randomly chosen element to be misclassified. This strategy is applied for all the splits, and so also for also deciding the root split. If it is not possible to have a more homogeneous group (gini = 1), the algorithm will not try to split this part, otherwise it will continue to split [10].

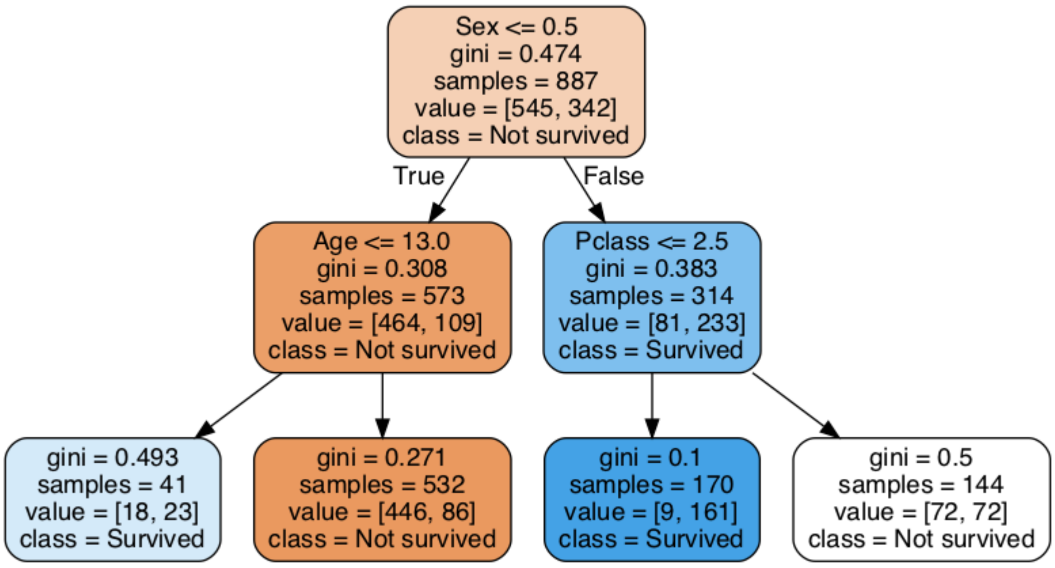


Figure 35 Deicison Tree representation

4.5.3 Random Forest

A Random Forest is a large number of individual decision trees that operates as an ensemble: “A large number of relatively uncorrelated models (trees) operating as a committee will outperform any of the individual constituent models” [11]. The key concept is the low correlation between models: each tree protects each other from their individual errors. This result is achieved by means of bagging and feature randomness. These techniques are based on the fact that decision trees are very sensitive to small changes in the training dataset: with a small perturbation is possible to get very different trees structures, and so the bagging is exploited to build low correlated trees: each tree can randomly sample from the dataset with replacement, resulting in different trees. In order to force even more variation among trees, the features randomness come to play: each tree in a random forest can pick only from a random subset of features. To summarize, in the final random forest there are trees that are trained on different sets of data and that use different features to make decisions, hence low correlated trees.

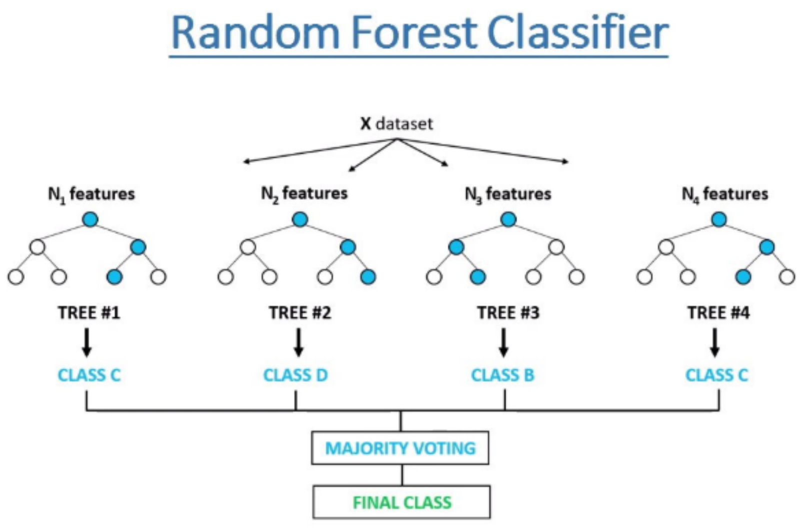


Figure 36 *Random Forest representation*

About Random Forest Hyperparameters, the following ones are considered for tuning the model performance [11]:

- ***n_estimators***: number of trees; it is increased until the model performance stabilizes; intuition might suggest that more trees will lead to overfitting, but bagging and features randomness avoid it;
- ***max_depth***: the maximum depth of decision trees used in the ensemble;
- ***min_samples_split;***
- ***min_samples_leaf;***
- ***max_features***: number of features randomly selected at each split point.

4.5.4 Extreme Gradient Boosting

The idea of Boosting derives from the idea of whether a weak learner can be modified to become better: a weak learner performance is at least slightly better than random chance. The basic concept is to use weak learners several times in order to get a succession of hypotheses, each one focused on the events that the previous one misclassified. By simplifying, boosting refers to a general problem of producing very accurate prediction rule by combining rough and moderately inaccurate rules-of-thumb, and so it means a conversation of weak learners into strong one [13].

In general, gradient boosting trains several models in a gradual, additive, and sequential manner. Therefore, it is possible to identify three elements involved in gradient boosting:

1. Loss function to be optimized: it is a measure indicating how good model's coefficients are at fitting the underlying data; then, models are fitted using loss function and gradient descendent optimization algorithm such that the minimization is performed as the model is fit;
2. Weak learner: decision tree is constructed in a greedy way, choosing the best split point taking into account purity scores;

3. Additive Model: trees are added one at time without modifying the existing trees. This procedure allows to correct the prediction errors made by prior models

Extreme Gradient Boosting (or XGBoost), in particular, is an efficient open-source Python library that implement the gradient boosting algorithm.

About XGBoost Hyperparameters, the following one are considered for tuning the model performance [13]:

- ***n_estimators***: number of decision trees used in the ensemble; considering what previously mentioned, trees are added one by one to the model in order to correct and improve upon the predictions made by prior ones: more trees is often better (but not always);
- ***learning_rate***: it controls the amount of contribution that each model has to ensemble prediction. Therefore, smaller rates require more decision trees in the ensemble and vice versa;
- ***gamma;***
- ***max_depth***: this parameter controls how specialized each three is affecting generalization or overfitting capabilities. Gradient boosting work well with trees that have a modest depth;
- ***subsample***: each tree is fit on a randomly selected subset of the training dataset, with the number of samples that can be customized. The reasoning behind is that using fewer samples allows variance introduction for each tree, but can overall improve the overall performance of the model;
- ***colsample_bytree***: the number of features used to fit each decision tree.

4.6 Model Evaluation Tools

The evaluation of the machine learning algorithm is an essential part of the project, and it is fundamental to choose the proper evaluation tool in order to judge accurately the model. As the matter of fact, how the model is able to generalize on unseen data is a very important aspect: there is the need to know whether the model actually works and if we can trust its predictions on unseen data.

A powerful tool exploited to describe the performance of a classification model is the confusion matrix:

n=165	**Predicted: NO**	**Predicted: YES**	
Actual: NO	TN = 50	FP = 10	60
Actual: YES	FN = 5	TP = 100	105
	55	110	

Figure 37 *Classification matrix*

As shows in the picture, four important elements are included in the confusion matrix:

1. ***True Positives***: cases in which the prediction is "Yes" and the actual output was also "Yes";
2. ***True Negatives***: cases in which the prediction is "No" and the actual output was also "No";
3. ***False Positives***: cases in which the prediction is "Yes" and the actual output was "No";
4. ***False Negatives***: cases in which the prediction is "No" and the actual output was "Yes";

In order to evaluate the model, the classification report can be also exploited: it shows a representation of the main classification metrics on a class basis giving an over global accuracy of the model in terms of true/false positives and true/false negatives.

The main classification metrics can be represented by means of the classification report. The following scores are taken into account:

- ***Precision***: ratio of true positives to the sum of true and false positives ("For all instances classified positive, what percent was correct?");

$$Precision = \frac{True\ Positives}{True\ Positives + False\ Positives}$$

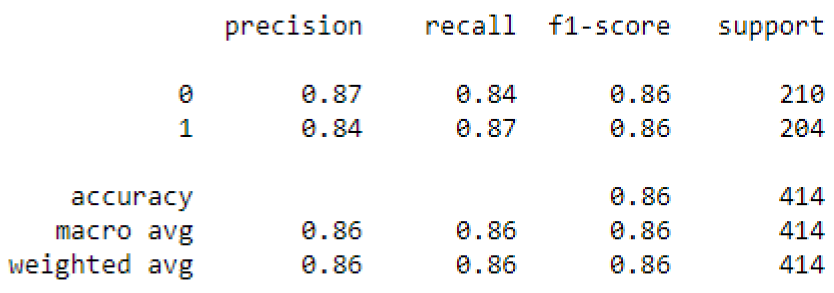
- ***Recall***: ratio of true positives to the sum of true positives and false negatives ("For all instances that were actually positive, what percent was classified correctly?");

$$Recall = \frac{True\ Positives}{True\ Positives + False\ Negatives}$$

- ***F1***: weighted harmonic mean of precision and recall such that the best score is 1 and the worst one 0 such that the weighted average of the F1 score is a good tool to compare classifier models;

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

- ***Support***: the number of actual occurrences of the class in the specified dataset



```
              precision    recall  f1-score   support

           0       0.87      0.84      0.86       210
           1       0.84      0.87      0.86       204

    accuracy                           0.86       414
   macro avg       0.86      0.86      0.86       414
weighted avg       0.86      0.86      0.86       414
```

Figure 38 *Classification Report example*

4.7 Framework

The present work was done exploiting the Jupyter Notebook, a free, open-source interactive web tool that allows to combine software code, explanatory test, and multimedia resources in a single document (notebook documents) by means of a web-based interface.

Figure 39 Jupyter Notebook logo

A Notebook document is a very powerful tool: it is a document that contains both code and rich text elements (figures, links, equations, …) and so it is suitable for Data Science works. As a matter of fact, it is possible to bring in a single place analysis description and analysis results, with the possibility to execute the code and perform a real time data analysis.

For Data Scientist, Jupyter has emerged as a de facto standard: it is possible to execute the contained in a cell, see what happens, modify and repeat in an iterative process allowing a powerful connection between user, theory, data and results. In fact, the structure of a Jupyter Notebook is arranged in modules (the cells) that can be executed out of order and not only in a sequenced way.

A very important concept is the kernel: it is a "computational engine" that execute the code contained in a Notebook document. In the present dissertation, the python kernel is taken into account because python programming language is used (other kernels exist, and they can be chosen from the Notebook Dashboard that can be considered a Jupyter Management System).

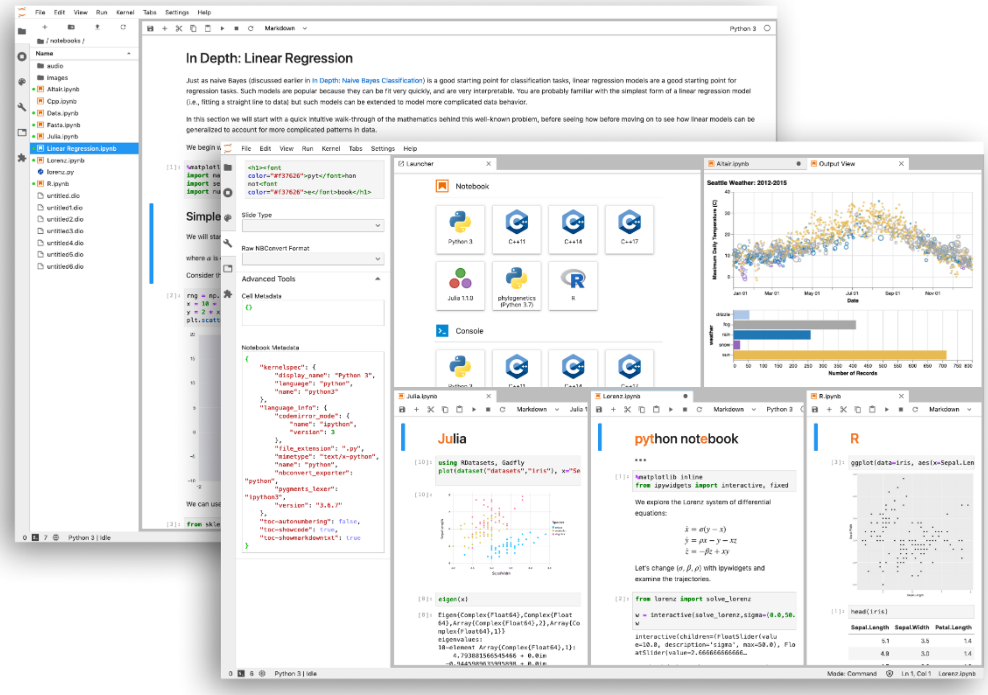


Figure 40 *Jupyter Notebook interface*

In the following, the reasons in according to which Python has been chosen among the available programming languages.

First, Python is a general-purposes language that is simple and consistent: developers can focus their effort into solving a Machine Learning problem instead of focusing on the technical nuances of the language. Additionally, Python is easy to learn because it is understandable by humans: it is very intuitive with respect to other programming languages. This pro is stressed by the availability of frameworks, libraries and extensions dedicated to the Machine Learning field that simplify the simplify the code: it is fundamental have a well-structured and well-tested environment in order to write reliable systems. Among the most relevant, the following ones are selected:

- Scikit-learn: a robust library that allows to bring machine learning into a production system. Scikit-learn provides a wide range of supervised and unsupervised learning algorithms: the focus is on modelling data, and so cross validation tools (scope: estimation of performance of supervised models on unseen data), ensemble methods (combination of predictions of multiple supervised models), features selections tools (identification of meaningful attributes in order to select a subsample of useful features upon which the model is built) and parameter tuning tools are available;

Figure 41 *Scikit-learn logo*

- Pandas: a data analysis and manipulation tool built on top of Python. It allows a fast and efficient DataFrame object creation that can be intelligently manipulated (data alignment, missing data, size mutability, merging and joining of data sets);

Figure 42 *Pandas logo*

- Numpy: it a Python library that provides a multidimensional array object and a range of routines for fast operations on them (shape manipulation, sorting, selecting, …) ;

Figure 43 *NumPy logo*

- Matplotlib: it is a library for creating static, animated and interactive visualizations in Python; it allows the creations of plots and their complete customization;

Figure 44 *Matplotlib logo*

- Seaborn: it is a Python data visualization library based on matplotlib focused on statistical graphics. It is a powerful tool when exploring and understanding data

Figure 45 *Seaborn logo*

Finally, it is worth noting that Python has a great community and popularity: it is among the top 5 most popular programming languages. Therefore, an active exchange of experience is easy and the probability to fix a code-related problem is very high: for any task, there could be advice and guidance from experienced developers.

5 Machine Learning: Additional Water Heater Case Study

As stated before, the Additional Water Heater is a very important component from the comfort point of view and that can also help in achieve better performance: the goal of the present work is to foresee a possible malfunction of the Additional Water Heater 12 hours in advance taking into account all the data arriving from the telematics. Once the Exploratory Data Analysis is carried out, the next step was to exploit the identified patterns in the data to build a model able to predict the Interlock with twelve hour in advance: the breakdown of the vehicle is avoided and a technical inspection can be organized with the workshop.

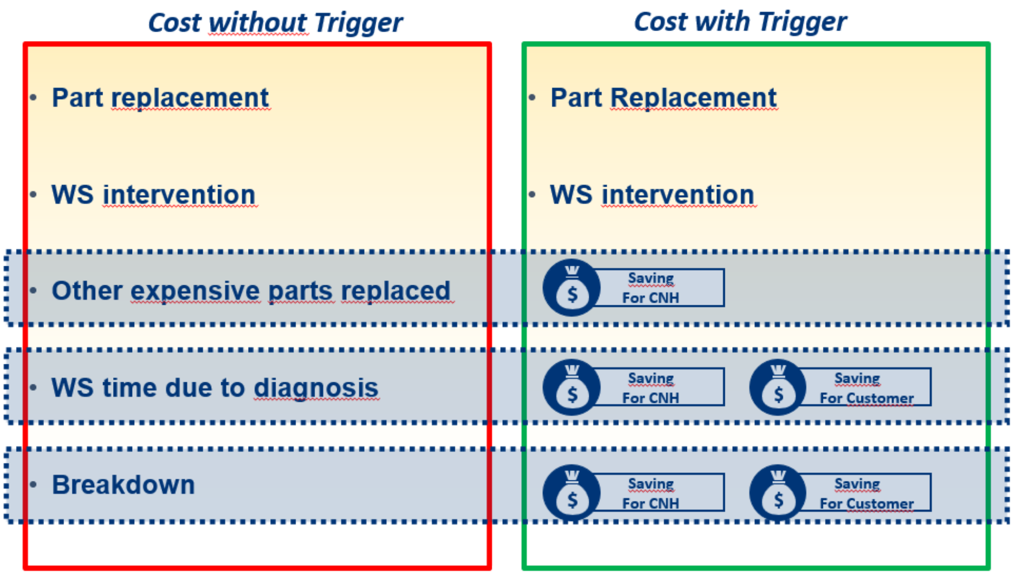
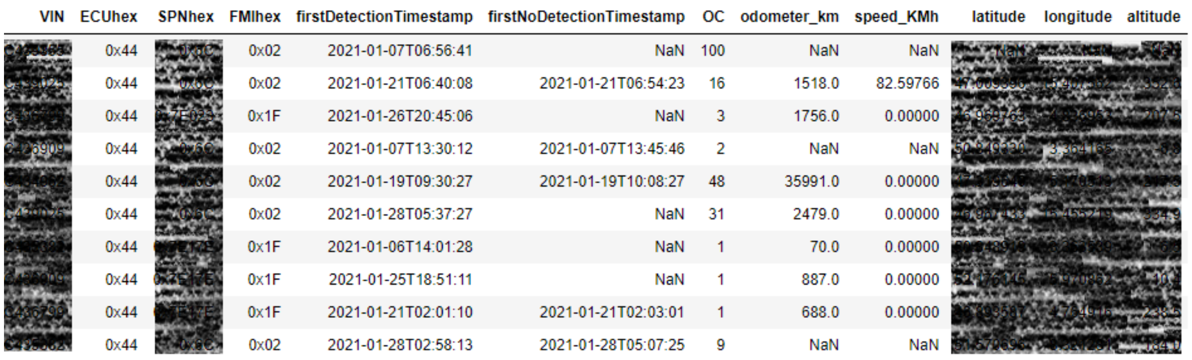


Figure 46 Savings due to the trigger development

5.1 Gathering Data Phase

In order to achieve the target of this work, several sources of data are considered. First of all, diagnostic trouble code's raw data extractions are made. It is worth noting that the considered data cover the range October 2020 – March 2021: all the diagnostic trouble codes sent by the Additional Heater during these months are stored along with several important and precious information. Among the most relevant:

- "First Detection Date" and "Last Detection Date";
- Duration;
- Telematics and ECU occurrences.



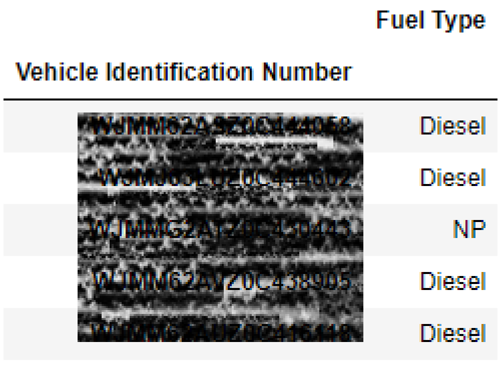
VIN	ECUhex	SPNhex	FMIhex	firstDetectionTimestamp	firstNoDetectionTimestamp	OC	odometer_km	speed_KMh	latitude	longitude	altitude
[redacted]	0x44	[redacted]	0x02	2021-01-07T06:56:41	NaN	100	NaN	NaN	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x02	2021-01-21T06:40:08	2021-01-21T06:54:23	16	1518.0	82.59766	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x1F	2021-01-26T20:45:06	NaN	3	1756.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x02	2021-01-07T13:30:12	2021-01-07T13:45:46	2	NaN	NaN	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x02	2021-01-19T09:30:27	2021-01-19T10:08:27	48	35991.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x02	2021-01-28T05:37:27	NaN	31	2479.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x1F	2021-01-06T14:01:28	NaN	1	70.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x1F	2021-01-25T18:51:11	NaN	1	887.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x1F	2021-01-21T02:01:10	2021-01-21T02:03:01	1	688.0	0.00000	[redacted]	[redacted]	[redacted]
[redacted]	0x44	[redacted]	0x02	2021-01-28T02:58:13	2021-01-28T05:07:25	9	NaN	NaN	[redacted]	[redacted]	[redacted]

Figure 47 *Additional Water Heater DTCs DataFrame*

Moreover, some vehicle's characteristics are taken into account in order to better identify the event:

- Vehicle Identification Number;
- Fuel Type;
- Odometer.

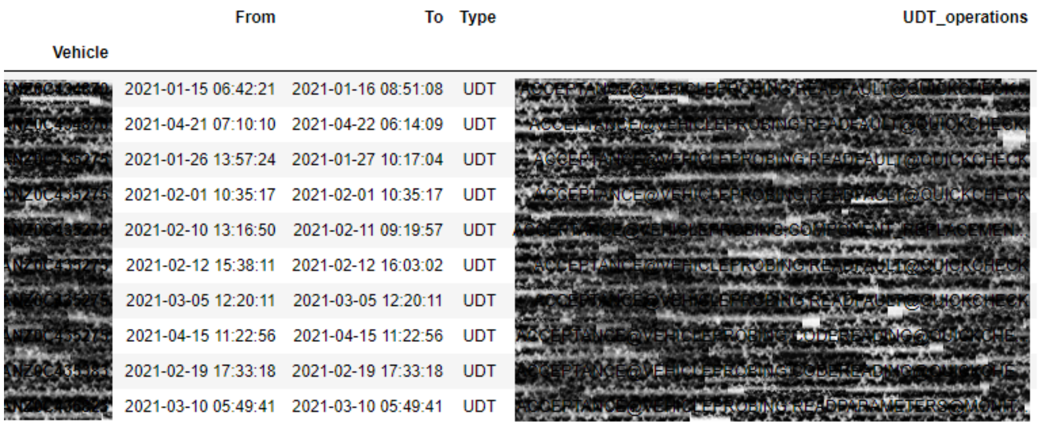
All the data are cross-referenced with another dataset in which the fuel type specifications are specified



Vehicle Identification Number	Fuel Type
[redacted]	Diesel
[redacted]	Diesel
[redacted]	NP
[redacted]	Diesel
[redacted]	Diesel

Figure 48 Fuel type DataFrame

All the data are cross-referenced with the DataFrame related to geofencing and teleservices: these data refer to the vehicles stops in the workshops. All the session's information is stored: operations done with the diagnostic tools such as date and odometer at the moment of the operations are known.



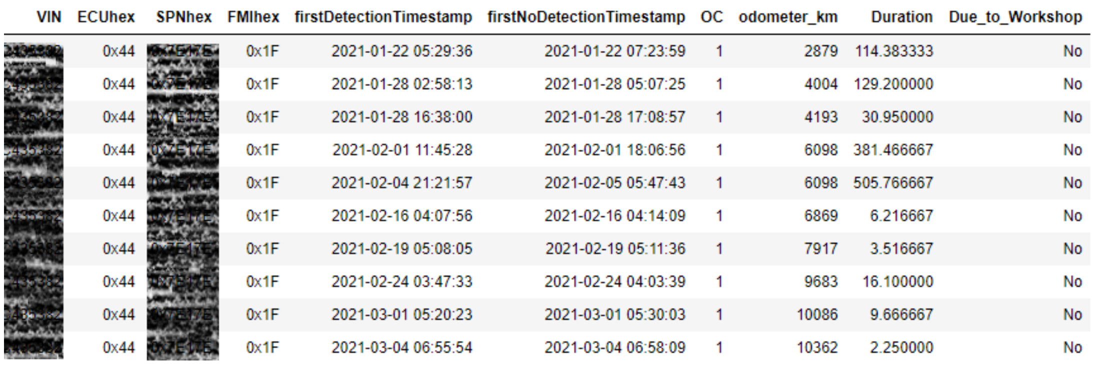
Vehicle	From	To	Type	UDT_operations
[redacted]	2021-01-15 06:42:21	2021-01-16 08:51:08	UDT	[redacted]
[redacted]	2021-04-21 07:10:10	2021-04-22 06:14:09	UDT	[redacted]
[redacted]	2021-01-26 13:57:24	2021-01-27 10:17:04	UDT	[redacted]
[redacted]	2021-02-01 10:35:17	2021-02-01 10:35:17	UDT	[redacted]
[redacted]	2021-02-10 13:16:50	2021-02-11 09:19:57	UDT	[redacted]
[redacted]	2021-02-12 15:38:11	2021-02-12 16:03:02	UDT	[redacted]
[redacted]	2021-03-05 12:20:11	2021-03-05 12:20:11	UDT	[redacted]
[redacted]	2021-04-15 11:22:56	2021-04-15 11:22:56	UDT	[redacted]
[redacted]	2021-02-19 17:33:18	2021-02-19 17:33:18	UDT	[redacted]
[redacted]	2021-03-10 05:49:41	2021-03-10 05:49:41	UDT	[redacted]

Figure 49 Teleservices and Geofencing DataFrame

These parameters are important because all the diagnostic trouble codes there are generated during the stops must not be considered.

5.2 Cleaning Data Phase

"Data Cleaning" refers to identifying and correcting errors in the dataset that may impact a predictive model. Before choosing the proper algorithm, the proper data must be chosen. Data Cleaning is one of the most important steps and the most time consuming, but create a reliable dataset is critical because the performance of the results depends on them. One of the first problems is "fill-out missing values". In this case study, there were missing values related to the odometer: the fastest approach is to delete data with no odometer, but this means a loss of information. It follows that if the target is to achieve high performance, all the details must be considered: a customized filling strategy is adopted and an approximation of the odometer based on the last and following odometer information are taken into account (media).



VIN	ECUhex	SPNhex	FMIhex	firstDetectionTimestamp	firstNoDetectionTimestamp	OC	odometer_km	Duration	Due_to_Workshop
[redacted]	0x44	[redacted]	0x1F	2021-01-22 05:29:36	2021-01-22 07:23:59	1	2879	114.383333	No
[redacted]	0x44	[redacted]	0x1F	2021-01-28 02:58:13	2021-01-28 05:07:25	1	4004	129.200000	No
[redacted]	0x44	[redacted]	0x1F	2021-01-28 16:38:00	2021-01-28 17:08:57	1	4193	30.950000	No
[redacted]	0x44	[redacted]	0x1F	2021-02-01 11:45:28	2021-02-01 18:06:56	1	6098	381.466667	No
[redacted]	0x44	[redacted]	0x1F	2021-02-04 21:21:57	2021-02-05 05:47:43	1	6098	505.766667	No
[redacted]	0x44	[redacted]	0x1F	2021-02-16 04:07:56	2021-02-16 04:14:09	1	6869	6.216667	No
[redacted]	0x44	[redacted]	0x1F	2021-02-19 05:08:05	2021-02-19 05:11:36	1	7917	3.516667	No
[redacted]	0x44	[redacted]	0x1F	2021-02-24 03:47:33	2021-02-24 04:03:39	1	9683	16.100000	No
[redacted]	0x44	[redacted]	0x1F	2021-03-01 05:20:23	2021-03-01 05:30:03	1	10086	9.666667	No
[redacted]	0x44	[redacted]	0x1F	2021-03-04 06:55:54	2021-03-04 06:58:09	1	10362	2.250000	No

Figure 50 *Cleaned Additional Water Heater DTCs DataFrame*

Another important step is to identify columns that are useless in building the model because of they are near-zero variance predictors: these columns are removed from the dataset because are not able to add information and/or characterize in a proper way the event. It's worth noting that low variance columns are here considered: a variance threshold can be considered.

Moreover, cleaning data includes dealing with duplicated data: considered the ECU's working logic and considered the row data are directly used to build the model, the starting DataFrame will be (more or less) double with respect to the final one. As the matter of fact, each line is associated to a particular "Detection" or "No Detection" of a DTC, hence there will be a row with the detection and another related to the detection. Some problems arise: not all the DTC have the "No

Detection", or some DTC will have the "No Detection" only after the stop in the workshop.

Finally, the final DataFrame is get. Each row is associated to a specific vehicle identification number: all the diagnostic trouble codes within 15 days are considered with the relative parameters. In order to be consistent with the objective of the forecast (i.e. twelve hours before the interlock), if a particular event (i.e. a row) is associated with an Additional Water Heater Interlock, the specific time coordinates of the lock are registered and all the DTCs sent within twelve hours before are neglected.

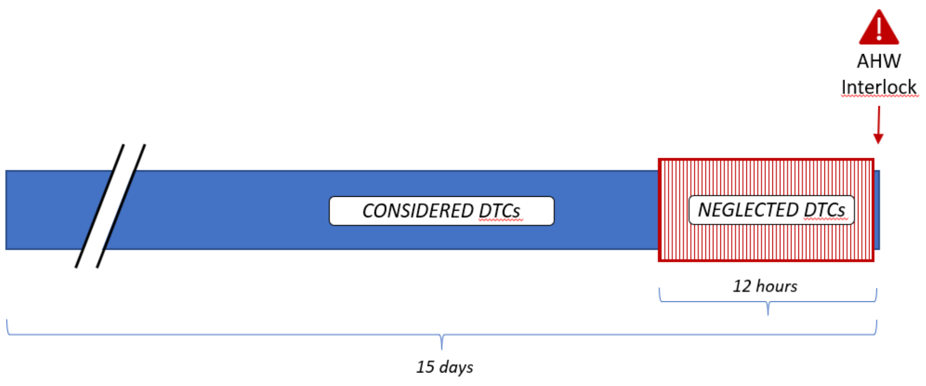
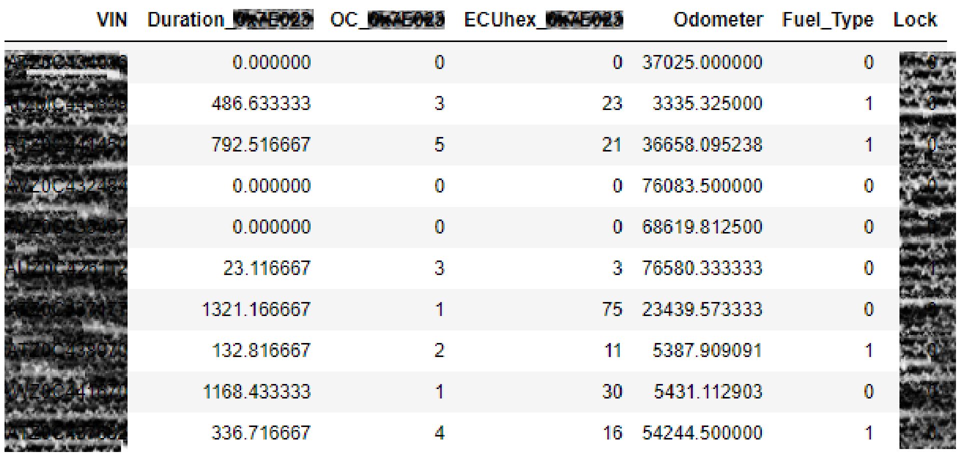


Figure 51 Considered and Neglected DTCs in the Training DataFrame

This operation is crucial: the training of the algorithm is done by linking all the DTCs sent by the ECU with the Additional Water Heater status (i.e. Lock or No Lock) neglecting data transmitted in the previous 12 hours. Therefore, the algorithm that is built upon the considered dataset will have as input *all* the available telematics data at a specific time and it will be able (with the support of the vehicle configuration details discussed in *Chapter 5.2*) to foresee the Additional Water Heater status twelve hours later. This dataset is the *training dataset* exploited for the learning of the model.



VIN	Duration_[redacted]	OC_[redacted]	ECUhex_[redacted]	Odometer	Fuel_Type	Lock
[redacted]	0.000000	0	0	37025.000000	0	[redacted]
[redacted]	486.633333	3	23	3335.325000	1	[redacted]
[redacted]	792.516667	5	21	36658.095238	1	[redacted]
[redacted]	0.000000	0	0	76083.500000	0	[redacted]
[redacted]	0.000000	0	0	68619.812500	0	[redacted]
[redacted]	23.116667	3	3	76580.333333	0	[redacted]
[redacted]	1321.166667	1	75	23439.573333	0	[redacted]
[redacted]	132.816667	2	11	5387.909091	1	[redacted]
[redacted]	1168.433333	1	30	5431.112903	0	[redacted]
[redacted]	336.716667	4	16	54244.500000	1	[redacted]

***Figure 52** Machine Learning training DataFrame*

This dataset is the output of the cleaning phase and is the most time-consuming activity done. However, Data Cleaning can be considered the secret ingredient of any Data Science Project: from a performance point of view, better input data beats fancier algorithms.

5.2.1 Features Engineering

In order to improve the performance of machine learning models, it is necessary to perform not only a cleaning phase of the instances, but also a features engineering phase must be present. Basically, as previously mentioned input data are needed, and they are arranged in a columns structure: the algorithm output is strongly affected by the used features. Moreover, better features mean simpler models: it is possible to choose less than optimal models and still get good results because with good features there is an underlying better representation of the real problem. In addition, this representation is characterized by a reduced complexity with respect to a model that include all the attributes, that is a simpler model to understand and explain: in a corporate environment, this aspect is fundamental [14].

In particular, it is possible to sentence that feature engineering is a representation problem: it is a phase in which there is a manual work aimed at achieving the training data from which the model can learn a solution and a pattern.

In details, it is possible to objectively estimate the usefulness of features: for example, each feature can be associated to a score, and the ones with highest score can be selected in the training dataset while ignoring the others. In other words, from many features only a subset of few features is selected: this process is called "Feature Selection".

In practice, several approaches can be adopted for feature selection [14]:

1. Univariate Selection
   It is a method that exploits statistical tests to select features with the strongest relationship with the output variable. Within our case study (numerical input, categorical output) the ANOVA F-value must be used: it is possible to implement it by means of the SelectKBest library provided by scikit-learn library;

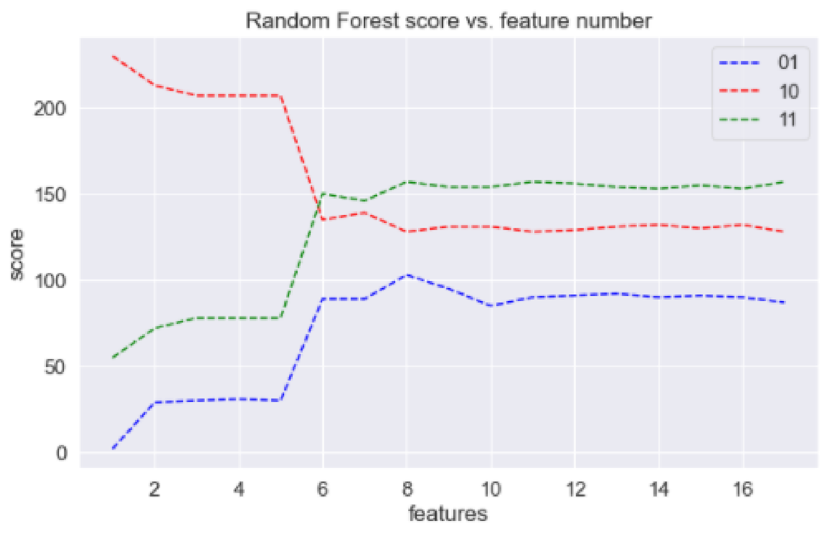


Figure 53 Univariate Feature Selection

2. Recursive Feature Elimination (RFE)

   This method works by recursively eliminate features and building a model on those features that remain exploiting the model performance as a metric.

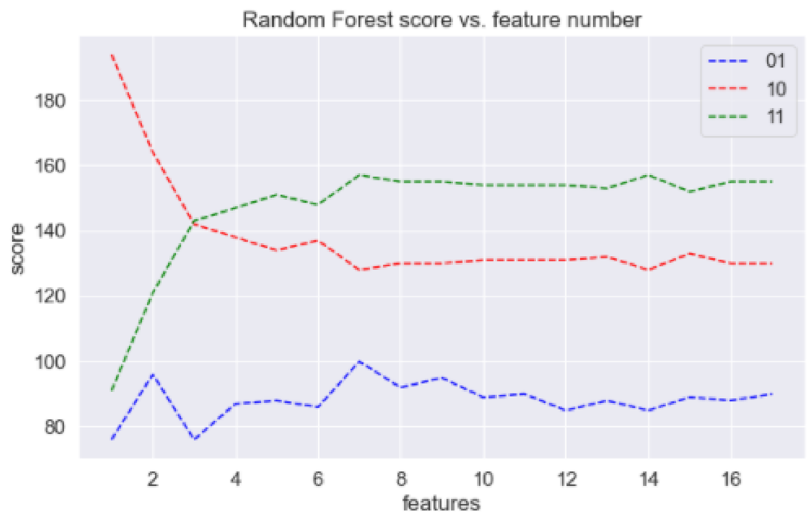


Figure 54 Recursive Feature Elimination

3. XGBoost built-in function

   Deploying ensembles of decision tree, while trees are built is possible to obtain an automatic estimation of the features importance. Each feature is associated to a score that indicated how useful it is in the construction of the boosted decision trees within the model: the more an attribute is used to make decisions trees, the higher its relative importance. At the end, a rank can be computed. Therefore, feature importance scores can be used for feature selection.

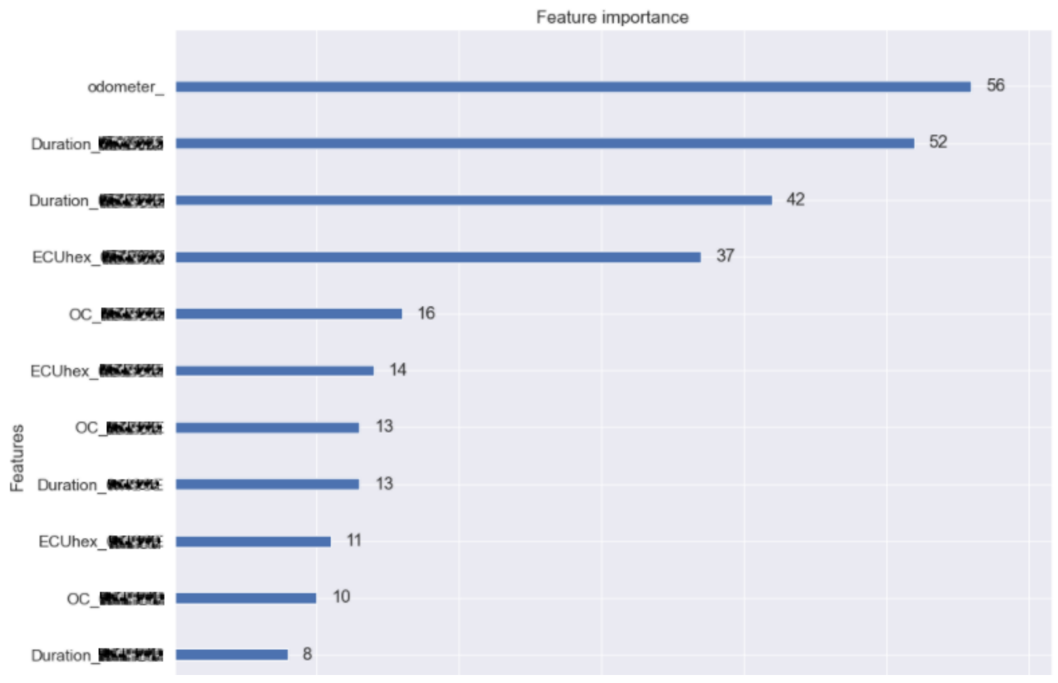


Figure 55 XGBoost built-in feature selection

5.2.2 Outliers Detection

"In statistics, an Outlier is an observation point that is distant from other observations": an outlier is exceptionally far from the mainstream of the data, and can be the consequence of a measurement error, data corruption or a rare event. Due to the definition, it is possible to adopt statistics methods to identify events that are rare with respect to the available dataset [16]:

1. ***Standard Deviation Method***

   Assumption: sample data must follow a Gaussian-like distribution. In this case, the standard deviation from the mean can be used to identify the percentage of values in the dataset, and so it is possible to impose a cut-off based on the standard deviation. A common approach consists in considering outlier all the data that falls outside 3 standard deviations;

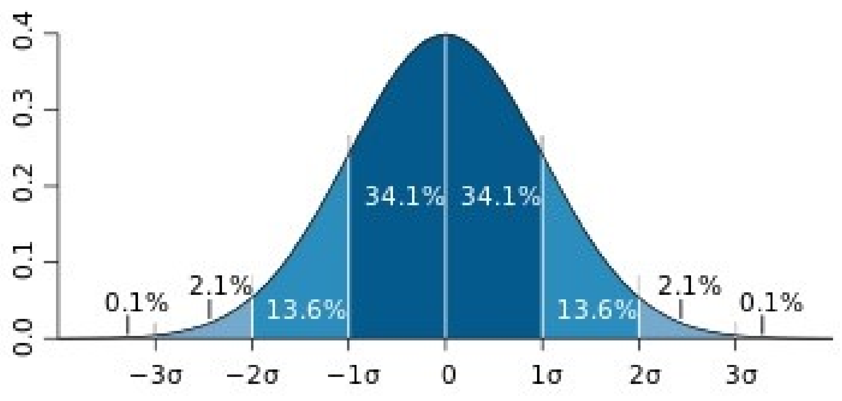


Figure 56 *Standard Deviation Method representation*

2. ***Interquartile Range Method***

   IQR is a good tool for studying a non-Gaussian distribution sample of data: it defines the middle 50% of the ordered dataset, and a region in which a normal data point has a high probability to belong to. On the other side, outliers should occur in the low probability region: these regions can be defined imposing a factor k of the IQR.

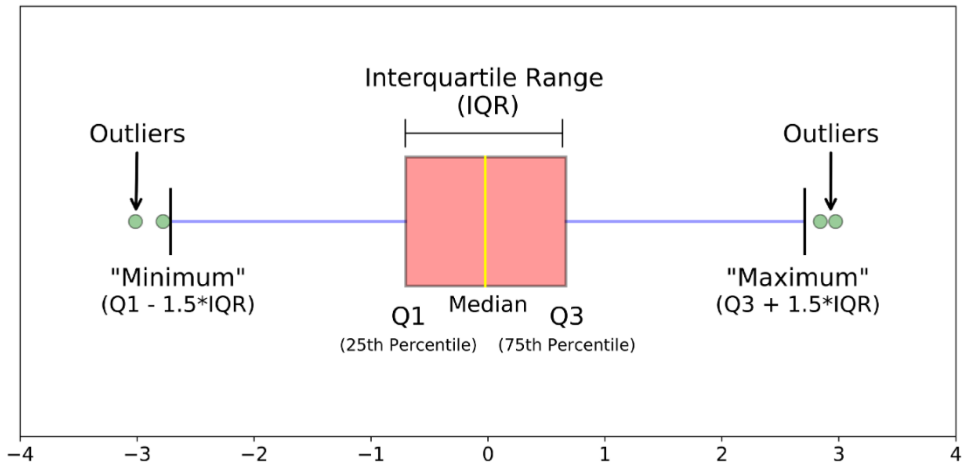


Figure 57 *Interquartile Range Method representation*

Within the present case study, these two statistic approaches cannot be uses. First of all, the outliers in the training dataset are events in which the Additional Water Heater is interested by DTCs with an abnormal pattern related to occurrences and duration that have caused the lock or not. It's worth noting that two different classes must be considered (Additional Water Heater locked and not locked): due to the heavy unbalance of the dataset towards the "Not Lock Events", in the starting DataFrame the largest part of the events is characterized by DTCs occurrences and duration equal to zero. This situation is stressed by the fact that around 15 different DTCs are considered, and even when the additional water heater lock occurs, only a few of them is characterized by values different from zero: it depends on the particular problem that the component is facing. Thus, the Interquartile Range Method would cause the elimination from the dataset of the largest part of events interested by DTCs parameters different from zero: there would be an enormous loss of information, and the real problem would not properly modelled. About the standard deviation method, as just stated the data are not Gaussian-distributed, but it is a skewed distribution toward the left [17].

However, outliers detection is necessary because machine learning algorithms are very sensitive to the range and distribution of attribute values: outliers presence can result in poor fit and lower predictive modelling performance.

An alternative approach to the statistics methods are the "Automatic Outlier Detection" ones. Among the different alternatives, the Isolation Forest was chosen.

The Isolation Forest is a tree-based anomaly detection algorithm: it takes advantage of two anomalies' quantitative properties:

- Minority with respect to the normal instances;
- Very different attribute-values with respect to the normal instances.

In particular, the Isolation Forest "returns the anomaly score of each sample, isolating observations by randomly selecting a feature and then randomly selecting a split value between the maximum and minimum values of the selected feature". Some important parameters are:

- ***n_estimators***: number of trees that will be built;
- ***max_samples***: maximum number of samples to be selected to train each tree;
- ***contamination***: expected proportion of outliers in the dataset;
- ***max_features***: the maximum number of features do draw from the total features in order to train each tree.

After having defined the model and its parameters, it needs to be trained using the training dataset and after it return the anomaly score of each sample: the outliers will be automatically identified.

5.2.3 Unbalanced Dataset

In the considered case study, a classification predictive modelling problem where the distribution of instances across the classes is not equal. the classes are not equally represented: the instances related to the “Not Lock” are much more compared with “Lock” events. This is very challenging because of the poor predictive performance of the algorithm for the minority class (that is the most important).

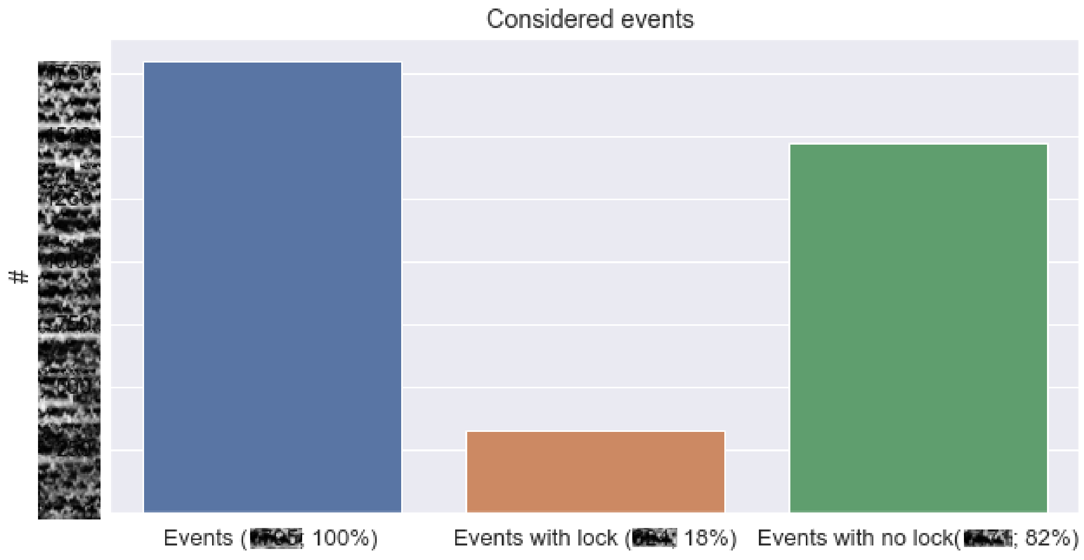


Figure 58 Unbalanced training dataset

It’s worth noting that the cause of the class imbalance is not a “Biased Sampling”: the training dataset is actually a fair representation of the problem domain, and so it is not possible balance the dataset by collecting more or better data.

Imbalance is a challenging situation, especially in our case study characterized by a severe imbalance: the goal of this study is to predict the Addition Heater Water lock, hence the minority class must be predicted. However, this is challenging because there are few examples of this class and the model is not able to learn its characteristics and differentiate the two classes: its learning process is often biases toward the majority class, so that the minority ones are neglected.

In order to combat Imbalanced classes in the dataset and considered what aforementioned, the performance metric was changed. The F1 Score is the choice, that is a weighted average of precision and recall and so capable of telling a more truthful story about the model's performances.

Moreover, a more balanced dataset was achieved by means of several different techniques [18]:

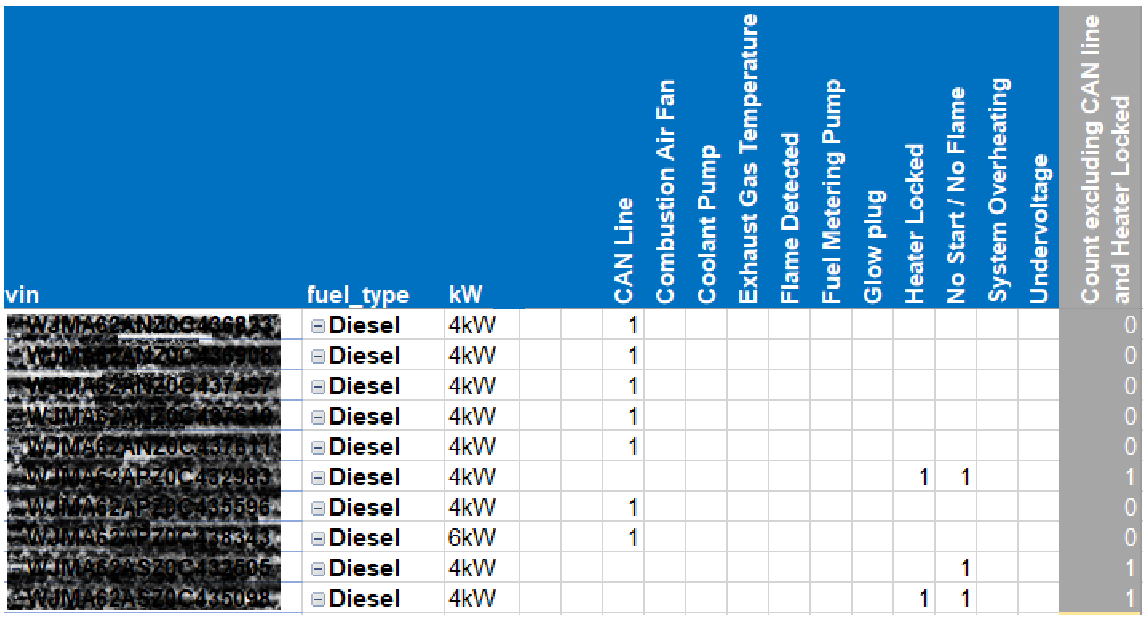
- Resampling: by means of random oversampling or random undersampling (or a combination of them), is possible to add copied of instances of the minority class and delete instances from the majority class, respectively;
- Generation of synthetic samples: SMOTE algorithm is used to create samples (and not copy)

5.3 Exploratory Data Analysis

Exploratory Data Analysis refers to the phase in which initial investigations about the phenomenon are performed in order to discover patterns, spot anomalies and check assumptions with the help of summary statistics or graphical representations. In other words, exploratory data analysis is a phase in which data scientists summarize the main characteristics of the dataset deciding the best way to manipulate data.

The main purpose is look at data before making any assumptions, ensuring that the obtained results are applicable without restrictions.

First of all, an analysis about co-occurrences was made: for each Vehicle Identification Number (and so for each vehicle) was performed a study about the Additional Water Heater DTCs present at the same time: in order to analyse a problem with a reduced dimensionality, all diagnostic trouble codes are clustered into categories.



vin	fuel_type	kW		CAN Line	Combustion Air Fan	Coolant Pump	Exhaust Gas Temperature	Flame Detected	Fuel Metering Pump	Glow plug	Heater Locked	No Start / No Flame	System Overheating	Undervoltage	Count excluding CAN line and Heater Locked
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	4kW									1	1			1
[redacted]	⊟Diesel	4kW		1											0
[redacted]	⊟Diesel	6kW		1											0
[redacted]	⊟Diesel	4kW										1			1
[redacted]	⊟Diesel	4kW									1	1			1

Table 2 *DTCs co-occurency*

The results were summarized as follows:

	% VINs
Only Can or Heater Locked	20%
Only one category	78%
Two categories	3%
Three categories	0%

Table 3 *DTCs co-occurency*

By means of this procedure, it was possible identify the top issues and failure patterns thanks to the telematics data:

Category	Volumes
No Start / No Flame	34
CAN Line	26
No Start / No Flame	26
Heater Locked	9
No Start / No Flame	3
Coolant Pump	2

Table 4 *DTCs occurency*

Moreover, DTCs occurrences and Additional Water Heater Lock are linked. The diagnostic trouble codes that cause the Additional Water Heater Lock were identified:

Co-occurrences with Heater Locked	% VINs
CAN Line	46%
Combustion Air Fan	1%
Coolant Pump	19%
Exhaust Gas Temperature	0%
Flame detected	0%
Fuel Metering Pump	0%
Glow Plug	0%
No Start / No Flame	90%

Table 5 *Co-occurency with Additional Water Heater lock*

Another goal of the exploratory data analysis was the correlation of the failure patterns with special vehicle configuration.

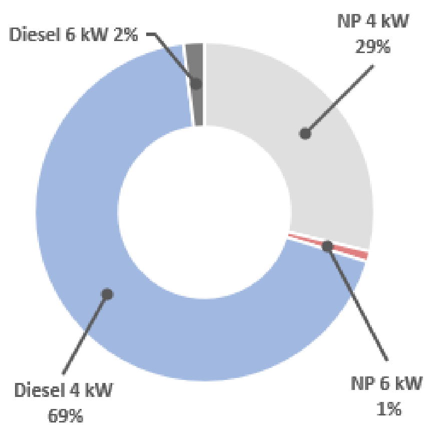


Figure 59 *Vehicle fleet and fuel type*

This forward step is justified by a major difference in the configuration. As previously stated, the Additional Water Heater performs the function of preheating the passenger compartment and the engine working independently of the latter. In particular, the fuel dosing pump allows the transfer of the fuel from the tank to the injectors: in Diesel vehicles, the tank is the principal one, while in the Natural Power Vehicles there is a dedicated tank. It worth analyse if this configuration difference is negligible or not. Therefore, the reference distribution based on the fleet of produced vehicles with a specific fuel and a specific Additional Water Heater is considered: each cell should be equal to the reference distribution, otherwise an imbalance exists.

Category	**NP-4kW 29%**	**NP-6kW 1%**	**Diesel-4kW 69%**	**Diesel-6kW 2%**
CAN Line	48%	0%	51%	1%
Coolant Pump	30%	0%	68%	3%
Heater Locked	53%	1%	44%	2%
No Start / No Flame	31%	0%	68%	0%
No Start / No Flame	75%	2%	20%	3%
CAN Line	29%	0%	67%	4%

Table 6 *Additional Water Heater configurations and vehicle fuel type*

If the distribution is not respected, a specific issue affects more some specific vehicles: relative values are more important with respect to absolute values since the distribution of the different Additional Water Heater configurations are not even distributed in the fleet. From the previous table, it seems that differences exist in the diagnostic trouble codes behaviour in the different fuel type, and not in different configurations. This means that there are no different issues in different configurations of the Additional Water Heater, but that there are issues related to the particular fuel type of the vehicle, e. g. the particular configuration in which the component works.

Therefore, a more in-depth study was carried out about diagnostic trouble codes telematic occurrences, ECU occurrences and duration in diesel and natural power vehicles. First of all, diagnostic trouble codes belonging to the "No Start / No Flame" cluster were considered. About ECU occurrences behaviours:

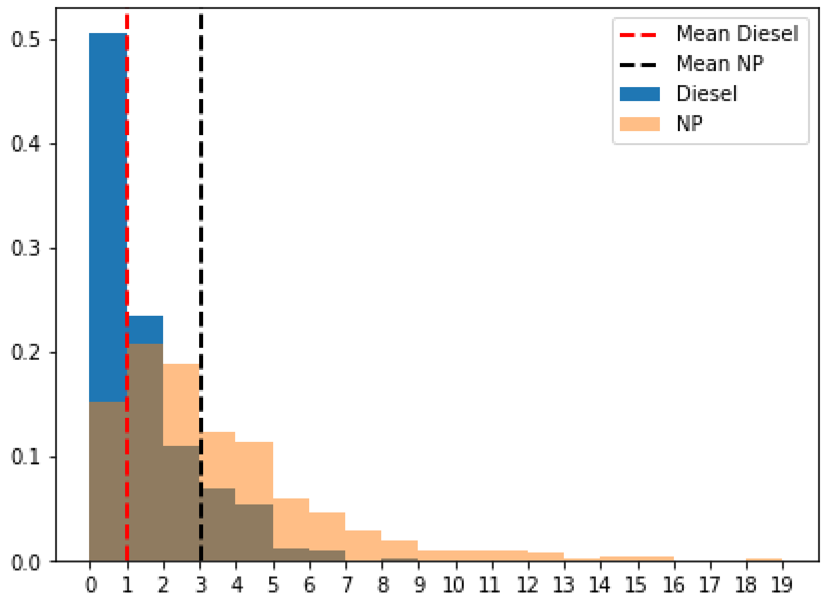


Figure 60 *Specific No Start/No Flame DTC occurences (ECU) without lock*

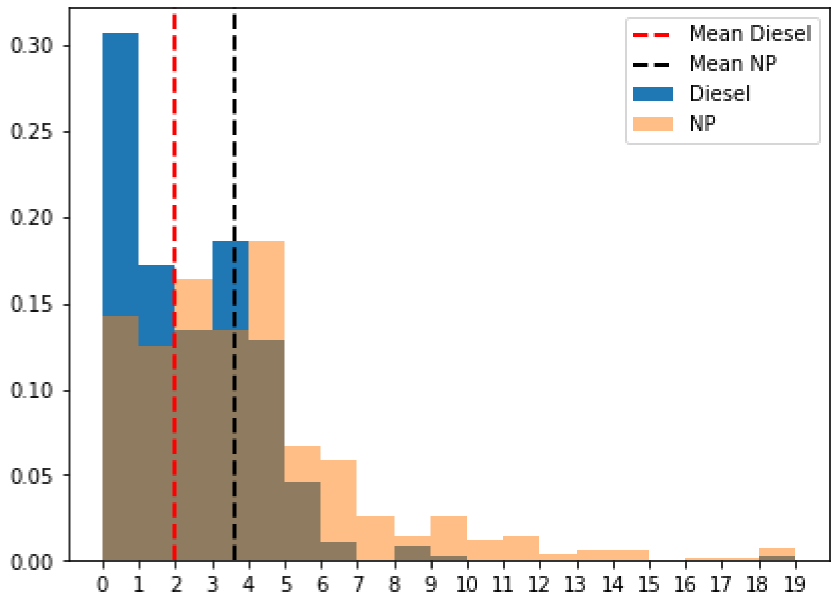


Figure 61 *Specific No Start/No Flame DTC occurences (ECU) with lock*

It is possible to state that differences in ECU occurrences exists in the two fuel type configuration: the two distributions are not superimposable and so different occurrences are needed for the lock in the two configurations. Same reasoning can be applied to the telematics occurrences:

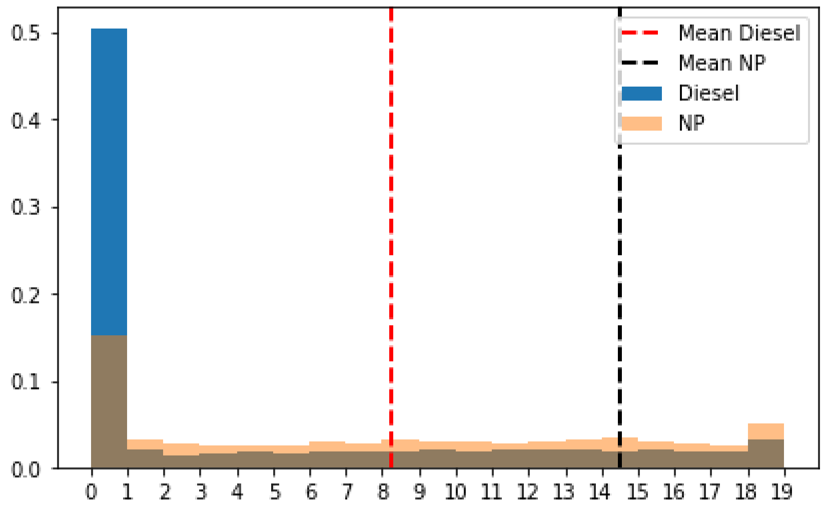


Figure 62 *Specific No Start/No Flame DTC occurences (Telematics) without lock*

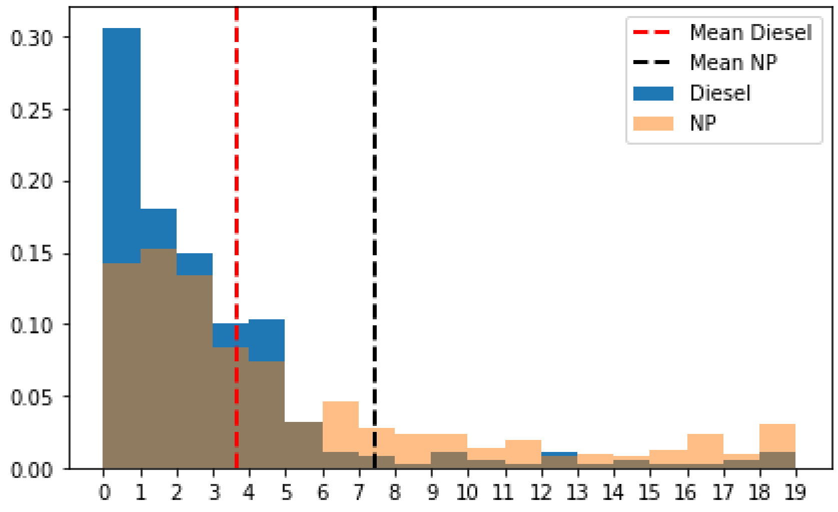


***Figure 63** Specific No Start/No Flame DTC occurences (Telematics) with lock*

Also, in this case, the distributions (and the relative mean) are far from themselves. Taking a look at the duration of the overall occurrences, it is possible to have another proof of the different behaviour:

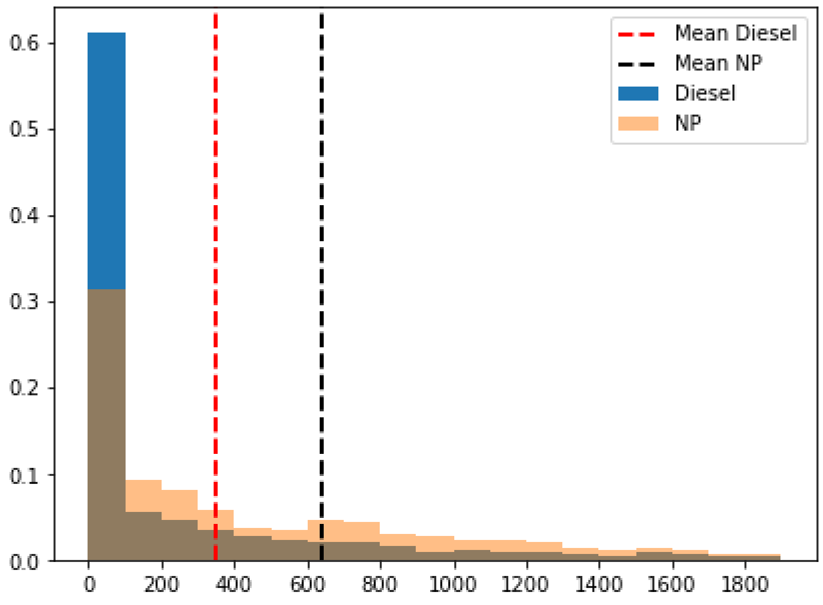


***Figure 64** Specific No Start/No Flame DTC duration without lock*

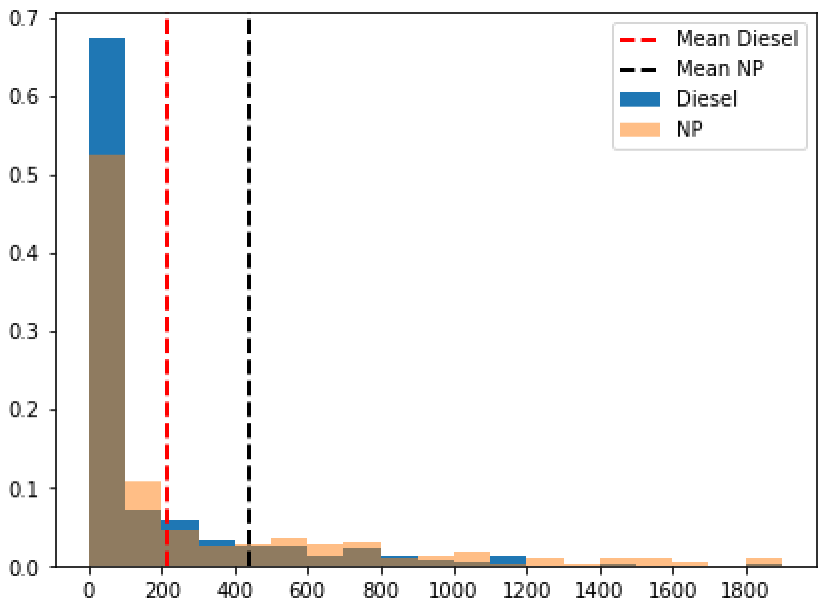


Figure 65 *Specific No Start/No Flame DTC duration with lock*

The output of the present analysis is the following: two different approaches are needed for the two fuel type configurations due to the fact that the events related to the different fuel type vehicles follow different distributions. In this dissertation only the Additional Water Heater mounted on diesel vehicles are considered: these represent more than 70% of the Additional Water Heater in the fleet, and so the effort is focused on diesel vehicles first.

5.4 Training Model and Test Phase

In the training phase, the goal is to build a model that can learn from and make predictions on data. Initially, the model is fit on a training dataset, that is a set of examples used to fit the parameters of the chosen model: in the studied case about Additional Water Heater, a training dataset includes different combinations of DTCs telematics occurrences, DTCs ECU occurrences and DTCs duration. Moreover, for each event is specified the vehicle's fuel type and the odometer (as we are going to see are very important parameters) and if the Additional Water Heater is locked or not.

Therefore, exploiting a supervised learning method and a pair of input and corresponding output, it is possible to identify patterns in the data and foresee if there will be an Additional Water Heater lock with a specified combination of DTCs parameters and vehicle's characteristics.

In particular, there is the train—test split procedure used for estimating the performance of machine learning algorithm when used on unseen data. The original dataset is divided into two subsets: the first one (Train Dataset) is used to fit the model (training phase), while the second one (Test Dataset) is used to evaluate the fit by comparing the predictions and the expected results, and so estimating the performance on unseen data. The split percentage must be chosen in according to:

1. Computational cost in training the model
2. Computation cost in evaluating the model
3. Training set representativeness
4. Test set representativeness

In the present work, the Train Dataset is the 90% of the original one: the goal is to have limited loss of information while training the algorithm because there will be a validation phase. Therefore, the Test Dataset will be 10% of the original one: it is exploited only to have guidelines about which algorithm to choose, or to get hints about the behaviour of the model after decisions about features, original data manipulation, optimization, etc.

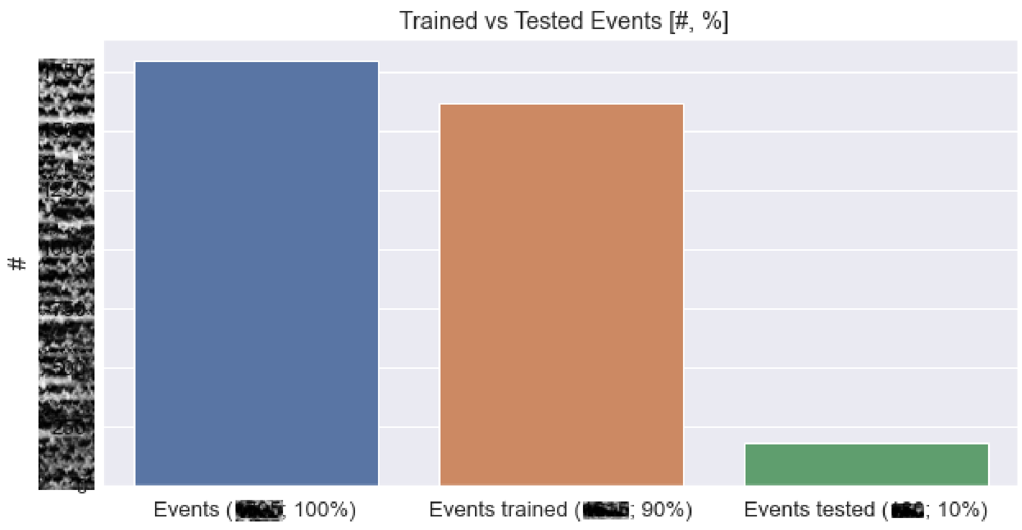


Figure 66 *Train-Test split percentage*

The train-test procedure is applicable when there are enough data to split the dataset into train and test datasets: each of them should be a proper representation of the problem domain, that is have enough records that cover all common and uncommon cases.

The next step consists in the model selection for machine learning: as previously sentenced, the scikit-learn library offers many models and the challenge is how to choose among them in order to achieve the target. It is worth noting that not only performance must be considered: other issues arise when evaluate the time necessary to train or test the model, or how easy it is to explain the model to colleagues that are not familiar with Artificial Intelligence.

In order to select the best model, the following requirements are considered:

1. Focus on Precision: Control Room triggers are characterized by a precision at least of 80%;
2. Focus on Recall: a good sensibility should be achieved in identifying the potential lock of the Additional Water Heater taking into account a large percentage of the fleet;
3. Moderate computational resources and time: generally, higher accuracy means higher training time; moreover, large training data mean higher training time. It should be considered that the lock is foreseen twelve hours

in advance: the time needed for the gathering phase and deployment should be small enough in order to allows the Customer Centre to arrange the workshop stop in time;

4. Moderate complexity: the model must be clearly explained to other IVECO departments at different hierarchical levels.

From a general point of view, each algorithm has its own advantages and disadvantages: "it is data that decides model, not us. Our task is to find the perfect model". First of all, the Logistic Regression is considered: it is easy to implement, very efficient to train and easy to interpret: the model coefficients that are associated to the variables can be seen as indicators of feature importance. Moreover, it makes no assumptions about distributions of classes in the feature space, but it is strongly limited by the linear boundaries that are considered. Implementing the logistic regression within the Additional Water Heater case study, the following results are achieved:

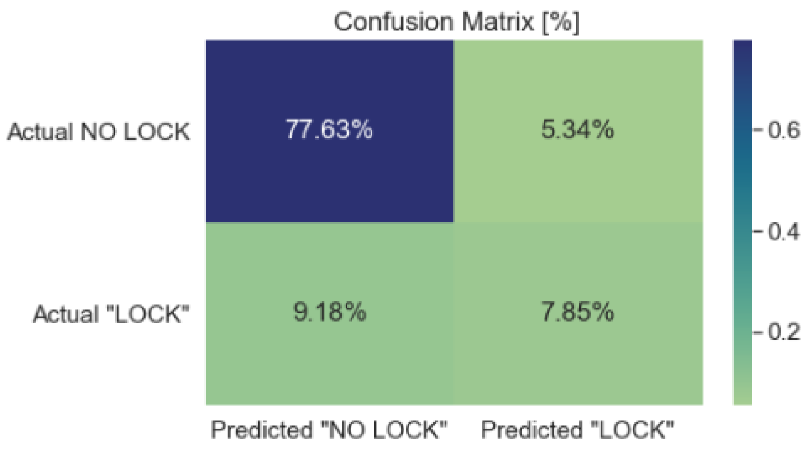


Table 7 Logistic Regression implementation: confusion matrix

```
              precision    recall  f1-score

           0       0.89      0.94      0.91
           1       0.59      0.46      0.52

    accuracy                           0.85
   macro avg       0.74      0.70      0.72
weighted avg       0.84      0.85      0.85
```

Table 8 Logistic Regression implementation: classification report

From the results, it is possible to conclude that the problem cannot be completely defined linear: the Logistic Regression seems to perform well on “0 cases”, that are cases in which the Additional Water Heater is not lock. About the “1 cases”, and so events characterized by the lock of the Additional Water Heater, a precision of 59% is get, with a recall of 46%. The result is not satisfying non-linear problems cannot be solved with Logistic Regression, but in real-world scenarios is difficult to find linearly separable data.

Therefore, from now on non-linear algorithms are used. The K-Nearest Neighbours algorithm is considered. Implementing KNN, it must be taken into account that it relies on majority voting based on class membership of K nearest samples with respect to the test point: it is a distance based algorithm, and so it is affected by the scale of the variables. Therefore, data normalization is needed: the algorithm should not be biased towards variables with higher magnitude. For each observation, the mean is subtracted and then divided by the standard deviation:

$$z = \frac{x - \mu}{\sigma}$$

Then, an instance of the class KNeighorsClassifier is created and the parameter K is initialized. Choosing the optimal value of K is critical: it is necessary to fit and test the model for different values of K using a for loop plotting the achieved accuracy.

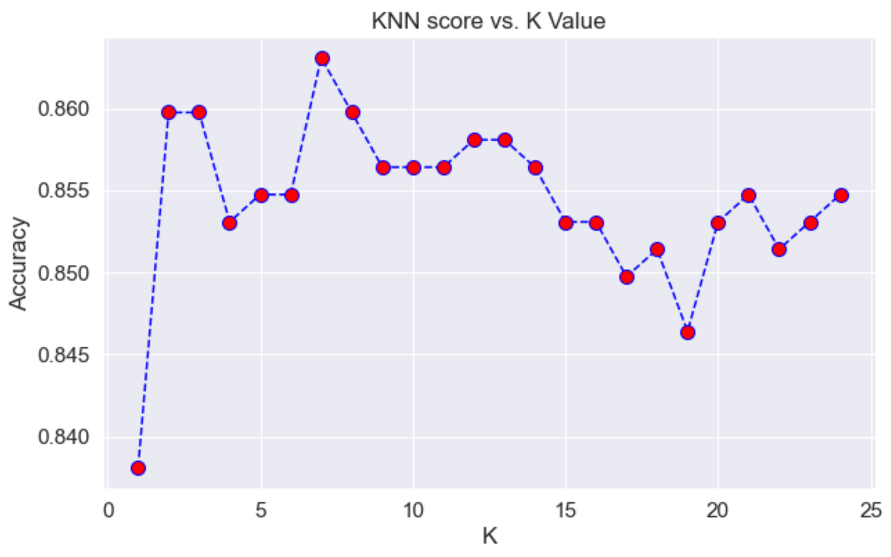


Figure 67 Accuracy and K parameter comparison

As we can see, there is a raise and fall in the accuracy. In general, as the value of K increases there is a raise in the accuracy because: larger K means increased complexity and smoother decision boundary but can lead to overfitting; small value of K can lead to underfitting: accuracy penalizes both and the optima K correspond to the right level of complexity. In this case study, a value of 13 is assigned to K: the corresponding accuracy is about 67%. The performance is not suitable:

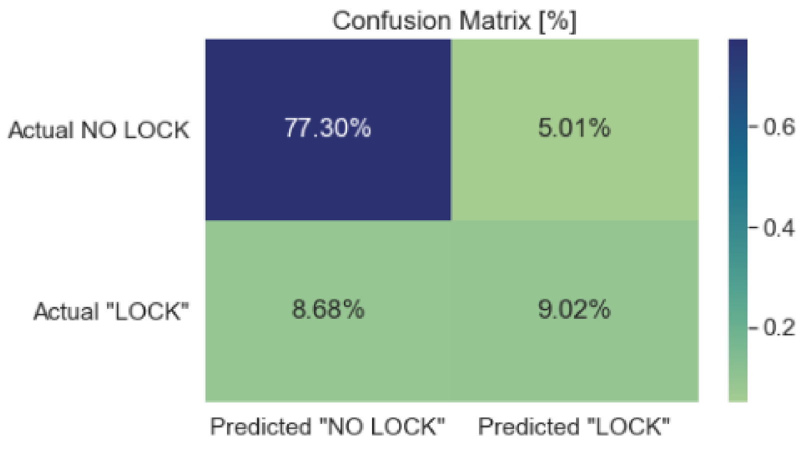


Table 9 *KNN implementation: confusion matrix*

```
              precision    recall  f1-score

           0       0.89      0.94      0.92
           1       0.64      0.46      0.54

    accuracy                           0.86
   macro avg       0.76      0.70      0.73
weighted avg       0.85      0.86      0.85
```

Table 10 *KNN implementation: classification report*

Also in this case no assumptions about data are made and the algorithm is easy to interpret. However, as it is possible to see from the classification report, there is only a slightly increase in precision (64%) while the recall remains steady (46%) in the events characterized by the lock of the Additional Water Heater. One of the reasons for a not high performance can be linked to the imbalance in the classes, or it can be due to relevant number of features.

Then, the Decision Tree is considered: it is also a non-parametric method that it easy to interpret. However, Decision Trees are prone to overfitting especially when a tree is very deep. Moreover, it is a greedy model: at each step, the algorithm chooses the best local results that can also be not optimal from an absolute point of view. Random Forest solve these issues: as previously stated, it is a collection of decision trees whose results are aggregated into one finale result, limiting overfitting and reduce room for errors. Therefore, Random Forests are a strong modelling technique much more robust with respect with a single decision tree, but they are also less interpretable.

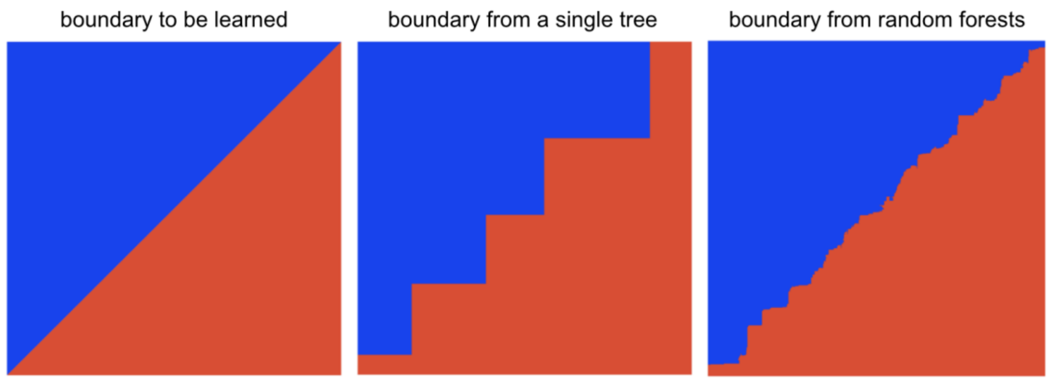
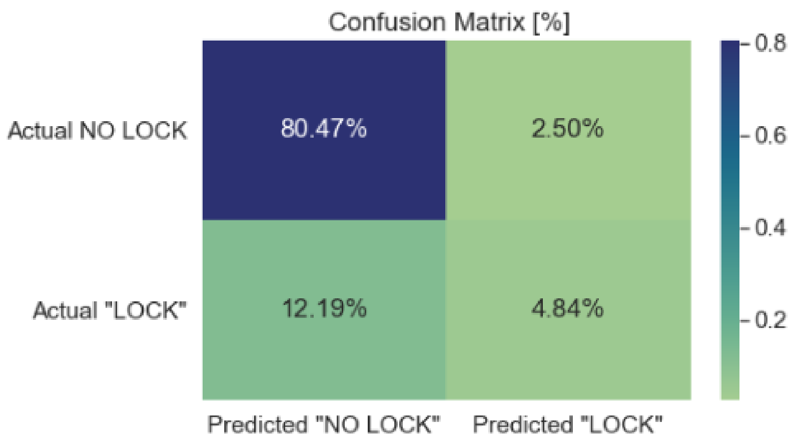


Figure 68 *Decision Tree and Random Forest performance comparison*

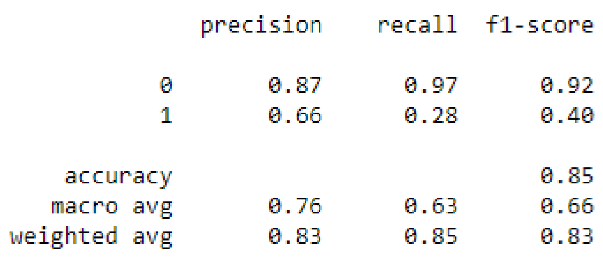
It is possible to consider the result of Random Forest implementation within Additional Water Heater case study:



Confusion Matrix [%]

	Predicted "NO LOCK"	Predicted "LOCK"
Actual NO LOCK	80.47%	2.50%
Actual "LOCK"	12.19%	4.84%

Table 11 *Random Forest implementation: confusion matrix*



```
              precision    recall  f1-score

           0       0.87      0.97      0.92
           1       0.66      0.28      0.40

    accuracy                           0.85
   macro avg       0.76      0.63      0.66
weighted avg       0.83      0.85      0.83
```

Table 12 Random Forest implementation: classification report

Looking at the results, it seems that Random Forest does not work properly. One of the main problems could be linked to the sparsity of the features. Data is considered sparse when certain expected values in a dataset are most likely zero: in the present case study, around 15 different DTCs are considered, and even when the additional water heater lock occurs, only a few of them is characterized by values different from zero: it depends on the particular problem that the component is facing.

Then, Gradient Boosting is implemented. Like Random Forests is a set of decision trees but differences exist about how trees are built and how the results are combined. Gradient Boosting can overcome Random Forest in performance: on the other side, they are more difficult to tune and the training generally takes longer. At the beginning, the default parameters are chosen (both for the diesel and natural power model):

Hyperparameter	Value
max_depth	3
n_estimators	100
learning_rate	0.1
colsample_bytree	1
gamma	0
subsample	1
random_state	42

Table 13 XGBoost baseline parameters

The results are summarized in the following confusion matrix and classification report:

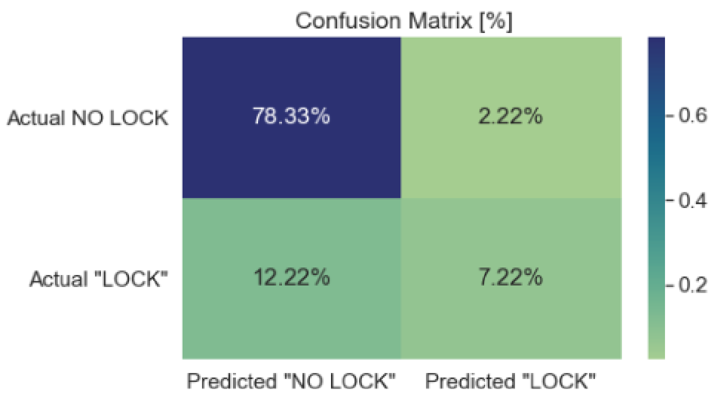
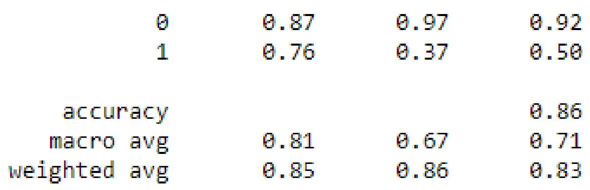


Table 14 *XGBoost implementation: confusion matrix*



```
           0       0.87      0.97      0.92
           1       0.76      0.37      0.50

    accuracy                           0.86
   macro avg       0.81      0.67      0.71
weighted avg       0.85      0.86      0.83
```

Table 15 *XGBoost implementation: classification report*

Therefore, Extreme Gradient Boosting performs better with respect to the other models: as it is possible to see from the classification report, there is an important increase in precision (76%) while the recall remains to 36% in the events characterized by the lock of the Additional Water Heater.

As previously stated, even within a classification problem it can be more convenient to predict probabilities rather than classes: different thresholds could be used dealing with errors made by the model. In the present case study this concept is very important: the goal should be maximising both precision and recall, but from the real application point of view only a balance can be achieved. In particular, the diagnostics tool that help in the interpretation of probabilistic forecast are ROC and the Precision-Recall curves: the last one is considered within this dissertation because it is suitable for cases where there is an imbalance in the observations between the two classes (only the fraction of true positives among positive predictions is considered).

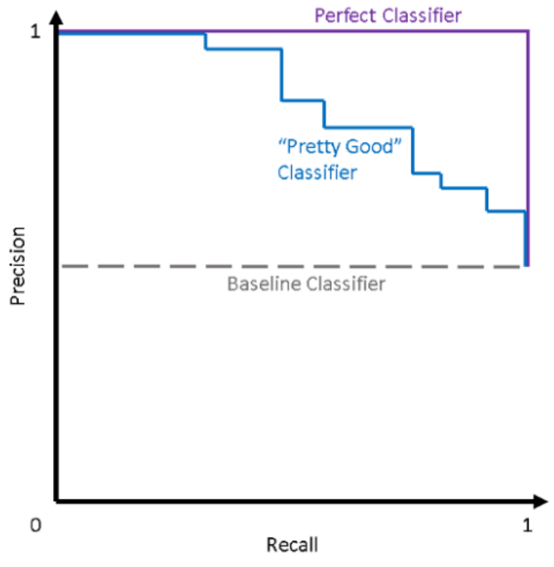


Figure 69 Precision – Recall curves

As the matter of fact, due to the large skew in the class distribution ROC curves can present an optimistic view of the algorithm's performance, leading to an incorrect interpretation of the model skill because of the use of the true negatives. A Precision-Recall curve is a plot of the precision versus the recall for different thresholds. Taking into account the baseline XGBoost model, the following Precision-Recall curve can be considered for the diesel fuel type:

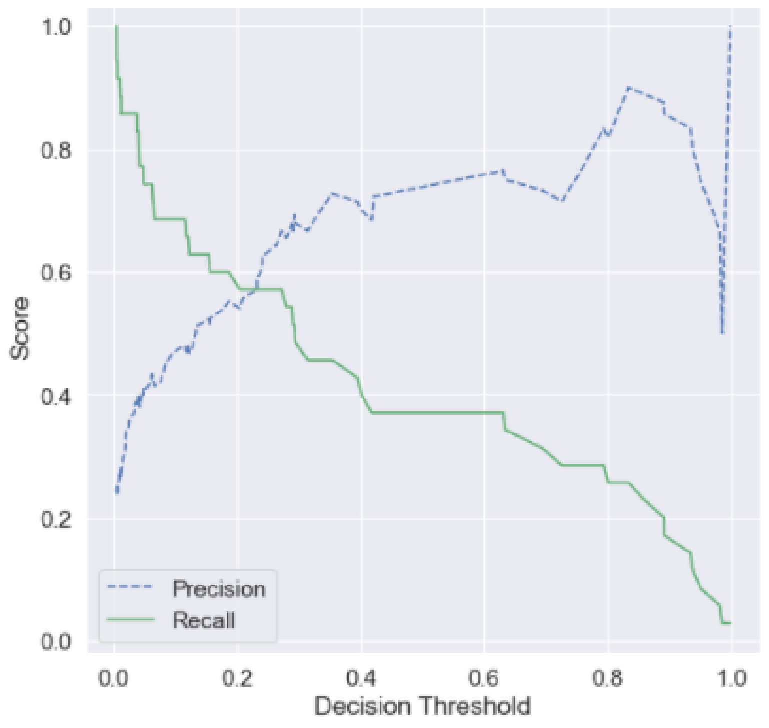


Figure 70 Precision and Recall scores before optimization

5.5 Optimization

As previously stated, machine learning involves using an algorithm to learn and then generalize from a training dataset in order to make predictions on new and unseen data. It's worth noting that machine learning algorithms are characterized by some hyperparameters that can be set to tailor the algorithm to the specific dataset: hyperparameters must be set manually, while the model parameter are the ones learned automatically.

Different approaches are available for the so called "hyperparameter optimization", as described in the following. A first strategy consists in search different values for the hyperparameters and choose the best subset that allows to achieve the best performance. A search space must be defined, with several dimensions (corresponding to the different hyperparameters) and scale of dimensions (corresponding to the specific value that the hyperparameter takes). Therefore, the optimization goal is to find a vector within the search space that results in the best performance of the model after learning: this process can be done iteratively exploiting several algorithms. Among the most common methods, it is possible to find [19]:

- ***Random Search***: random sample points are evaluated:
- ***Grid Search***: every possible combination is evaluated.

These two approaches have different pros and cons. First of all, Grid Search approach starts becoming very time-consuming if there are many hyperparameters and huge search space assuming that the list of candidates is properly chosen. In order to deal with this problem, it's possible to rely on randomness through the Random Search that does not ensure that the best combination is found.

In the present dissertation, the optimization phase can also take hours: normal personal computers are used without dedicated equipment, and so a smarter approach was found. Within the XGBoost algorithm, there is the possibility to tune the performances with "Learning Curves": at each iteration the results are plotted, and it is possible to interpret them understanding hyperparameters changes that should be done.

The starting point is the set of hyperparameters that characterize the XGBoost. Then, a standard procedure is adopted [20]:

1. Set initial reasonable values (at the beginning, the default ones):
    - ***learning_rate;***
    - ***n_estimators;***
    - ***max_depth;***
    - ***subsample;***
    - ***colsample_bytree;***
    - ***gamma;***
2. Set the proper evalutaion metric;
3. ***max_depth*** hyperparameter optimization: starting from a low max_depth, it was increased incrementally by 1, stopping when there's no performance gain. A balance between underfitting and overfitting should be always considered: a high depth of the tree is not advisable for complexity reasons, while a too low values can cause a "non-generalization" issue. Before the optimization process, the following plot can be considered:

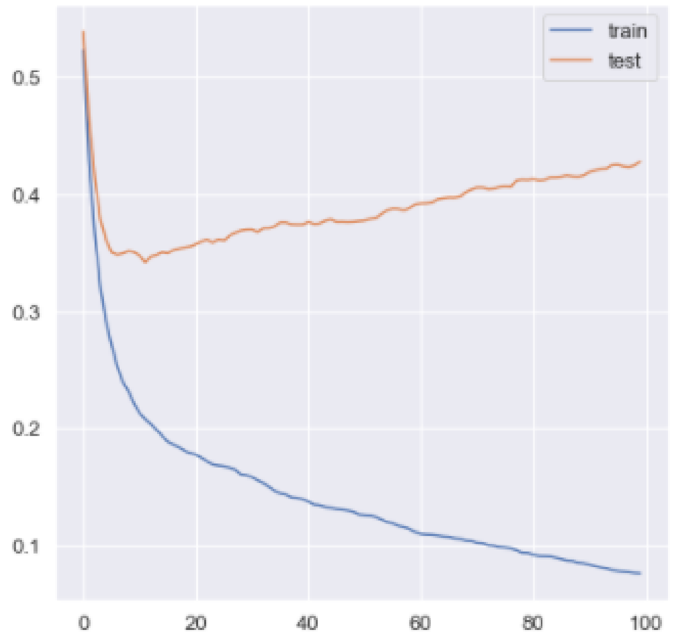


Figure 71 *logloss and iterations comparison before the optimization*

4. ***colsample_bytree*** evaluation, necessary in order to avoid that some columns become too much importance for the prediction: therefore, a good proportion of columns is taken out;
5. ***subsample***: percentage of rows taken out to build each tree (if too many rows are taken out, the performance will drop down). After the optimization, the following plot can be considered:

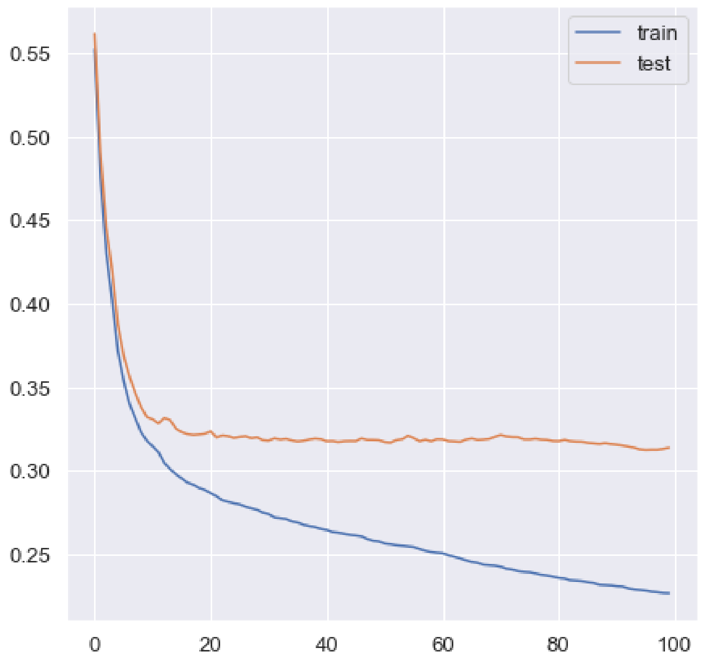


Figure 72 logloss and iterations comparison after the optimization

6. ***n_estimators***: performance of the model on the training dataset and on the test dataset are considered with respect to the number of estimators. Considering the curves, it is possible to understand if a better performance is achievable by adding more iterations. In other words, if the curve has reached a plateau, adding more iterations has no sense;
7. ***learning_rate*** evaluation: a balance between performance and learning time should be achieved.

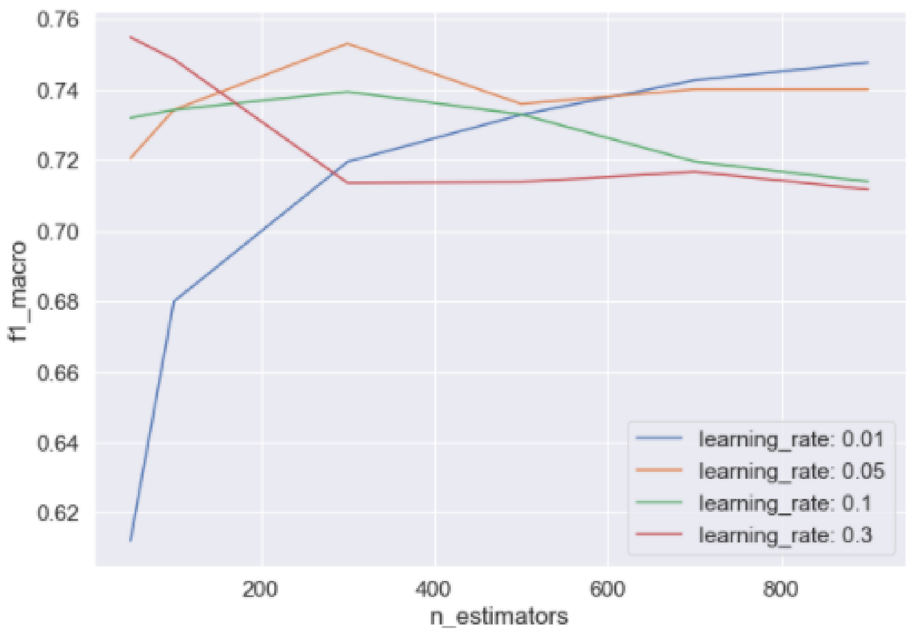


Figure 73 Learning rate and estimators number in diesel vehicles

Therefore, the optimal hyperparameters for the diesel model are:

Hyperparameter	Value
max_depth	2
n_estimators	100
learning_rate	0.3
colsample_bytree	0.6
gamma	0
subsample	1
random_state	42

Table 16 XGBoost optimal parameters

After the optimization process previously described:

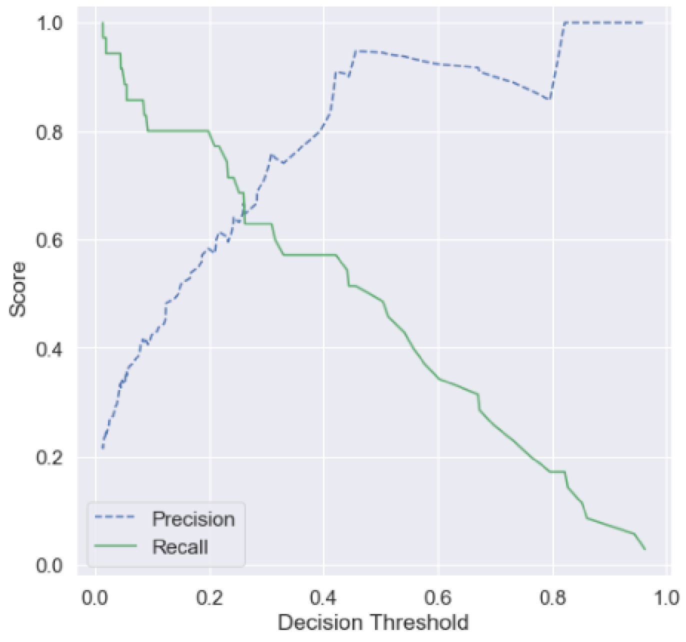


Figure 74 *Precision and Recall scores after optimization*

It is possible to summarize the model performance by means the classification report and the confusion matrix:

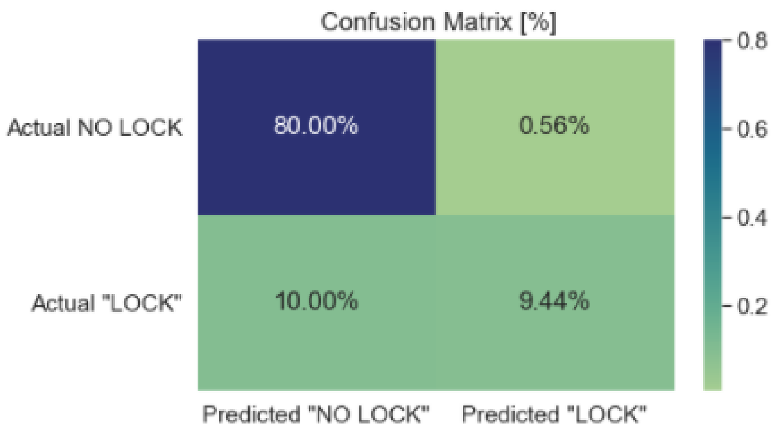


Figure 75 *Optimized XGBoost: confusion matrix*

```
           0       0.89      0.99      0.94
           1       0.94      0.49      0.64

    accuracy                           0.89
   macro avg       0.92      0.74      0.79
weighted avg       0.90      0.89      0.88
```

Table 17 *Optimized XGBoost: classification report*

5.6 Validation Phase

The optimized algorithm was further tester in a real world environment: two months data were taken into account in order to ensure that the algorithm performs as expected: within the Control Room procedures, there is an internal test phase in order to test triggers before the actual deployment.

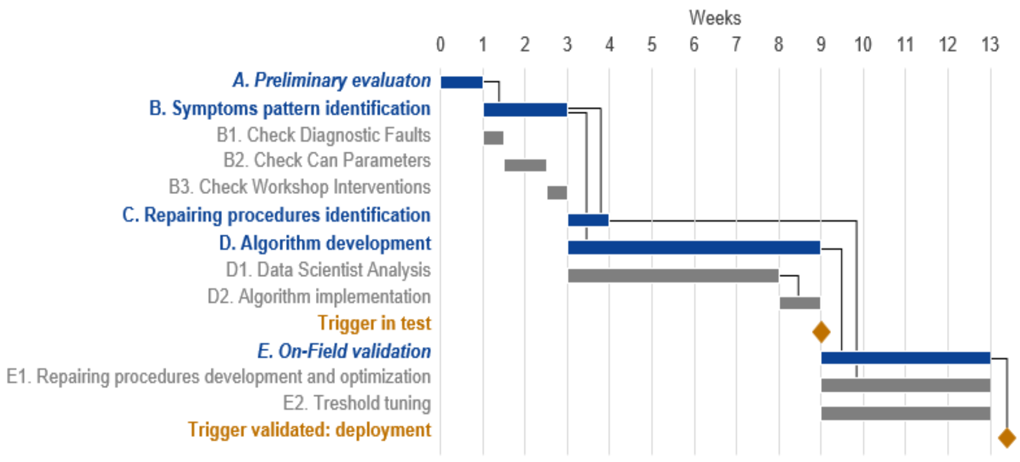


Figure 76 Gantt: chart: complex trigger development

The validation is performed on the Additional Water Heater data of April 2021 and May 2021. The developed algorithm has been tested on a totally new dataset in order to assess the actual performance with an on-field validation before the actual deployment.

The validation phase was conducted as follows in order to be compliant with the actual strategy of deployment of the algorithm. First of all, a considerable number of time checkpoints has been established (order of magnitude: several checkpoints per hour for each day of the identified validation time interval). At each time checkpoints the algorithm is fed with:

- the details about the particular vehicle configuration;
- the telematics data related to the last fifteen days.

The output is the Additional Water Heater Status (i.e. Locked or Not Locked) twelve hours later: then, the forecast is compared with the actual status and the results are the following:

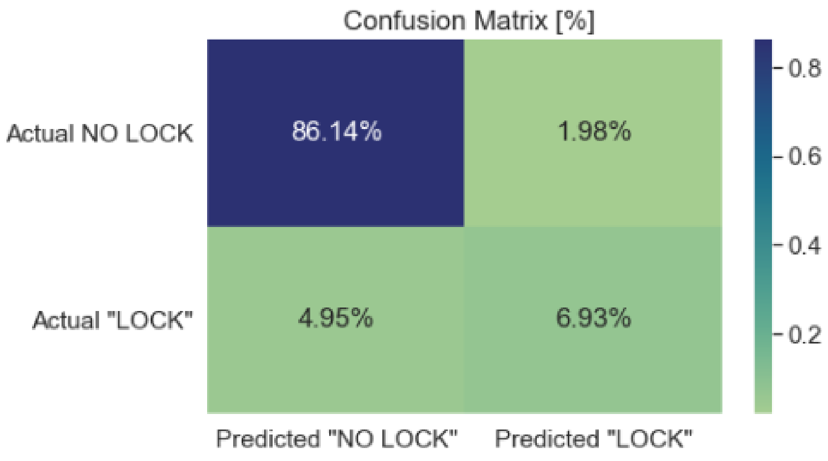


Table 18 *Optimal XGBoost implementation: confusion matrix*

```
              precision    recall  f1-score

           0       0.95      0.98      0.96
           1       0.78      0.58      0.67

    accuracy                           0.93
   macro avg       0.86      0.78      0.81
weighted avg       0.93      0.93      0.93
```

Table 19 *Optimal XGBoost implementation: classification report*

6 Conclusions

6.1 Results

As stated, the new S-Way is the first vehicle in the new Iveco Way heavy range developed to deliver a package of features and services focused on an advanced level of connectivity: these features are fully integrated into the ecosystem in order to exploit digitalisation, automation, and vehicle electrification to anticipate and meet the demands.

In this context works the Control Room Team that analyses real-time data coming from the the ECUs to maximise vehicle's uptime and productivity through a proactive approach. The current output of this approach is a *Standard Alert Development* based on vehicle symptoms study, previous breakdowns analysis and statistical analysis, while the present dissertation is about the first *Complex Alert Development* by exploiting more advanced tools related to Machine Learning field.

Therefore, this work can be considered a *Proof of Value*: among the goals of the dissertation there is the introduction of these new technologies within the Control Room in order to enable a sophisticated investigation about the phenomena of the Additional Water Heater Interlock by analyzing complex and large data sets. As described in *Chapter 5.3: Exploratory Data Analysis* this goal was largely satisfied, and several interesting patterns were discovered.

First of all, it has been found out that Coolant Pump DTCs, No Start / No Flame DTCs and Additional Water Heater Interlock were highly correlated. This result was the starting point for advanced technical inspections about Locked Additional Water Heater: corrective actions from a design point of view were made in order to ensure an higher level of reliability. These corrective actions (without describing the details to preserve confidentiality) were based also on the particular AWH configuration (4 kW or 6 kW) and fuel type (the layout of the fuel system slightly changes). A very important outcome was the following: by studying the DTCs distributions about time duration, telematics occurrences and ECU occurrences, differences exist in the diagnostic trouble codes behaviour in the different fuel type, and not in different configurations. This means that there are no different issues in different configurations of the Additional Water Heater, but there are issues related to the particular fuel type of the vehicle, e. g. the particular configuration in which the component works: the Additional Water Heater Interlock is caused by different factors in Diesel and Natural Power Vehicles.

However, these corrective actions affect vehicles that will be assembled in the future or already working vehicles that replace the Additional Water Heater components under investigation, that is why the algorithm discussed in the present dissertation was developed.

As Stated, Control Room's work aims at identifying patterns able to recognize potential breakdowns with a precision higher than 95% (average). Therefore, the following requirements were fixed for the development of the Complex Trigger under examination:

1. Optimum Precision: Control Room triggers are characterized by a precision at least of 80%;
2. Less than Optimum Recall: a good sensibility should be achieved in identifying the potential lock of the Additional Water Heater considering a large percentage of the fleet;
3. Moderate computational resources and time: generally, higher accuracy means higher training time; moreover, large training data mean higher training time. It should be considered that the lock is forecasted twelve hours in advance: the time needed for the gathering phase and deployment should be small enough in order to allows the Customer Centre to arrange the workshop stop in time;
4. Moderate complexity: the model must be clearly explained to other IVECO departments at different hierarchical levels.

All these goals are actually achieved by means of the built model: the performance is superior with respect to the one expected, with a forecast of a potential lock of the Additional Water Heater of about 12 hours.

6.2 Improvements

Enhance the performance of the machine learning model should be always the final goal. In the near future, performance could be improved by means of data: this approach consists in create new and different perspective on the data in order to better identify the underlying problem. Therefore, one approach could consist in getting more data characterized by better quality: there are modern nonlinear machine learning techniques nowadays available that works well with larger dataset with respect to the one used in this dissertation. So, more data means the possibility to improve performance by exploiting new algorithms tailored on the new data characteristics: deep learning can be an option.

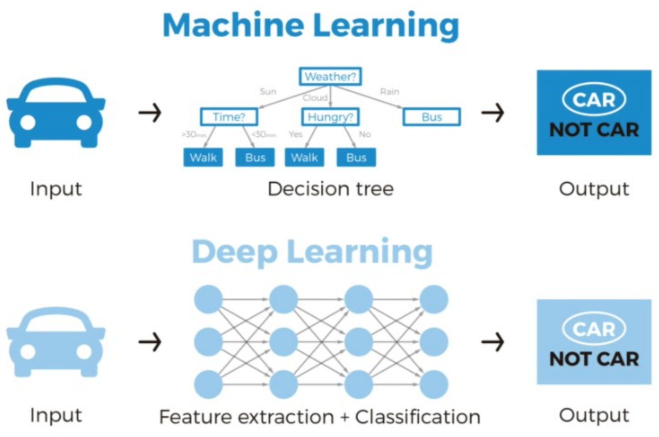


Figure 77 Machine Learning and Deep Learning comparison

Deep learning goal is to simulate the human brain enabling systems to cluster data and make predictions with incredible accuracy exploiting large amounts of data: neurons are simulated by the nodes that allow information flow in a complex, multi-layered "deep neural networks". From one layer to the following one, the level of abstraction increases: it is much less explainable with respect to the machine learning algorithm seen in this work, but with the proper quantity of data the performance increase is important [21].

Moreover, deep learning is outperforming if there is a not complete knowledge about the problem domain: a different solving approach is adopted because there is no the need of features engineering phase and so the domain expertise is not so useful as in Machine Learning applications. Therefore, complex data can be shaped, with more features and more data: this should ensure better performance.

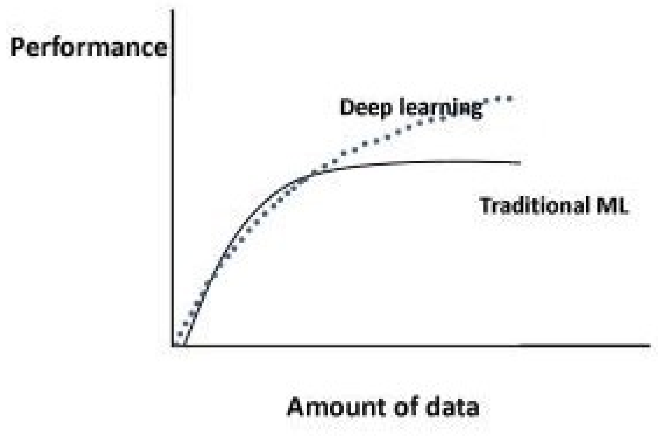


Figure 78 *Comparison between deep learning and machine learning performance*

7 Appendix

7.1 Customized functions

7.1.1 Considered_events

The considered functions allow to visualize by means of an histogram the distribution of the classes (Additional Water Heater locked versus not locked ones): the only input needed is the DataFrame.

```
def considered_events(dtc):
    sns.set(font_scale=1.2)
    plt.figure(figsize=(10,5))
    dtc_info =
pd.DataFrame({'#':[len(dtc.index),len(dtc[dtc['D4']==1]),len(dtc[d
tc['D4']==0])]},
                    index=['Events ({};
{}%)'.format(len(dtc.index),round(len(dtc.index)*100/len(dtc.index
))),
                           'Events with lock ({};
{}%)'.format(len(dtc[dtc['D4']==1]),\

round(len(dtc[dtc['D4']==1])*100/len(dtc.index))),
                           'Events with no lock({};
{}%)'.format(len(dtc[dtc['D4']==0]),\

round(len(dtc[dtc['D4']==0])*100/len(dtc.index)))])
    event_plot=sns.barplot(x=dtc_info.index,y='#',data=dtc_info)
    event_plot.set_title('Considered events ')
    sns.set(font_scale=1.2)
```

7.1.2 Considered_events_tts

The considered functions allows to visualize by means of an histogram the percentage split of the original dataset into the training and test dataset. The input are:

- Original DataFrame;
- Inputs of the Training DataFrame;
- Input of the Test DataFrame;
- Output of the Test Daataframe.

```
def considered_events_tts(dtc, X_train, X_test, y_test):
    sns.set(font_scale=1.2)
    plt.figure(figsize=(10,5))
    model_info =
pd.DataFrame({'#':[len(dtc.index),len(X_train),len(X_test)],

'%':[100,round(len(X_train)*100/(len(dtc.index))),round(len(X_test
)*100/(len(dtc.index)))]},
                    index=['Events ({};
{}%)'.format(len(dtc.index),100),'Events trained ({};
{}%)'.format(len(X_train),round(len(X_train)*100/(len(dtc.index)))
),'Events tested ({};
{}%)'.format(len(y_test),round(len(X_test)*100/(len(dtc.index))))]
)
    events_plot =
sns.barplot(x=model_info.index,y='#',data=model_info)
    events_plot.set_title('\n Trained vs Tested Events [#, %]')
```

7.1.3 Plot_confusion_matrix

The considered functions allows to visualize the confusion matrix with a customized format. The input are:

- Output of the Test Dataframe;
- Algorithm predictions.

```
def plot_confusion_matrix(y_test, predictions):
    sns.set(font_scale=1.4)
    plt.figure(figsize=(15,5))
    #CM - #
    plt.subplot(1,2,1).set_title('\n Confusion Matrix [#]')
    cm=confusion_matrix(y_test,predictions)

hm=sns.heatmap(cm,annot=True,fmt='d',cmap='crest',xticklabels=['Pr
edicted "NO LOCK"', 'Predicted "LOCK"'],
            yticklabels=['Actual NO LOCK', 'Actual "LOCK"'])
    hm.set_yticklabels(['Actual NO LOCK', 'Actual
"LOCK"'],rotation=0)
    #CM - %
    plt.subplot(1,2,2).set_title('\n \n \n Confusion Matrix [%]')
    hm=sns.heatmap(cm/np.sum(cm), annot=True, fmt='.2%',
cmap='crest',xticklabels=['Predicted "NO LOCK"', 'Predicted
"LOCK"'],
            yticklabels=['Actual NO LOCK', 'Actual "LOCK"'])
    hm.set_yticklabels(['Actual NO LOCK', 'Actual
"LOCK"'],rotation=0)
    plt.tight_layout()
```

7.1.4 Adjusted_classes

The considered functions allows to select custom classes by choosing a custom threshold that confines them. Therefore, the inputs are:

- Probabilites of each event to belonging to the "Lock Class" assigned by the algorithm;
- Treshold.

```
def adjusted_classes(y_scores, t):
    """
    This function adjusts class predictions based on the
prediction threshold (t).
    Will only work for binary classification problems.
    """
    return [1 if y >= t else 0 for y in y_scores]
```

7.1.5 Plot_precision_recall_vs_treshold

The considered functions allows to plot the precision and recall curves with respect to the thresholds. Therefore, the inputs are:

- Precision ndarray obtained by means of the “precision_recall_curve” tool;
- Recall ndarray obtained by means of the “precision_recall_curve” tool;
- Tresholds array.

```
def plot_precision_recall_vs_threshold(precisions, recalls,
thresholds):
    """
    Modified from:
    Hands-On Machine learning with Scikit-Learn
    and TensorFlow; p.89
    """
    plt.figure(figsize=(8, 8))
    plt.title("Precision and Recall Scores as a function of the
decision threshold")
    plt.plot(thresholds, precisions[:-1], "b--",
label="Precision")
    plt.plot(thresholds, recalls[:-1], "g-", label="Recall")
    plt.ylabel("Score")
    plt.xlabel("Decision Threshold")
    plt.legend(loc='best')
```

7.1.6 Select_Best_Feature_Number_Univariate

```
def Select_Best_Feature_Number_Univariate(x, y):
    scores_01 = []
    scores_10 = []
    scores_11 = []
    for i in range(x, y):
        feature_selector = SelectKBest(score_func=f_classif, k=i )
        X_train_select = feature_selector.fit_transform(X_train,
y_train)
        X_train.columns[feature_selector.get_support()]
        rf = RandomForestClassifier()
        rf.fit(X_train_select, y_train)
        X_test_select = feature_selector.transform(X_test)
        y_pred_rf_select = rf.predict(X_test_select)
        cm = confusion_matrix(y_test, y_pred_rf_select)
        scores_01.append(cm[0][1])
        scores_10.append(cm[1][0])
        scores_11.append(cm[1][1])
    plt.figure(figsize=(10,6))
    plt.plot(range(1,y), scores_01 ,color='blue',
linestyle='dashed', label='01')
    plt.plot(range(1,y), scores_10 ,color='red',
linestyle='dashed', label='10')
    plt.plot(range(1,y), scores_11 ,color='green',
linestyle='dashed', label='11')
    plt.title('Random Forest score vs. feature number')
    plt.xlabel('features')
    plt.ylabel('score')
    plt.legend(loc='best')
```

7.1.7 Select_Best_Feature_Number_RFE

```
def Select_Best_Feature_Number_RFE(x, y):
    scores_01 = []
    scores_10 = []
    scores_11 = []
    for i in range(x, y):
        feature_selector = RFE(XGBClassifier(),
n_features_to_select=i)
        X_train_select = feature_selector.fit_transform(X_train,
y_train)
        X_train.columns[feature_selector.get_support()]
        rf = RandomForestClassifier()
        rf.fit(X_train_select, y_train)
        X_test_select = feature_selector.transform(X_test)
        y_pred_rf_select = rf.predict(X_test_select)
        cm = confusion_matrix(y_test, y_pred_rf_select)
        scores_01.append(cm[0][1])
        scores_10.append(cm[1][0])
        scores_11.append(cm[1][1])
    plt.figure(figsize=(10,6))
    plt.plot(range(1,y), scores_01 ,color='blue',
linestyle='dashed', label='01')
    plt.plot(range(1,y), scores_10 ,color='red',
linestyle='dashed', label='10')
    plt.plot(range(1,y), scores_11 ,color='green',
linestyle='dashed', label='11')
    plt.title('Random Forest score vs. feature number')
    plt.xlabel('features')
    plt.ylabel('score')
    plt.legend(loc='best')
```

7.1.8 Select_Best_Feature_Number_PCA

```
def Select_Best_Feature_Number_PCA(x, y):
    scores_01 = []
    scores_10 = []
    scores_11 = []
    for i in range(x, y):
        feature_selector = PCA(n_components=i)
        X_train_unb_select =
feature_selector.fit_transform(X_train_unb, y_train_unb)
        rf = RandomForestClassifier()
        rf.fit(X_train_unb_select, y_train_unb)
        X_test_unb_select = feature_selector.transform(X_test_unb)
        y_pred_rf_unb_select = rf.predict(X_test_unb_select)
        cm = confusion_matrix(y_test_unb, y_pred_rf_unb_select)
        scores_01.append(cm[0][1])
        scores_10.append(cm[1][0])
        scores_11.append(cm[1][1])
    plt.figure(figsize=(10,6))
    plt.plot(range(1,y), scores_01 ,color='blue',
linestyle='dashed', label='01')
    plt.plot(range(1,y), scores_10 ,color='red',
linestyle='dashed', label='10')
    plt.plot(range(1,y), scores_11 ,color='green',
linestyle='dashed', label='11')
    plt.title('Random Forest score vs. feature number')
    plt.xlabel('features')
    plt.ylabel('score')
    plt.legend(loc='best')
```

7.1.9 Make_training

The “make_training” function takes the following input:

- Training dataset;
- Start date;
- End date;
- Target date.

The Start and the final dates define a time window: only the diagnostic trouble codes belonging to this period are considered, and it is evaluated if 12 hours after the closure of the time window there an Additional Water Heater or not. Then, the function aggregates all the events associated to the same vehicle identification number and create a Pivot Table in which occurrences (Telematics and ECU) and duration are taken into account. Moreover, within the same function multiple DataFrame are considered to add other information about the vehicle such as odometer and fuel type.

The structure of “make_training” function:

```
def make_training(training, start_date, end_date, target_date):

    # it gives all dtc in the time window
    training1 = 
(training[(training['firstDetectionTimestamp']>start_date)&(traini
ng['firstNoDetectionTimestamp']<end_date)]).copy(deep=True)


    #check if in the day of the training there was almost 1 dtc 
occurrence
    training1['Same_Day'] = ''
    for i in np.arange(0, len(training1.index)):
        support58 = (end_date-(training1.iloc[i, 
training1.columns.get_loc('firstNoDetectionTimestamp')]))/np.timed
elta64(1, 'h')
        if ((support58<24) & (support58>0)):
            training1.loc[training1.index[i], 'Same_Day'] = 
'same_day'
        else: training1.loc[training1.index[i], 'Same_Day'] = 
'not_same_day'

    training1 = training1[training1['Same_Day']=='same_day']

    #check if AHW locked between start date and end date
    training1['Status'] = ''
    training1_locked = 
training[(training['firstDetectionTimestamp']>start_date)\
&(training['firstDetectionTimestamp']<end_date)].query("SPNhex == 
'0x7E026'").index.unique()
    for i in np.arange(0, len(training1.index)):
        if training1.index[i] in training1_locked:
            training1.loc[training1.index[i], 'Status'] = 'Lock'
        else: training1.loc[training1.index[i], 'Status'] = 
'No_Lock'
    training1 = training1[training1['Status']=='No_Lock']

    #create params
    support54 = pd.DataFrame()
    nolock_vin_unique_training = 
pd.Series(training1.index).unique()
    for i in np.arange(0,len(nolock_vin_unique_training)):
        support57 = training1.loc[[nolock_vin_unique_training[i]]]
        odo_mean = support57['odometer_km'].mean()
        support55 = support57.pivot_table(index='VIN', 
columns=['SPNhex'], values=['ECUhex', 'OC', 'Duration'],\
                                          aggfunc={'ECUhex': 
'count', 'OC': 'max', 'Duration': 'sum'})
        support55['odometer'] = odo_mean
        support54 = pd.concat([support54, support55])


    # from multindex to a flat one
    support54.columns = ['_'.join(s) for s in 
support54.columns.to_flat_index()]

    #add column fuel type
    for vin in support54.index:
        try:
            support54.loc[vin, 'Fuel_Type'] = config.loc[vin, 
'NP']
        except:
            support54.loc[vin, 'Fuel_Type'] = 'Not_Available'

    #gives the list of vin locked -> must check if training1 vins 
are in this list
    result1 = 
training[(training['firstDetectionTimestamp']>end_date)&(training[
'firstDetectionTimestamp']<target_date)&\
                   (training['SPNhex']=='0x7E026')].index.unique()


    for vin in support54.index:
        if (result1.isin([vin]).sum()>0):
            support54.loc[vin, 'Lock'] = 1
        else: support54.loc[vin, 'Lock'] = 0



    return support54
```

7.2 Statistical Additional Water Heater plots

In this section are considered the adopted methods in plotting useful statistical information about Additional Water Heater DTCs: the focus is on ECU occurrences, Telematics occurrences and Duration.

7.2.1 ECU occurrences

```
bins1 = np.arange(0,20,1)
plt.figure(figsize=(15,60))
plt.subplot(10,2,1) #no lock OC 7E023 diesel vs np
plt.title('OC 0x7E023: DIESEL vs NP with NO LOCK')
plt.hist(training_piv_diesel_0_OC['OC_0x7E023'],bins=bins1,
weights=weights_diesel_nolock, label='Diesel')
plt.hist(training_piv_np_0_OC['OC_0x7E023'],bins=bins1,
weights=weights_np_nolock, label= 'NP', alpha=0.5)
plt.axvline(training_piv_diesel_0_OC['OC_0x7E023'].mean(),
color='r', linestyle='dashed', linewidth=2, label='Mean Diesel')
plt.axvline(training_piv_np_0_OC['OC_0x7E023'].mean(), color='k',
linestyle='dashed', linewidth=2, label='Mean NP')
plt.legend()
```

7.2.2 Telematics occurrences

```
bins1 = np.arange(0,20,1)
plt.figure(figsize=(15,50))

plt.subplot(10,2,1) #no lock occ 7E023 diesel vs np
plt.title('occ 0x7E023: DIESEL vs NP with NO LOCK')
plt.hist(training_piv_diesel_0_occ['ECUhex_0x7E023'], bins=bins1,
weights=weights_diesel_nolock, label='Diesel')
plt.hist(training_piv_np_0_occ['ECUhex_0x7E023'], bins=bins1,
weights=weights_np_nolock, label= 'NP', alpha=0.5)
plt.axvline(training_piv_diesel_0_occ['ECUhex_0x7E023'].mean(),
color='r', linestyle='dashed', linewidth=2, label='Mean Diesel')
plt.axvline(training_piv_np_0_occ['ECUhex_0x7E023'].mean(),
color='k', linestyle='dashed', linewidth=2, label='Mean NP')
plt.legend()
```

7.2.3 Duration

```
bins2 = np.arange(0, 2000, 100)
bins3 = np.arange(0, 2000, 200)
plt.figure(figsize=(15,60))
plt.subplot(10,2,1) #no lock Duration7E023 diesel vs np
plt.title('Duration 0x7E023: DIESEL vs NP with NO LOCK')
plt.hist(training_piv_diesel_0_dur['Duration_0x7E023'],bins=bins2,
weights=weights_diesel_nolock, label='Diesel')
plt.hist(training_piv_np_0_dur['Duration_0x7E023'],bins=bins2,
weights=weights_np_nolock, label= 'NP', alpha=0.5)
plt.axvline(training_piv_diesel_0_dur['Duration_0x7E023'].mean(),
color='r', linestyle='dashed', linewidth=2, label='Mean Diesel')
plt.axvline(training_piv_np_0_dur['Duration_0x7E023'].mean(),
color='k', linestyle='dashed', linewidth=2, label='Mean NP')
```

7.3 Preprocessing

7.3.1 Train Test Split

```
training_piv_diesel = training_piv_diesel.copy(deep=True)
X_pred = training_piv_diesel.drop(['D4'], axis=1)
y_pred = training_piv_diesel['D4']
X_train, X_test, y_train, y_test = train_test_split(X_pred,
y_pred, test_size=0.1, random_state=42)
```

7.3.2 Outliers selection

```
iso_for = IsolationForest(contamination=0.01, random_state=42,
n_estimators=200, max_samples='auto',  bootstrap=False,
n_jobs=-1, verbose=0)

iso_for_pred = iso_for.fit(find_outl)
scores = iso_for.decision_function(find_outl)
iso_for_pred = iso_for.predict(find_outl)
mask = iso_for_pred == -1
anomalies = training_piv_diesel[mask]
mask_2 = iso_for_pred != -1
training_piv_diesel = training_piv_diesel[mask_2]
```

7.4 XGBoost Optimization

7.4.1 RandomSearchCV

```
params={
 "learning_rate"    : [ 0.01, 0.05, 0.8 ] ,
 "gamma"            : [ 0.0, 1, 5 ],
 "n_estimators"     : [ 1000, 1500, 2000 ],
 "min_child_weight" : [ 2 ],
 "max_depth"        : [ 3 ],
 "colsample_bytree" : [ 0.3 ]}

random_search =
RandomizedSearchCV(model,param_distributions=params,n_iter=25,scor
ing='f1_macro',n_jobs=-1,cv=3,verbose=3)
random_search.fit(X_train,y_train)
random_search.best_params_
```

7.4.2 GridSearchCV

```
grid_search = GridSearchCV(model, params_2, scoring="f1_macro",
n_jobs=-1, cv=3, verbose=3)
grid_result = grid_search.fit(X_train,y_train)
print("Best: %f using %s" % (grid_result.best_score_,
grid_result.best_params_))
```

7.4.3 XBGoost Maximum Depth evaluation

```
evalset = [(X_train, y_train), (X_test,y_test)]
model.fit(X_train, y_train, eval_metric='logloss',
eval_set=evalset, verbose=False)

predictions_training = model.predict(X_test)
score = f1_score(y_test, predictions_training, average='macro');
print('f1_score: %.3f' % score)

results = model.evals_result()
# plot learning curves
plt.figure(figsize=(7,7))
plt.plot(results['validation_0']['logloss'], label='train')
plt.plot(results['validation_1']['logloss'], label='test')
# show the legend
plt.legend()
# show the plot
plt.show()
```

7.4.4 XGboost Estimators Number Evaluation

```
n_estimators = [ 50, 100, 300, 500, 700, 900 ]
learning_rate = [ 0.01, 0.05 , 0.1, 0.3 ]
params_2 = dict(learning_rate=learning_rate,
n_estimators=n_estimators)


means = grid_result.cv_results_['mean_test_score']
stds = grid_result.cv_results_['std_test_score']
params = grid_result.cv_results_['params']
for mean, stdev, param in zip(means, stds, params):
    print("%f (%f) with: %r" % (mean, stdev, param))


plt.figure(figsize=(10,7))
scores = np.array(means).reshape(len(learning_rate),
len(n_estimators))
for i, value in enumerate(learning_rate):
    plt.plot(n_estimators, scores[i], label='learning_rate: ' +
str(value))
plt.legend()
plt.xlabel('n_estimators')
plt.ylabel('f1_macro')
plt.savefig('n_estimators_vs_learning_rate_3.png')
```

7.4.5 XGBoost Performance Evaluation: standard threshold

```
model.fit(X_train, y_train)
predictions_training = model.predict(X_test)
plot_confusion_matrix(y_test, predictions_training)


print(classification_report(y_test, predictions_training))
scores_training = model.predict_proba(X_test)[:,1]
p, r, thresholds = precision_recall_curve(y_test, scores_training)
plot_precision_recall_vs_threshold(p, r, thresholds)

ax = plot_importance(model)
fig = ax.figure
fig.set_size_inches(15, 15)
```

7.4.6 XGBoost Performance Evaluation: customized treshold

```
predictions_training_adj =
adjusted_classes(model.predict_proba(X_test)[:,1], 0.79)
plot_confusion_matrix(y_test, predictions_training_adj)
print(classification_report(y_test, predictions_training_adj))
```

References

1. https://www.iveco.com/corporate-en/pages/history.html
2. https://private.iveco.com/Common/PublishingImages/New-Services-Parts/brochure/Heavy-UK.pdf
3. https://www.iveco.com/en-us/press-room/kit/Pages/iveco-s-way-new-connectivity.aspx
4. https://www.csselectronics.com/screen/page/simple-intro-j1939-explained/language/en
5. https://www.sae.org/standardsdev/groundvehicle/j1939a.htm
6. https://www.ibm.com/cloud/blog/supervised-vs-unsupervised-learning
7. https://scikit-learn.org/stable/modules/linear_model.html
8. https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.LogisticRegression.html
9. https://scikit-learn.org/stable/modules/generated/sklearn.neighbors.KNeighborsClassifier.html
10. https://scikit-learn.org/stable/modules/generated/sklearn.tree.DecisionTreeClassifier.html
11. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.RandomForestClassifier.html
12. https://scikit-learn.org/stable/modules/generated/sklearn.ensemble.GradientBoostingClassifier.html
13. https://towardsdatascience.com/getting-started-with-xgboost-in-scikit-learn-f69f5f470a97
14. https://towardsdatascience.com/feature-engineering-for-machine-learning-3a5e293a5114
15. https://machinelearningmastery.com/discover-feature-engineering-how-to-engineer-features-and-how-to-get-good-at-it/

16. https://towardsdatascience.com/a-brief-overview-of-outlier-detection-techniques-1e0b2c19e561
17. https://machinelearningmastery.com/model-based-outlier-detection-and-removal-in-python/
18. https://medium.com/digital-catapult/dealing-with-imbalanced-data-8b21e6deb6cd
19. https://www.kdnuggets.com/2020/02/practical-hyperparameter-optimization.html
20. https://machinelearningmastery.com/tune-xgboost-performance-with-learning-curves/
21. https://www.analyticsvidhya.com/blog/2021/05/a-comprehensive-tutorial-on-deep-learning-part-1/
22. IDC5 Truck: Basic instructions, Trainee Manual, Texa, 2017
23. Service Manual: independent and oil heaters, IVECO, 2019
24. Service Manual: HI-Tecnhonoly & Telematics, IVECO, 2019
25. Advanced Training for the Water Additional Heater, Webasto, 2021